Speech-to-text reports from DjangoCon Europe 2015

Release 0.1

Hilary Maclean and Sheryll Holley

August 13, 2015

Contents

See the Git repository.

CHAPTER 1

Contents

1.1 How to help improve these transcripts

DjangoCon Europe 2015 was supported by two speech-to-text reporters, Hilary Maclean and Sheryll Holley.

We have been provided with a transcript of their output (so far, covering day one, the open day).

We need to tidy this up, make corrections, fill in gaps, and publish it so that it is a useful resource and a near-complete record of proceedings.

AOTV will soon be publishing the video recordings they made of talks, and together those videos, presenters' slides and these transcripts will be an invaluable resource.

There's a lot of work to do, but if each speaker takes a little time to submit a pull request with amendments for their own talk, the bulk of the work will be concluded very swiftly.

Once the conference videos have been edited and published by AOTV the work to create final versions of the transcripts can begin in earnest.

See also Joachim Jablon's start on a project to turn the transcripts into synchronised video subtitles.

1.1.1 How we should work

In the `raw_files` directory of the GitHub repository:

- `.docx` files (such as `Djangocon 31st May 15 Open Day FINAL.docx`) are the originals provided by the STTRs - they should not be edited in any way
- `raw_transcript.txt` files (such as `open_day_raw_transcript.txt`) are plain text versions of the above - they should not be edited in any way
- `working_copy.txt` files (such as `open_day_working_copy.txt`) are copies of the above - they **can** be edited

When a section of a `working_copy.txt` file has been moved to a new location in the transcripts, it should be removed from the `working_copy.txt`, so we can see what remains to be transferred.

1.1.2 Structure

Please use this directory/file structure so we can easily link from the published videos on Vimeo to each transcript:

- /transcripts
  - `index.rst`

  - /open_day
    * `index.rst`
    * `daniele_procida_welcome_to_djangocon.rst` Daniele Procida: Welcome to DjangoCon Europe 2015
    * `roger_whitaker_welcome_to_djangocon.rst` Roger Whitaker: Welcome to Cardiff University
    * `russell_keith_magee_what_on_earth.rst` Russell Keith-Magee: What on earth are Python and Django?
    * and so on
  - /talks_day_one
    * `index.rst`
    * talk one
    * talk two
    * ... etc

Introductions, questions and thank-yous should be included with each talk file.

Announcements in-between talks probably don't deserve to be kept for the published transcripts, unless they are important or particularly interesting.

1.2 Open Day

1.2.1 Daniele Procida: Welcome to DjangoCon Europe 2015

DANIELE PROCIDA: Good morning, welcome to Cardiff and welcome to Djangocon Europe 2015 our open day so I hope there are a few people at least who are not part of the main conference but have come just for today. Oh great. 2 people twice as worthwhile as 3. Put your hand up if you are here just for the opening. That's great wonderful. {applause}.

Unfortunately slightly bad timing because it's half term and middle of exams so I don't think we'll get all the sixth-formers who might have been here otherwise. Obviously it's quite early on a Sunday morning for sixth-formers and because of the debacle we had with one direction we had to change our dates a bit so that hasn't held with the timing for school pupils but any way ...

First of all, if you need help, just ask a volunteer and a volunteer is wearing a blue lanyard. Any Committee Members in here could you stand up - there is Vincent and David. Any volunteers apart from the Committee? There is Ben. So you'll see a few of us wearing these and just ask us for anything and if we know the answer we will help you.

Speaking of lanyards at this conference we have a photography policy so if someone is wearing a black lanyard which is most people that mean they're happy to be photographed. If they're wearing a white lanyard they don't wish to be photographed, they don't wish to be asked if they're sure and they don't wish to be asked why they don't want to be photographed. People have various reasons so please respect this policy. If someone is wearing a blue lanyard I think they want to be photographed as much as possible!

If you are an open day visitor and don't have a lanyard but don't want to be photographed you can borrow a white one for the day.

As every Python or Django conference we've governed by a code of conduct and the code of conduct is there to ensure everyone who attends is there to enjoy the event, that everyone feels welcome and feels like they'd like to come back

to this again. If you've got a programme booklet it's in full in there. If you want to look at it, have a look at our web-site it's one of the most prominent things on there.

In brief, it is that nobody at the event is to suffer harassment or abuse of any kind and no one should behave in a way that causes distress to anyone else. If you become aware of a problem or something bother S you please come and speak to a volunteer immediately and we have a written procedure for dealing with it. Not that we expect anything to happen, partly they don't happen because we have these policies.

So, what is a Django con?

It's an open source software community conference. It's a non-profit conference run by volunteers. When I say volunteers, even we the members of the Committee are unpaid volunteers. In fact everybody who comes to Djangocon including the members of the Committee who organise it have to buy their own tickets. That's the way it works that everybody pays to attend. The only people who don't pay to attend are people who need financial assistance and can apply for financial help for a free ticket or even for things like transport and accommodation. So, bear that in mind. It might be different from other software conferences you've been to. But it is important and it's why it feels more like a festival or a party than a conference and why the first thing people do when they turn up and ask how can I help. Yes a bunch of attendees asking how they could help by volunteering so it's a nice thing to be involved with.

So thank you to all our ad hoc volunteers who have volunteered.

This is our programme for our 6 days. It's our open day which will give you a taste of it. Monday to Wednesday we'll have our formal programme of talks not at the university but at Cardiff City Hall. Then on Thursday and Friday we'll be back here at the university for 2 more days of workshops, clinics and sprints.

If you want to look at the web-site it's at Djangocon.eu. Timetable is on the web-site in your programme booklet and is up on the walls in various places and we'll get more copies printed as we speak.

We've got various workshops there is Django Girls all day. This is another non-profit organisation that has been set up to help encourage more women into computing and software development. It's been extraordinarily successful, it's been running less than a year and already has held workshops in I'm not sure how many dozen countries.

During the day we've got various workshops in different places starting from 11 o'clock. Very briefly we have Python and Django for PHP coders not because we're trying to poach anyone.

NEW SPEAKER: Yes we are!

DANIELE PROCIDA: Yes we are ... Tracey Osborn, not sure if she is here but she's doing a Django tutorial specially aimed at people with a background in web design but not in programming. Stefan is doing Django CMS. An introduction if you are interested in web content management. Marcus is doing a tutorial for people who have some Python experience already but would like to start applying it to the web. Leonardo is offering a workshop to dive into object oriented Python for people with some Python but want to understand more. And Erik is doing a tutorial on cryptography which will be suitable for people who only have a minimal grasp of Python and mathematics.

Go to them and have fun. You should have registered for them if you want to go but if you haven't registered ask nicely at the door and there might be a place for you.

The talks are in batches, 2 before lunch and 2 after lunch. We'll try and compress the time that we have lost a bit. We have breaks at 11 o'clock, lunch will be served from 12.30 to 14.30 at Aberdare hall there will be signs for people to take you there, and there will be an afternoon break at 4 o'clock. And we aim to finish by about half-past 5.

This only applies to the people who signed up for the whole conference. Please, you must tell us which meals and dinners you are planning to come to. So if you haven't you need to come and see somebody in the registration desk as soon as possible otherwise there may not be dinners for you. This is part of our policy on food waste. We don't want to be catering for people who don't turn up for meals so we only cater for people who have told us they are coming to meals. That includes the main conference dinner tomorrow night if you haven't told us that you're coming for that it means you do not have a ticket and if you do want to come and don't have a ticket it means you will have to go on a waiting list so come and see us about that as soon as possible. If there is a mean you signed up for but you're not coming to please either tell us or remove it from your ticket on-line so we can allocate it to someone else. We'd be very grateful.

Also on the topic of waste you'll notice that everyone who registered for the full conference has a bottle - can you wave that bottle - so we're not serving, not using plastic cups or any other plastic bottles. There are water coolers. Fill up your bottle. That bottle will not last for ever but as long as you do if you look after it.

Our sponsors are setting up in the Jones gallery in the foyer. Go and talk to them. They're as much part of this community as everyone else is. Some are here just for the hell of it because they want to participate in the community. Drop by, say hello, they'll appreciate you talking to them and there will be a jobs fair so they'll be expecting to see you, a lot of them are recruiting at the moment, drop by between 12 and 2. They're making a huge contribution to the conference and we're very grateful for that, couldn't possibly happen without the sponsors, would be about 2 or 3 times the price if it weren't for our sponsors.

Finally there is Cardiff university that's a really important part of this conference and we're very grateful for the contributions they've made.

Several members of the committee are from Cardiff University. Most of the volunteers are students of Cardiff University and they've provided facilities resources and the staff so thank you for giving up your Sunday for us {applause}.

The university through the Vice Chancellors Office has provided a number of scholarships for Cardiff University students to attend conference. A number of the schools have offered funding to their own students separately or given tickets as prizes to their students so we've got a lot of Cardiff University involvement. Also got some speakers 3 of whom you'll be hearing this morning and we also have 3 very special visitors who have travelled from Namibia and that is also partly in thanks to Cardiff University so welcome to Maria, Michael and Jessica all the way from Windhoek {applause}.

If you are functioning on twitter do please mention and thank Cardiff University because they have made a huge contribution to this - or whatever social media.... That's enough from me. I am going to introduce Prof. Roger Whitaker who is Dean research in the College of Physical and Engineering Sciences. Physical sciences engineering. And do you need a - no...

PROF WHITAKER: Just going to say a few words.

DANIELE PROCIDA: Thank you Roger. {Applause}.

1.2.2 Prof. Roger Whitaker: Welcome to Cardiff University

PROF WHITAKER: Good morning everybody and thank you for the invitation to come along. I'm Roger Whitaker Dean of research for physical sciences and engineering and I'm also here in a personal professional capacity because I'm a researcher in computer science and in particular mobile and social computing and the group that I work with have made really good use of Django over the years so it's something we like to see prosper. But on behalf of Cardiff university and on behalf of the Vice Chancellor it's a really great pleasure to welcome again the event at Cardiff and it's nice to see so many people here, it requires dedication to come along, as a delegate on a Sunday morning when it's raining, so very impressed indeed.

I'm biased because I'm in the discipline but Django is really important and I was asked by some senior staff in the university what Django was and I managed to recall the tag line around helping perfectionists meet their deadlines and I think that sums it up really well.

I think also what particularly pleasing about this event and much of the activity going on at Cardiff in this area is that it is very much grassroots participation and snow balling of people's interest collectively that has led to so much activity in this area. And that goes hand in hand with what the university is trying to achieve and is much higher on the university agenda at the moment and the umbrella term is innovation and innovation in whatever form is something the university is keen to help try and catalogue and I just thought I would say to close, I know we're running behind so I just want to keep this short and sweet, but I just thought I'd give a few pointers around some of the investments that the university is making in terms of innovation in ways that the university here at Cardiff hasn't done so in the past, this is very much new activity.

First and foremost, there is a very large project going on in the order of £200 million and innovation system which is an extension to our campus on the Maindy Road site and there are a number of facilities and centres that have been

earmarked and developed to go forward around the area of innovation.

One of them on that site is the university social science park which will be one of the first kinds of facilities certainly in the UK and beyond and within that there will be a computational social science lab where people from different disciplines will be interacting with ways that we haven't seen before.

There is also going to be an innovation centre there on site, a separate facility, which will allow SMEs and companies to come in and work alongside Cardiff university staff, Cardiff university research.

The university is also planning data innovation institute which is going to be a large pan university facility that will be working across biological and life sciences, arts, humanity social sciences and also physical sciences and engineering bringing together people around data science and broadly speaking pulling knowledge from data in new ways. So these are really exciting times for innovation at Cardiff not least in the technological domain.

Last but not least I should mention my home school computer science and informatics which is developing a new way of teaching software engineering so that is going to be coming forward and advertised very shortly but something that is intimately connected to Django and this conference and I am sure there will be a way to link those activities together.

So, on behalf of the university I hope it is a great success, I am sure it will be and thanks to everybody involved organisation.

{Applause}.

1.2.3 Russell Keith-Magee: What on earth are Python & Django?

By the way, if you are just an Open Day visitor and don't have a lanyard but would like one of these lovely, member of the Djangocon Community badges, go the Council Chamber and ask for one, then you will have an official badge.

Okay, very pleased to welcome Russell Keith-Magee who is the President of the Django Software Foundation, a job glamorous as it sounds, travelled from Perth in Australia, I will let him say all the rest thank you Russell.

(APPLAUSE).

RUSSELL KEITH-MAGEE: Thank you. Well good morning everyone as I am Russell Keith-Magee, I would like to ask you a question, what do you see when you look at a computer? A lot of people look at a computer and what they see is a fancy typewriter or a fancy calculator or a fancy postman delivering your electronic messages.

What I see?

I see Lego, now a typewriter, a calculator, these are tools when the postman comes with the parcel and doesn't ring the doorbell. ... These tools are single purpose devices they do one thing, a computer is also a tool, like Lego, infinitely recon fig rabble, just as Lego can be any toy you want it to be, a computer is a tool you want it to be. You want to have an idea of what you want to do. If you need a typewriter you can make one.

If you need a calculator, you can make one.

If you need someone to deliver your mail, it can do that as well.

Anything you can conceive of, or you can programme a computer to do it, the catch of course is that in order to make any of your dreams into reality, you have to be able to programme the computer to realise your dream.

Over the years, two things have changed, if first is the size of computers the first ones were big, filled an entire room. These days almost everyone who is in this lecture hall can be carrying in their pocket a computer considered a supercomputer decades ago.

But, secondly, the way you programme computers has changed a lot ore the years as year, this was ... patch chords, you ran wires through, you can see the ladies doing.

The configuration of the chords can control the calculation that was being made, after a while, people worked out how the replace the patch chords, but then you are working close with the metal. You are working with low level instructions.

This is Grace Hopper, in this picture, holding the piece of magic, the Cobol programming manual, a machine independent programming language, the idea to express the solution to a complex problem using a high level abstract language...

Interestingly, this actually exploit it is strength of computer to make them more powerful, computers are really good at taking a set of instructions and following the rules over and over again. Converts from a high level to a machine language is a matter of following a set of rules over and over and again.

As a result you now have abstracted higher level way of communicating what it is you want to computer to do.

This gains you two things, first the concepts you can express that you are able to express easily, become more complex.

If you want to bake an apple pie from scratch, first create the universe, early computing was like that. You had to make everything from scratch.

High level language, when you bake the pie, assume flour, ... the less time you have spent time bogged down in flour, the more time you can spend making pies.

The second thing is you make mistakes they are a lot easier to find, in computing circles those are often refer to as bugs, why? Because in the early day they were bugs, this is Grace hopper's logbook, a moth stuck in a mechanism, causing the calculations to go haywire.

So these newfound powers to ability to encode complex ideas, the ability to fix problems, computers started to be used to do more interesting things, twenty years ahead, this lady, Margaret Hamilton. The reason that Neil Armstrong walked on the moon.

In this picture, standing next to the code for the lunar landing, if you listen to the transcripts, the last thirty seconds, you can hear him say, programme alarm code ... that is an error condition being reported by the computer on board the lunar lander, warning, too many requests passed to the computer and certain operations ignored as a result.

Now the reason that that error message wasn't a problem, programmed the computer to prioritise important over other operations, one of the reasons they got it right, using a high level programming language that allowed them to express the priorities in the case of an overload.

Okay, wind toward another twenty years, Commodore 64. Commodore 64. Gave me a blinking prompt, started in basic, it was called that for a reason, an approachable language compared to other that is existed at the time.

Of course that didn't mean it was great, but just better than the alternatives F you wanted to do anything interesting had to understand all sorts of internal details, drawing on the screen, using operations to write into the memory of the machine I was able to programme, it could be done. Now a lot of time that was just meant typing in code I found in the back pages of computer magazines. It was coding, my first game, sand castle type the number appeared on the screen. The length of time, every time you typed in the number, a bit of sand thrown on to the sand castle, the tide was coming in, eventually the tide wash away your sand castle. Grand Theft Auto, eat your heart out!

To me as a nine year old, I turned this into a new source of fun, it could do something tangible, that an hour before it couldn't do.

So wind forward another twenty years, so what do our tools look like today? The languages continue to be more and more expressive, the first part of the talk, Python. It isn't named after a snake; it is a high level programming language, not only does it abstracts away the hardware, but the operating system.

Where BASIC likes to call itself a beginner's all-purpose language, Python has a much better claim to the crown, descended from language, abc, designed as a teaching language to replace basic. Python replaces a lot of the properties, but extended the ideas into a powerful all-purpose language.

Now one of the reasons that it is a good teaching language, the underlying philosophy, aimed to be straightforward and not too clever, it has a set of unofficial guiding principles referred to as the ZEN of Python.

[on screen].

Now, hopefully, it is clear why these are all good properties for a language that you are going to teach someone to programme, explicit you always know what is going on, simplicity means it is easy to understand what is going on,

readability and beauty, make it easy to consume whatever it is you are reading.

But as it turns out, these principles make for expert programming language as well. A running joke that you are only half as smart when you are debugging code as when you wrote it.

Having a language that encourages you to be explicit about what you are doing encourages you to embrace your ability ... these are properties that make it easier to read your code in a year's time, but for someone else in the team, decipher what it is that you have done.

Just because Python is clear and legible doesn't mean it can't be powerful. It has really strong housing introspection, code that is running; work out what is running, why it is running and change the way it is running as a result. An object can look at another object and say, does that have property X and respond appropriately. Means you can Meta programme, write programmes that write programmes.

... the further you can abstract the more power you get, the more power you gain and the more specialised you can become. Plus, programmes that write programmes are the happiest programmes in the World.

Python comes out of the box with a library of built in tools, maths, dates and times, data compression if you have all you have got a standard install, you have got a huge amount of power.

Python community, congregates ... known as the Cheese Shop, the Cheese Shop is a massive library, everything they couldn't put into the box ... it is a huge range of packages for every conceivable purpose, reading every possible file time and interfacing with any possible system you can think of.

So when faced with the problem of exploiting a wonderful international telecommunication networks we call the internet, inevitable someone will come up with a Python package to help you to build in a website.

That is where Django comes in.

What is it? It is a high level Python web framework that encourages rapid development and clean pragmatic design.

A web framework is a library of software that abstracts out the common development, short cuts for tasks that you have when building a website, dealing with log ins and maintaining a session state. Permissions to make sure that the right people can see the content you need.

A good web frame work finds the pain points and smooths them over but not getting in the way, working at the high level abstraction, so you don't need to worry about the protocol or cookies or how the database is working, shouldn't get in your way if you want to dig into the internals to make it work.

Django, it builds on top of what Python does. If you are using Django, you can use Python.

So, it encourages rapid development, regardless of how many powerful features a language has or a framework has, a web framework is worthless if it doesn't save you time ... with Django, build websites in a matter of hours not days.

This comes out of a set of real world programmers.

When a big story broke they didn't have the luxury of a long development cycle, the tools are there in Django to make you more productive, to help you get your grand idea from your head into the world as quickly as possible.

Encourages clean pragmatic design, maintains a, tries to maintain a clean, desire through its own code and makes it easy to follow best practices in the applications you are creating. The philosophy to make it easier to do the right thing.

For example, there is a group out there on the internet, open web application security project. Who its name suggests, draws attention to the security in applications, every couple of years, publishes the top ten security issues.

One of the reason the organisation exists, many of the tools outed there for developing web software don't have a good security story, sure you can build a secure website in php, it can be done ... if you learn php by reading tutorials on the web, the chances are you have unwittingly embedded some security problems in the website, slow the pick up on the fault, having security as a default, having something difficult to do something the wrong way, ... in Django, it is the default setting, it is (INAUDIBLE).

But, if you follow the path of this resistance, if you do what the documentation is telling you to do, you at least won't introduce security problems accidentally. As the professor said before, the (INAUDIBLE).

Like I said, Django came from the pressures of the news room, to the credit, news rooms that it came out of, the small newspaper in Kansas, they didn't think of the website as the way of printing newsprint on screens, they introduced the idea of data journalism, it is presenting the information to the reader in a way that is compelling.

So you don't just write a thousand words saying the crime is on the increase, you show a map, showing you where the crimes occurred, trends over time, break them down by crime time, interesting highlights but let the user explore the data for themselves, given access creates content and perspective to the argument you are trying to make, allow it is user to draw their own conclusions, what is compelling to you, may not be to someone else the things that convinced you of the argument, won't be the connection that convinces someone else that the argument you are putting forward is correct.

For me, that is why it is worthwhile learning Django, some much of our lives are governed by data, logs, use of a resource, ... if you look at your computer as a fancy note pad, you might be inclined to take the data, stick it in a word document and be done with it.

But that is hardly a good use of that data. I am constantly amazed how many businesses I encounter, where the core is the "spread sheet" the master list of things can be done, how it is managed. This document can be used by one person, the master of the spread sheet. Nothing gets done unless you talk to the person, master of the spread sheet.

My day job...

I use Django all day every day to solve this exact problem, at its core, (INAUDIBLE) is a business, we ... a customer has a leaking pipe, every customer has an address, the business owner needs to know what is outstanding and what needs to be done.

Employees need to know ..., how much the customer is invoiced for. When I go to the company, the admin strategy, one of two things, a huge pile of printed work orders this deep, of everything they need to have done or "the spread sheet" which lists all the jobs to with done.

We are able to replace it with a website that anyone in the company can look at and anyone in the company can update. Everyone has a full history of everything that is done on the website, they can see it on the website, or sitting on the beach in Bali.

The more you look through the lens of data, the more you see day ... to be visualised in a more engaging way and while in the past turning the dreams into a reality, might be a pipe dream., with ... you have the tools to make the vision a reality if you have, if you have got a problem ... easy to start a business, the website, rent servers by the hours, pay tens of dollars a month.

But you don't want to start your software empire, you don't have to publish the...

If you need to know how to turn the bricks together, you can turn the computer into whatever you want it to be.

Python and Django, all open-source projects, doesn't anything to down load them, when you get them, you get all the source code as well. Lego kit, a set of instructions, most people follow the instructions, if you are curious, you can pull it apart, see how it works, connect it to another model.

If something breaks you can fix it yourself, if you find something interest you can share the knowledge.

Open-source...

If you find a problem ... (INAUDIBLE) ... not only can you add the new feature, you are encouraged to add the new feature and...

So that is my pitch for why you should learn Django and Python. I hope that has teased your interest. ... I hope you have a great day and I hope I can see you around the community very soon. (APPLAUSE).

DANIELE PROCIDA: Thank you very much Russell, you'll see and hear a lot more of Russell during the week and - no don't try and avoid him.

1.2.4 Mark Steadman: A web framework for the creative mind

NEW SPEAKER: Our next speaker is Mark - {inaudible} he's going to be speaking about web framework for the creative mind. Whenever you are ready Mark and as you are running out of time I'll be putting up pointers.

MARK STEADMAN: I am timing myself any way and the time starts now. Hello. My name is Mark. I wanted to talk about making the case for Django as the web framework for the creative mind so I like to think of myself as creative, I'm a decent programmer, I'm good, I'll get on to me and what I do in a tick but that's the case I wanted to put forward.

Introductions, my name is Mark, I have been a developer since 2001 and like I said I'm an OK programmer but I have lots of different passions I'm interested in, I'm a musician, I write songs, I try and write comedy songs and also make pod casts. In my day job I work for Substract. I am technical director there; I'm in charge of the technical strategy making sure things work and keep running and stuff like that. We're do you sign led organisation and deal a lot with creative businesses so it's kind of the exact right job for me really.

And when I'm not doing that I run a thing called poddle which is a podcast and entertainment network that I created for me and my friends so that we can make stuff and share it and have fun doing it and start to build a community. I have been a fan of that medium podcasting and those various arts and pursuits since I left university in 2004 and I made my first pod cast in 2008 and I haven't shut my mouth since. I'm a fan.

So, previously on Djangocon, I was at Djangocon Europe 2 years ago in Poland and gave a 5 minute lightning talk so this is my second of any talks like this so if I ramble, if it goes off the rails, you have my apologies. But stick with me.

So, 2 years ago I talked about contributing to a community via open source and I was also about 3 and a half stone heavier at the time so that was - over a good thing, nothing dreadful happened - and since then I've been able to contribute to the community and be part of it so the kind of things Russell has been talking about I've been lucky enough to do, contributed open source code via the web-site hub and I've had that stuff come back, people have had improvements to make to my code and you start to build something really cool.

So, speaking of building something really cool because this talk was an open day I'm not going to go into big technical conceits here. But one of the things I wanted to do was - I make one of the shows I make is me doing other things and sort of talking while I'm doing other things, it used to be called thoroughly distracted, and I wanted to make that pod cast entirely on my phone so I wanted to be able to record it on my phone with a head set and up load it and have it available to people to listen to and I did it via drop box and I was able to build a solution thanks to Django in apart 5 hours on an evening, got home from work thought this is what I want to do and I was able to build a working solution where I could up load a piece of audio from a drop box folder and my thing would read the meta base using - some of the batteries included in Python using some of the extensions in the Python package or the {inaudible} I was able to read the meta data, get the audio, images and text and put it into the {inaudible} for the podcast. And one of the reasons I was doing that is because I'm using this amazing language, a language that not only has a lot of stuff in it that comes free but it also means because it is such a popular language there are APIs available for pretty much anything that you want to work with whether that's twitter and Facebook which I'll get on to in a bit or it's drop box or it's some Apple web service or something else, there are Python packages available to let you work with those things because you are using a popular language that makes a big thing of having an extensive library.

When you start getting into Django itself if you not familiar with Django you'll soon become familiar with the admin. It's a content management system that comes included in your project so when you start writing your model which is how you want to store your data, you start to get with only a little bit of work you start to get a full content management back end if you like that you can start to up load your content with and there are ways to customise that. That is huge if you want to be able to build something that is based on creativity because it means you are not having to build a content management system to do all of that stuff, you get it for free, that's one of the big bonuses you get over something like ruby on rails if you are making a choice for a web framework.

Django is really good at handling media not just on its own it does it really well but also there are some amazing libraries that handle that so we deal a lot with images and when you are dealing with images whether running a store or something else you might want to - for example with poddle we have big feature images for articles and with a library like sore thumbnail as an example that lets you deal so simply with images that you can say here's my main image I want it in lots of different sizes, I don't have to know what those sighs are beforehand I can tinker and try things in my image tags and see how it looks and make different response image size whatever it needs and you've got

the tools to do that in an incredibly simple way with very little code no functional code, it's all in Django template tags and it's really simple so if you are dealing with images that is one of the great ways that you can get up and running really quickly so you can focus on making stuff rather than writing programmes.

So also you've got file browser which is a fantastic thing that plugs into your admin so if you want to re-use the same images or content, audio, video whatever, if you want to put part of it on a store or link to it on a blog post you don't have to up load same content multiple times so with Django file browser which is an extension on top of an extension but very easy to use this gives you not only a way to select a file from a server be that your own server or be it Amazon cloud storage it also gives you the ability to embed those images in a text box, you can embed those easily select the file, it also gives you for images gives you more sizes so you've still got flexibility there and it's a great way of being able to manage the content within your project.

So, one of the things that I mentioned about twitter and Facebook is trying to build a community so when you've built something cool and something that you want people to look at maybe you want to say, well, it's a small web app and I want people to be able to log in and check it out and share and that kind of thing and one of the little batteries you can plug into your project is Django social. It's a lovely, very simple way of being able to authenticate your users you are not worrying about building authentication system. Django has a solid system to use but if you don't want your user to create user name and password because sometimes people are funny about that or don't want to fill in from they simply want to get in authenticate from twitter or Facebook and say I'm happy for this site to use my profile and come back to your site then that is a fantastic tool to be able to do that and these are things you can use with writing very little code because the point is what I was able to do with my drop box example is I was able to connect lots of pieces together so I wasn't really creating anything new, I was simply stitching things together so it made my ability to make stuff a lot easier and a lot simpler.

What's wonderful about Django itself is it doesn't prescribe that you have to build things in certain ways. It gives you a bit of flexibility to say if you want to write your views which is the functional views that maps URL to something that you can see effectively, if you want to write that in an object oriented way you are familiar with or write it more functionally you've got options to do that, you've got options to write things in lots of different ways and structure your project in lots of different ways. There are project templates that exist that are created in Django that allow people to have that flexibility so you absolutely get to build it your way by - it's configuration over I don't know what the ... any way - thank you, configuration over convention. You get to host it your way and there are lots of ways you can host it, I used to use Apache and my SQL and there are people in here sucking their teeth going ... but there are lots of different ways you can do it and Django lets you do that and I then started to use a host called digital ocean which gave me a bit more speed and I thought OK I'm going to learn how to do this properly using services like engine X, which is another web server, but the idea is because Django is abstracted from that, you can choose the layers that connect these different pieces together so you are not having to think OK it must be on - it has to be on Linux and Apache and started building my site on windows because I was a windows guy before I started and someone encouraged me to use Django. Window was what I used. So because Python as Russell says is cross platform it doesn't care what operating system it's on I was able to build my first Django web-site and get them running on a server so you've got huge level of flexibility whether you use Amazon web services to host your site or small host digital ocean, you want to put it on a shared box with lots of other web-sites you've got huge flexibility to do that.

And then once you've got your site hosted you can scale it up. Django is by default really scalable. It knows how to handle a lot of things but there are also lots of great system that's plug into Django to help with things like cashing so when you view a web page and lots of people are viewing the same web-site the same home page if you like, Django isn't having to constantly go back to the database figure out how to display it, put it into the template, it goes I have just done this I'll get it direct from memory and show it to the user and that can make your web-site incredibly fast. You have huge levels of flexibility in terms of how you can configure that caching.

It's not just about caching, it's about other things in terms of scalability but caching is one of those things people will talk about.

And you've got fantastic support community in Django. There are people who are incredibly helpful. You'll meet people here who'll give you all sorts of advice; people tend to be very open. You've got places like stack overflow and the Django users Google group is useful. Also the IRC channel and there is a small plug, set up a slack community, called Django launch, it's very, very small at the moment but it does mean - I use slack every day so it's a room I can sit in and if people have questions then we're able to hopefully answer those and at least commiserate and go yet I've

had the same problem don't know how to fix it - but at least you've got someone to talk.

So that's it really. Let's see how we did. Not too bad at all I'm quite happy with that. I've got a lot of details here where you can find the various things. All the stuff I've been talking about the pod casting stuff is on a web-site called poddle and there is a load of code stuff you can see of mine as well and you'll find me bumbling around the conference.

So I want to say thank you to Django guys Djangocon organisers for letting me speak. I am visually impaired so this kind of thing is a bit daunting but I couldn't be with a better community and I couldn't be using a better platform, Django is my favourite platform to use. In my day job I use PHP, Word Press, I use a few other things, mainly those things but Django is always a thing if I want to build something that I care about in terms of or something that's a hobby, I go straight back to Django and I always will because you get stuff up quickly and you can focus on the stuff you want to do rather than the nuts and bolts of the problem. If you are new to it, get involved, it is so much fun, so thank you very much and I thought I would leave a bit of time if anyone had any questions other than that thank you.

{Applause}.

NEW SPEAKER: Thank you Mark. Does anyone have any questions for Mark? My question is you put up a whole bunch of great tools that you say are helpful to what it is you do. For an absolute beginner who've been told you want to build a web-site use Django and more or less nothing else. In my experience it is sometimes hard to find these tools in these communities so what is the best advice -

MARK STEADMAN: For me it was the Django project web-site, it was starting with the building of a small web-site with the poll and how you work with the area so I am a digital programmer but I didn't know how to work with Python this was the first time I worked with Python so it took me all the way through even on windows, it didn't judge me for trying to build a web-site on windows, it said it's fine we've got this and took you through the whole process so when I speak to people and they say where's the best place to start one of the few frameworks I can say actually start from the official web-site because the documentation and starter help are really, really good.

NEW SPEAKER: Anyone else like to ask anything?

NEW SPEAKER: {Inaudible} when did you create - a couple of month ago?

MARK STEADMAN: Yeah it was only a couple of month ago so it is still very small but if you find me around I can give you the details but I think it is Django noughts dot slash dot com.

NEW SPEAKER: Is that a hand going up over there? anything else? OK thank you. {Applause}.

1.2.5 Rhiannon Titcomb: Understanding Bezier curves with Python

Before we set up the next speaker we're going to see if we can get the light system a bit better. So with a little bit of pride and immense of pleasure I'm introducing Rhiannon, Rhiannon is a first year student at the school of mathematics and she is going to be telling you about how she uses Python to get a ticket to some here really.

[adjusting laptop and projector] It is the frequency.

Let's again, Rhiannon.

RHIANNON TITCOMB: Hi everyone, thank you for the opportunity to be here. I am going the be talking to you about bézier curves today, I will tell you about who I am and I am here.

I am Rhiannon; I am a student at Cardiff university and studying math. I am first year, this year I had a module called computing for mathematics in which I was taught how the use Python and various other things. This was my first look at coding, never done anything before, but I really loved it.

So, I did a piece of course work in this module and I ended up winning a Djangocon ticket as a prize for doing well in the course work, so that is why I have got a ticket.

(APPLAUSE).

Thanks. So, when I found out I could talk here, I thought it is a really great opportunity and I thought to make it easier on myself I would talk about what that course work was about because clearly it was good.

So, I am talking about bézier curves today. What is it? Well they curves obviously and used a lot in the computer graphics and drawing and stuff on the computers. I will show you a couple of examples. So this is quite simple bézier curve and this is a slightly more complex looking bézier curve. So to describe it, it is a curve starting at one point and ending at another and it is curvature is defined by control points which are movable and you can have as many as you wanted, that is what shapes it.

So, why are they used? Well they are the most intuitive for a user to draw in these programmes, they are very, it is easy to see where you can move them.

Okay. Yes, so, they are used a lot because they are really intuitive and to manipulate. I will show you an example in a minute. They are also really simple and quick for the computer to draw on, take it and compute it.

So I will show you a quick example of drawing a bézier curve. So, so there is a tool here that draws bézier curves, it is a control point and you can bend it and those are more control points you can see there I am moving it around and stuff.

You can do whatever you want, that is a bézier curve there, if I click on this now. Those that you can see all up and down the control points and those are what shapes it.

Yes, so, how do they work? Well here is where it goes to the maths bit! (LAUGHTER). So, hopefully this will come up. Yes. No!

It is based on something called the Berstein... polynomial ... the T as the increases over time ... so, this, you can't see it but it draws out, sort of shows how the control points can be reduced to draw it.

(APPLAUSE).

Yes. So this image here sort of shows how the control points are used to help sketch out the curves you can see as it increases it goes increases goes to each point and stretches it out. Very simply I am trying not to confuse you.

But yes. So, moving on to what I did? I decided to try and approximate these bézier curves and using an algorithm. What this does is algorithm takes the control pointings and finds the midpoints and then finds the midpoints of the midpoints and keeps going until you reach some point. At this point it will draw the curve however I chose to use it an approximation and draw the lines between the mid points so, this is what it will look like, obviously it is very rough. Yes? I will get ton why that is in a second because it is basically the more a control points that you have in this curve, the better and approximation of the curve that this algorithm will give because it will have more midpoints to find and more line segments to draw. You will see a better curve.

So, I decided to try and create this algorithm and use it and so, I made a function here that I focused on the bézier curves, I made a function that take control points and this bit here is just a sort of ... what is control points are, like the X and Y coordinates. Then next it calculates the midpoints between those, so it takes the X and the Y's of two consecutive points add the both the X's and the Y's divide by two. Then it does it again for the midpoints found and then once more, down the one point.

After this I used (INAUDIBLE) to plot all of the points that I had and lines between each the first and the last of each generation, I am trying to call it. So it will do the first control point and then the last control point and then the first of the midpoints and the last and keep going like that.

So, that is what I started off doing.

I then realised it wasn't very, it didn't show the approximation for a cue bit one, I wanted to see if it worked so I decided to extend it for a bézier curve with 5 points and I will hopefully now it will work and I will draw a bézier curve with 5 control points and then put those coordinates in and show you what it does and hopefully, you will be able to see that it is an approximation of the curve. I will draw a cubit curve.

I will try and translate this using the – I have to do coordinates to make it work. I will write this in down here, the first one is, minus 8...

Minus 8, 8.

Then the next.

So there are the control points, the first control point was minus 8, 8, then the second one is minus 6, minus 4. And then the next one is, I will call that the 2, 8. Then 2, 1.5.

Then 7, 3.

So, from this, hopefully, sometimes it takes a while to load. So bear with me.

Okay so. This is the approximation that my algorithm has given. So you can see it is a rough approximation, it has the right kind of shape and it looks like on a really rough basis. So, yeah it works, awesome.

So. I have already mentioned how you can get better approximations of the curves by using more line segments and stuff. However I notice that whenever I wanted to try and do a curve with more control points to get a better approximation, I couldn't because I would have to keep extending my curve, it was very badly written and inefficient because I couldn't do it as many points as I wanted.

So what I have been trying to do recently and it is not complete yet, I am in the middle of exams. I have been trying to write a function that using an end number of points and it will still work out. So this is the code for this. Input a list of points as X and Y coordinates that is before, so however many coordinate points you have on that, it takes the X part and the Y part and adds those to a list and then it will plot those X and Y points. Then after that. It will however many X and Y values there are, it will take the first X value and the consecutive X value in that list and then find out the midpoint then add those as coordinates to another list of midpoints which will then plot. So, I am going the show you that as well. Hopefully that works as well.

I will just use; I have got some points in there already. I think I have got 8 control points there, so I will just run that. Show you what it does.

Okay, so you can see that it has got in red are the control points and in blue are the midpoints and you can see that there aren't any lines or anything there, that is because I haven't got yet to the point where it keeps finding the midpoints, I haven't got it to take the midpoints of the midpoints, that is the next step. I am on the way there. That is good. Then, that is the end. I would have on the screen my e-mail address if you wanted to contact me and I link to I have got all the files on-line. So if you wanted to, you can down load those, I have put the piece of course work on there in case you wanted to read it. That is it, thank you very much. (APPLAUSE).

VINCE: A tremendous well done in dealing with a situation, you did exceptionally well. Any questions to ask for Rhiannon?

FROM THE FLOOR: I don't know much about the bézier curves but what is the motivation behind using ... as oppose to using splining?

RHIANNON TITCOMB: I don't know much about anything about splining. For me I wanted to use it from a mathematical point of view, which is interesting, as a maths student my work focuses on that. In terms of why it would be a good way of producing it? I don't really know. I think it is you can use it to draw, draw the curve ifs you implemented the (INAUDIBLE) I mentioned but I don't know if that is the most traditional way of doing it. It is a way as of a maths student I thought it would be a good way to take that. VINCE: Anymore questions? Let's thank Rhiannon again.

(APPLAUSE).

DANIELE PROCIDA: If you would like to find some nice fresh cold water for your bottles go back through the foyer where you came in where all the sponsors have set up and in the corridor just off there is the restaurant and at the back of the restaurant is the water cooler so we'll get some more signs out as soon as we find some tape.

1.2.6 Tom Bakx: Python in Astronomy

I'm very pleased to welcome Tom Bakx who is a doctoral student here in the school of astronomy and physics. He is from the Netherlands studying here for the next few years and he is going to be talking to you about his work in Python and astronomy.

TOM BAKX: So, in 1995 a group of research astronomers used the Hubble Space Telescope to observe a small piece of sky over Christmas, they didn't know any galaxies to be in this piece of sky and not even stars and it's a very small region they looked at, so optimistic people were assuming there to be say 100 galaxies in a single viewing and normal

people were thinking about a handful of galaxies but researchers were astonished when they found out that 6,000 galaxies were in this small piece of sky.

So, 6,000 galaxies is a lot to process and a lot of data to gather from a single image and back then there was a lot of galaxies but now there are samples of 3 million galaxies and recently has been launched a satellite going to observe one billion stars and observing that many different sources is a thing you really want to reserve to code as going through it by hand takes a lot of PhD students.

About this image it's called the Hubble B fuel image and some images are so far away that the universe was a lot smaller when the light was emitted back and ever since the light has been emitted the universe has expanded and while the universe expands the light travelling towards us has expanded as well, expanded from bluer colours towards more red colours and if we're able to find out how much it shifts towards the red we're able to find out what the distribution through cosmic time is of the different properties of the galaxies.

Now to do this researchers use Python and other programming languages, especially the open source ones as astronomy has a very big community and mostly the easy to use ones as we're pretty lazy. So, what did the researchers find when they plotted the activity of the galaxies through cosmic time?

They found that we live in a rather dull time. We're at 13.8 billion years since the big bang but 8 to 10 million years ago when the universe was 3 times smaller than it is now star formation and other very active processes peaked and these can be seen in these galaxies here but optical light does not give the full breadth of spectrum of star formation. Star formation is very important to understand why we see as many galaxies with as many stars around as to day and it is crucial for our understanding of the entire problems. However when a star forms a large clump of gas falls together to form a smaller clump of gas, gas heats up, a star forms in the centre but cannot radiate its heat away because there is dust and gas in the way and the gas and dust absorbs the light, becomes warm and starts radiating heat radiation and this heat radiation we can see in sub millimetre wavelengths but as you can see from the optical image and sub millimetre image there is no way of saying these are the same images. The optical shows a lot of detail and the sub millimetre shows 5 things you can call to be galaxies and they don't correlate to each other, the bright box in the centre of the optical doesn't correlate to the bright box in the sub millimetre and this is the thing {inaudible} statistical properties Python enables us to use allows us to correlate the optical and sub millimetre in a way that we can be sure we're looking at the same optical. The very brightly coloured spot in sub millimetre is so far away it is a 8 per cent of the age of the universe it is now, it is forming stars at the rate of a thousand solar masses per year where around us nowadays we only see galaxies forming one solar mass per year, so this a very, very active region in the universe.

Not only is it very far away, it is extremely bright, but it is being helped by something, and it is helped by a galaxy that is in the way, but instead of the galaxy in the way absorbing the light in the line of sight, the mass of the galaxy that is in the way bends the light around it and makes it appear less like this high resolution image can show us. This is recently been observed with a large observation array and Chile and it shows an Einstein ring as it is called of the lens image of the galaxy behind, image has been magnified 80 times and that allows us to peer even deeper into the universe. My PhD thesis is based around finding as many of these as possible because we can do some amazing science with this.

These galaxies are very, very heavy intense phase in their lives especially the far away ones and they're bumping into each other a lot. Like these galaxies these are 2 galaxies - in the midst of a merger. You can see the 2 spiral arms of the different galaxies swirling round each other and the structure they had before, 2 spirals has been destroyed you can only see the small rings next to them and it brings with it a burst of star formation. Now we understand star formation a bit and can trace it from elementary physics but if we want to understand how it really works we have to use Python in a different way to data analysis. We use it to simulate and this is a simulation of the Illustris model where we can see on the left hand side the optical light, the light we were able to see with our telescopes and on the right hand side we see the gas intensity where there is a lot of gas where a lot of gas is able to form. And as these central 3 sources, or central 3 galaxies move closer together they come entangled and the value you want to look at is rather small but says SSR as the third digit and that means star formation rate, covers around 100 the entire time but as the merger starts to occur you'll see it will peak to around 200 causing a phase of star formation throughout the galaxy, the galaxy we just saw in the sub millimetre is probably in the midst of a merger.

This is a small phase dwindling right now as we move closer to the present time, the number dwindles and falls below what it started as becoming not spiral galaxies but elliptical galaxies {inaudible}red, and dead galaxies because they don't form any {inaudible} and these correspond very well to galaxies we see today.

This is only a snapshot of the big simulation which I'd love to show you now because this simulation is of a cosmic scale, it shows a very significant piece of the universe and what you can see here is dark matter distribution within it you can see sort of {inaudible} light structure of the dark matter and the dark matter is {inaudible} like the web indeed and consists of 5 times more mass than the normal matter we can see. However it's invisible to optical light, doesn't interact with it, so the only way we're able to detect it is either by simulations or by the way it curves light.

Now {inaudible} the gas inside the galaxies and you can see there will occur explosions which happen at the centre. Supernovas exposures very big galaxies form very big stars; very big stars live a short time but go out with a bang creating heavier elements. Heavier elements we need to understand the formation of plants to understand the formation of water and perhaps the formation of life. In a nutshell, Python helps us understand the data we're looking at to make sure we get the most out of it and helps us figure out if the theory we're thinking about actually corresponds to the data we get from our measurements.

So, I think Python is a tool to put us in a bigger, in a cosmic perspective. Thank you. {Applause}.

VINCE: Any questions for Tom?

NEW SPEAKER: Wonder if you could dig into a bit more detail how you're using Python, it is numerical computation or how the graphics are rendering or -

TOM BAKX: Python is able to do all the things I showed you here but I assume I didn't use Python for the simulation I showed you - just for some cool images, probably used C {inaudible} because it's faster but you could use Python.

NEW SPEAKER: Specifically you say you use Python to crunch the data. Do you use any packages like I don't know {inaudible} or do you write everything from scratch?

TOM BAKX: There is a module called astro pi(?) and I use that mostly to {inaudible} data. Let me grab an image ... what happens here is if you were to just state the peak value of this image, you will get a less good estimation of the intensity the source has so you have to assume the profile you expect it to have then fit it and I use Python as a function to {inaudible} but I use it as a means of getting the shift from the red out of these galaxies just with functioning filling and do that and I use LM fit which is a different model but I'm not really into the module heavy stuff in Python. Thank you.

NEW SPEAKER: When you do simulations do you do everything on your local machine or use a computer and which set up you use like purely Python or you have some more supporting programmes?

TOM BAKX: The stuff I do myself I am able to do on my mac myself because it is not computationally heavy, it will take at most a couple of minutes, enough time to get a cup of coffee, but for example if you want to do large scale simulations you really have to go to super computers. It would take for the simulation I showed in the end to run it would take 2000 years on a normal computer but you are used to these {inaudible} because super computers? Yes, does that answer your question?

VINCE: Time for perhaps one last short question. Let's just thank Tom again. {Applause}.

A couple of announcements after we've finished that first session. There will be coffee in the foyer and otherwise one of the workshops is more or less started the Django 4 for PHP programmers. If you go to foyer where coffee is people will take you there. Finally thank all speakers from the session one last time. {Applause}.

1.2.7 Cory Benfield: Security Vulnerabilities - A Story About Panic

DANIELE. Okay just about ready to kick off. So, someone knows how to dim the lights so just to let you know, that lunch today is going to run between twelve thirty and two thirty at Aberdare hall it is on the website. I have mentioned it in the Djangocon twitter account as well. There is a map to get there. Go out of the back of the building, that side. Then you walk up the avenue until you get to the very end ... and then Aberdare Hall is in front of you. Check what times you are supposed to be in a tutorial or workshop, go at the appropriate time. The talks break will be as close as we can get it to between 1 and 2 I think is what is said in the programme? Is that correct?

Yes, the talks break will be between one and two, no talks between one and two, if you want to stay in the talks all day, that is a good time to go, if you want to go earlier or later that is helpful for the crowds.

If you have special dietary requirements if you have a vegan or if you have serious allergies, tell the staff there because some of the meals are being kept back for you, so nobody else takes your vegan lunch for example.

And that is all I have to say I think actually. So, we will David?

Yes, when you come in and introduce Cory while he sets up.

NEW SPEAKER: So he is going to tell us about a security nightmare discovered in the open-source product. Good.

(APPLAUSE).

CORY BENFIELD: I am using about eight devices. A quick warping I too too fast I have way too much material. I will have questions at the end.

You can find me at places, feel free to ask me questions at these places, if you can't grab me at the talk. I am kind of what you would call, an open-source software medium shop. I piggy back on other peoples work, I am a core maintainer on the requests project.

I am also on the ... three project and it is the most important project you have ever heard of. Maintain a few projects of my own that you don't care about, I maintain hyper, Python's only http2. I have done various bits of open-source, landed patches in open-source projects, some of them Python and some of them not.

I am going the tell you a story this is a story about one of my experiences with handing a critical vulnerability ... it is called requests. Little, it is kind of a big deal.

Anyway, like (INAUDIBLE) title it is catchy that is CVE 2015-2296 this will be a blockbuster with rights at end of the month.

2014, a long time ago, the world was a different place, we were in London, where it rained as much as here. The world was wonderful, the requests project, we had butterflies and bunnies cavorting; do bunnies cavort? Yes. Cool, I am worried they might frolic. They are bunnies, they hop around.

Everything was all right with the world, more importantly the requests project by this point still a fairly notable, had not had a single security vulnerability, we were feeling super-cocky, then we discovered that if you configured requests to website A, which we will call good site, that website redirect to another one Chaplaincy we will we all evil.inc., we will send your password to there as well as good.org.

So, we felt (INAUDIBLE).

So, yes, we leapt into action, a well drilled project and so we filed for two CVE numbers, CVE common vulnerabilities and exposures, a formalised number for keeping tracks.

So we filed two of them.

At the same time decided we should probably have a protest for these, this is a good thing to have. This is a list to look at to work out what to do. We had no idea what we were doing, we are idiots we don't deal with this stuff.

So, we kind of just made this stuff up. This is the crux of this story is, this was the error we made, we didn't talk to anyone about what we do. The first thing we did right, put up contact e-mail addresses, if you don't put up contact e-mail addresses, people will report vulnerabilities in the public bug tracker. Anyone can read them., this means the evil happens can read them and say hey, that might be a problem, I should do something with that, then exploit the users, get into trouble and then get fired. Put out a public e-mail address, let someone contact you, could be a personal, doesn't need to be anything fancy.

Along with it, if you are sufficiently technically savvy, put up a gpg keys., who knows, that is less than half the room if you don't know anything about GPG it is probably not the end of the world. If you can have them, do, that is my GPG fingerprint.

We can do this quite right like even for fairly technical project, one GPG key per developer. Rather than having a central.

Reporting a bug, either pick the one that will respond quickly or send one mail to every developer, not the ebbed of the world, it is not great, we do plan to having a project GPG key at some point in time. Those two steps fantastic, hard core, everyone can securely report bugs to us, nothing can go wrong.

All right so we fast forward, early 2015, doesn't count as now anymore, but it is no ... set the scene, the morning of Saturday 14th March, weather nothing like this. There are no wading birds here. I woke up around 7:30 a.m., I rolled over, grab my phone and read all the e-mails. Sitting at the top of the inbox, was this. So ...

Coffee has got nothing on that subjects like for waking you up in the morning. That is way, way worse, so I am panicked then I noticed the time stamp. I was already 8 hours behind, no one in the team got the this yet. I was the first person to see it 8 hours after reported. I freaked out a little bit. Started my week end a little differently to the way I wanted it. Trying to read the e-mail on a device that was good at reading e-mail.

So, reporter did something right. I want to credit Matthew for this, he produced a tarred of detail, it was probably two pages long and can take a thorough description of what the bug was, why it was a problem, and reproducible working code F. you run this, you will see the bug and in fact because it is request he went so far as to write a web server that would reproduce the problem. If you report a vulnerability, I reported one recently, found the bug via code read, I spent hours writing ... so this was great.

One caveat there, our bug was a security leakage vulnerability, so leak information to other people, if your vulnerability is code execution, don't run reproduction code that is an obvious way for someone to take over your machine, so I wouldn't do that if I were you. I mean feel free! Just don't hold my liable.

Matthew was right, this bug definitely did exist and like most relatively experienced programmers at this point, I didn't stop to think about it. I just immediately charged into trying to find the bug. So, quickly jumped into the Python debugger, within 5 minutes narrowed down the problem. I will have an interlude, I don't want to go into too much detail. I want to say what it was, if you don't care, there is an okay to put on the screen, octopus, talk to her while I explained the bug.

If you set, if a website set a http cookie, at the same time as it redistributes to the website, we will assume it is belongs to the website that redirected to. It is made worse, that some frameworks will set cookies on every request sent. We would happily send your session to wherever flask sends you, the specks say you shouldn't do that.

Right fine good, so I worked it out. Didn't tell me long to work out what the fix was, it was a simple one. There is a take away, we hear in the news, hot lead or log jam or insert vulnerability here, there are weird overflows that you can't have in Python, so easy to thing these vulnerabilities don't happen and they do. But they happen because of simple logical errors this bug here would not have been caught by any type checker that is available in Python, both objects of the same type and passed into the function, we were passing the wrong one of two.

There is nothing you can do about this, code review, this function is a hundred lines long, didn't get effectively coded. We would have been extremely lucky. That is mostly where you get the errors from.

I found the bug, prepare a fix, ship it in the next release, great. The answer is no, the second you push a patch for a vulnerability you should assume that it is public knowledge, pain in the ass to look for vulnerabilities but quick to check a log.

The news is contained I know about it, Matthew knows about Is, Ian probably knows about it. He is in America.

But, for us to find out, order our code base. The second I push this they know and can start exploiting the users. So remain calm and level headed.

I am not good at calm and level headed at the best of times, this is not one of the times. Up until this point I did everything right, I could have pat t myself on the bag, given a most mortem. I said don't do any of this, I am terrible.

Don't let panic overcome you and don't move really fast. We prepared a patch, pushed it, cut the zero relux, pushed it to PIPI and contacted ... there is a stream on the, you will want to update your stuff.

That kind of feels like a success because we are like yeah, went from a report to a fix in twelve hours, eight of which I was asleep for, soy feel pretty good about myself. This was so quick we didn't have a CVE number for this. We said security vulnerability, CVE pending. I did it the next day.

So here is the problem, that is wrong, we did a number of things wrong we should have done better. So, thing one, released on a Sunday. Don't release on weekends it is a terrible idea. Firstly, lots of your users will be businesses, most businesses don't work weekends and they are not paying attention. Attackers do not work nine until five – they live in basements with plenty of hours and ... releases patches on weekends gives attackers a day or two of head start on your vulnerable users who are businesses who don't know what they are doing., businesses are (INAUDIBLE) and those require that pretty much the second someone becomes aware of vulnerable they have to assess who is at risk, I have people following us on twitter. Forcing people to work on a Sunday doesn't enforce ... release on weekdays, preferably Tuesdays.

Don't release before you have a CVE number, this gets into the same concerns of businesses and complaints, CVE's are how businesses track vulnerabilities, whether they are exposed to them. If you don't know how to get them, Google "how to get a CVE number and great."

I did that, it is fine, super-easy to get, not that hard, make sure you get one.

If you have downstream redistributors, warn them ahead of time. This admittedly might have been a privilege for larger projects, you might have surprised how much have got them ... if you have them, find them out, form a relationship with them and make sure you can warn them ahead of time. If possible, and you are GPG savvy, have a conference, so you can definitely contact them and only them. That way you can coordinate a release over all the channels, minimise the risk to the users ... they are people too.

Also, if you put ... (NAME - INAUDIBLE) users to hate you, ... they write nasty e-mails.

Right, we will speed through. Clear what versions of software are affected and how to fix them. Just upgrade. We had to request all the back verses so people know, most importantly have a public policy about the stuff, have a document, here is who you contact, how you do it. Request, promises to get back to you within twenty four hours we will acknowledge the e-mail, then a separate time how to push for it. If you don't know how to write one, steal ours, look for the vulnerability section, steal it, maybe change the keys in the GPG keys.

Number one tip show, scar, this is good, be prepared for this sort of thing, it can happen to you, you want to know, hey, I saw a talk and maybe there was some good advice in there. Cool, I am done. Thank you very much. (APPLAUSE).

NEW SPEAKER: Thank you, sorry to cut you short.

Whilst Amit comes to set up, many a couple of questions?

Anybody?

FROM THE FLOOR: I have got one.

RUSSELL KEITH-MAGEE: Fantastic, thanks very much for that, by the way Django has one of the policies in Wales, I would be interested. Any comment how it changes in?

CORY BENFIELD: It was, vulnerability is not disclosed to you, you have become aware of it, already exploited. Everything moves much faster, my ... it was entirely inappropriate in the case of a well disclosed vulnerability, again, Alex has a really good log post on this, I recommend you read, I steal most of my good ideas from this. This is one of those, get ready to check it out.

There was another question?

FROM THE FLOOR: So first off thank you for riding requests or. Have you had a chance to put your new improved process into action?

CORY BENFIELD: We have not, I think the best kind of work is work that I do, that I don't have to use, I am happy to waste the 8 hours.

NEW SPEAKER: Any other questions for Cory?

Thank you very much. (APPLAUSE).

Amit is going the set up for the next talk, while he is setting up. I would like to mention two people who are doing a really great job for us here today, Sheryll and Hilary, here at the front from Action on Hearing Loss who are providing the speech to text transcription, so thank you very much.

(APPLAUSE).

Is this more difficult than most jobs?

Yes, it is.

So, they are doing a really valiant job and it is fantastic, if you know, if English is not your first language, if it benefits you to see this, make your way over here so you can see the screen, I hope that it is proving useful to people.

If it is proving useful to you, mention it next time you go to a conference, there are people with hearing impairments who don't go to conferences because they feel they don't hear things. So, thank you very much once again, they will be here with us for the next 4 days, I hope we don't wear you out completely. So thanks. (APPLAUSE.).

1.2.8 Amit Nabarro: RESTful APIs in Django

Very pleased now to introduce Amit Nabarro who is going to talk about RESTful APIs with Django. {Applause}.

AMIT NABARRO: Working? No? Is this working?

NEW SPEAKER: For the video...

AMIT NABARRO: Hi everyone my name is Amit. I am going to talk about RESTful API with Django today. A few words about me. I have been doing software development for a living for far longer than I care to admit. 7 years ago I read a book about ruby on rails and a saw a comparison with Django and I pretty much never looked back. I only work with Django these days.

So, anyone who has done anything with Django in the past knows that traditional Django what I like to call classical Django is based on a concept which is called model view control or a variation of it. Most of the frameworks out there today use this model and Django is no different. And if you have written even the simplest example in Django you know that views return with email, you send a request and they return HTML - - but that's not always enough because this was maybe could 10 years ago when only browsers taught your application but today other kind of devices talk your application, mobile devices talk your application, machines talk to your application applications, other servers talk your applications and they don't always want your fancy beautiful HTML you created, they just want raw data because they have their own presentation data their own thing they do with your data. So the traditional or classic model which Django was built on isn't that useful in those cases.

Well, before I go further I'd like to remind you something which you all should know probably very well that the internet works in a very simple concept which is called request response. You fire up your browser, type in a URL, that request goes to the server then the server generates a response, a response goes back to you. The protocol this whole thing works on is http. Http is a text protocol, it happens to use HTML in classical Django - - - but it doesn't have to use HTML, it could use other text based formats as well. It could be used Json, XLL, the cool formats you came up with.

So since you can do that with http and since you can return other forms of data rather than just HTML, you could be using the same mechanism in Django to return your data.

Now, why would you want to do it?

The first thing it allows multiple platforms to access your application. For example mobile devices have their own user interface, their own platforms, they're not interested in your HTML, they just want to get raw data then display it for their users, so they would like to request data in the format of Json most likely.

Other machines, servers that may access your own application and down load information from your database. That's another good reason to do it.

Modularity, separation, we're going to talk about this in a little bit.

Then the last part which is also very important, it makes it very easy to develop single page applications. Anyone who doesn't know what those are, those are Java script applications, called skateful application, the entire application down loads to the browser in one request then all subsequent requests are on HS. You've all seen those, they're very common these days and they're very useful.

So, in order to have your general project expose these capabilities you need to develop something called a web API. What is a web API? I ask that question on the web and I found this definition: an application programmatic interface to a defined request response message system.

That's great, that fits really well with the basis on which Django works on the http request response message system. So all we have to do now is to modify our views instead of returning HTML get them to return something else, return Json for example.

A lot of you might say OK fine, I can do that no problem, I don't need the risk framework or a third party tool, I can return Json no problem I can modify my review and return Json and here is a good example the most simple example which are a function of a view, returns the first name and last name of the user. This will work, this is actually valid code, and if you make a request with any tool to your web server you will get a Json which replies to the first name and the last name of the user currently in session.

Now while this is a very valid example it's really not very useful because most projects are much more sophisticated, they require a lot more capabilities than just something simple like this and for that reason most projects that involve a web API end up using some kind of a web API framework.

Now I've mentioned the term REST and RESTful API, ever since I started talking. What is REST? REST stands for representational state transfer. It's a software architecture style consisting of guidelines and best practices. REST is not a technology. Rest is an agreement. It's a contract between a client and a server. I will serve you a restful API. OK I know what that is, I'll write the client to consume your restful API.

And there are multiple ways to develop web APIs today but REST is considered one of the good ones. And in general alone there are multiple web frameworks that support the model.

Here is a couple of examples and principles of REST. The most important principle of REST is every resource is unique and in REST terminology every request you make is made to a resource, a resource can return data, and here are some examples of URL that is a unique URL for server, requesting first user or requesting user based on first name or requesting - here is an example, the last one is an example of a nes {inaudible} resource it's a resource that belongs to another resource - this is outside the focus of this talk.

Another important principle of REST is that the inter actions are state less. There is no state carried from one to another. Nothing is saved on a server. Everything has to come either as a body request or in the URL.

So, I said earlier that yes you could write your own views that return Json or return another form of text based response but if you are going to develop something that is a little more sophisticated and simple user name than the simple name of a user then you will want to use a web framework which a - a REST framework which will give you a lot - now what can you expect from a REST framework?

The first thing you can expect is see realisation, take your data, see realising it, Json. HTML very important. Pagination, you have a lot of data and you need to paginate through your data you can't obviously make a single request from all the data in your database because that will take for ever. Validation another important thing especially when you submit data if you are posting or putting data and want to validate it that's another thing you can expect from a REST framework authentication anything to do with users logging into your system. Authorisation anything that has to do with what the user is allowed to do on the system. Throttling, throttling deals with managing too many request's when your server is bombed with requests you want to slow down, you want to allow users to have only certain amount of requests per se second or certain amount per millisecond that is where throttling comes in. Caching very important. If you have a single request data being called multiple times you want to cache your response so you don't have to get it from the database then serialise it again because it's a very good concept and it's a very rooted in Django itself. API discovery that's also very important. If you are developing an API which goes to another third party user which will use your API you want that API to be documented and discoverable and you can expect your REST framework to expose its own API or its own scheme and say OK these are APIs I'm exposing, unit testing if you don't write unit test you should be ashamed of yourself. And a lot more ...

Existing REST frameworks are for Django. The first is the Django REST framework DRF, written by Tom Christie, excellent piece of software, fantastic, huge user community, very well documented, highly recommended.

Django tasty pie just as good just as well documented, huge user community, actually it's even been around longer

than DRF, and it's a very good option. Django piston is a third one. I wouldn't use it. Django piston is dead; it hasn't been maintained for a while, documentation code, just don't use it, if you happen to run into it move on.

Here's an example of something which takes a model, a Django model and turns it into a resource. So I mention before that in the REST terminology we consider our data as resources. Here is an example of taking a book model or a database model which represents a book and turning it into a - this is all you need to do in order to get, take a model and turn it into 4 lines of code. Obviously a very simple example, probably more simple than the tutorial you're going to see in tasty pie, this is done in tasty pie not DRF, but DRF is very, very similar. And what will happen is these 4 lines of code is going to turn your model into a resource which you can perform crude operations on, create up-date - all that in 4 lines of code. It's pretty cool.

Working with SQL that's not a problem at all. REST is not limited to relational databases. You can wrap your data sources with a REST framework to any data source you are using whether it's a relation database like {inaudible} or using mongo DB or any other thing. In fact on the fourth day of this conference, there is a workshop that I'm doing on mongo DB and Django and I'll be touching that subject in very detailed way.

That's it. Pretty much. {Applause} I would like to say one more thing. REST is a very big concept and I can probably talk about it all day but given that it was only 20 minutes I tried to condense it as much as possible and give you an introduction to it but if you are interested in talking more about REST I am going to be here for the REST of the conference.

NEW SPEAKER: Thank you very much. Questions if anyone is interested.

NEW SPEAKER: Which of the 2 recommended frameworks do you think is most likely to end up in the core of Django?

AMIT NABARRO: I don't know if I can answer this question unopinionated. You want an opinion I have no idea. I use Django testify {inaudible} for familiarity and no other reason. Django framework is fantastic but it's my personal preference.

NEW SPEAKER: What's the main reason you decided to use Django and not rails for your APIs except to {inaudible}.

AMIT NABARRO: Before I was doing computation I was using Python so it was just easier for me to do it but if you look at - I like to look at trends and Django is really trending up and ruby on rails is ... that's not my opinion ... {applause}.

NEW SPEAKER: My question is - ruby on rails doesn't have much support on-line {inaudible} engineering things like that, whereas Python has a lot of support on things like data management and engineering things like that. What's your take on that like?

AMIT NABARRO: It's a known fact Python has way more third party modules than ruby so you are more likely to find something in Python than - I can give an example, just not too long ago I was writing a driver for an engine and it was talking something called mud bus which is just a protocol for motors to talk and it's based on http and {inaudible} wait a minute...motor bus ... that was it. Anything else? OK thank you very much. {Applause}.

DANIELE: So while Yamila sets up, to let you know that there is a jobs fayre with the sponsors in the foyer, so if you are interested in talking to them, now might be a good time. Otherwise they are going the be here all day, you can talk to them at any time.

If you didn't get the message earlier, whatever it says on any piece of paper you may see, lunch will start at 12:30 that is the earliest we are doing lunch. If you are only here for the Open Day, sorry, we are not able to provide you lunch because we have only provided lunch for the people who signed up for it because, those are the numbers we have, but there are plenty of places to grab a bite to eat nearby.

1.2.9 Yamila Moreno: Lessons learned

NEW SPEAKER: Okay, the next talk is by Yamila, Ten lessons learned during the a big Django project. (APPLAUSE).

YAMILA MORENO: Thank you for coming to this talk. As I said, it is about ten lessons I learned in the Django project. My name is Yamila, this is my twitter and any other resources I work Kaleidos. So this talk about the

good and the bad decisions we make in our big project built in Django, it was a complete renovation of our website, roadmaps, meant for all of you who have facing a big project for the first time or maybe starting with web development if you have working with Django for a while now, you h identify issues and the questions and other things.

So it was a traditional brand with roadmaps relating to our massive oil company in Spain and we had to, to repeal the brand and it was traditional brand we needed to keep the essence, so it was quite difficult. We decided to focus on five main features were to tourist, gastronomic cards, search engine, power map, planner, editorial content. So how big was it? We had to handle eighteen thousand cards, this among is data is not big if you any of the big data stuff, we had to show all the cards in a map at the time. It was hard. Firstly, took about 18 months then took time transferring., so today, continues to evolve.

So, the original team was formed by two user experience one designer, one front end, six back end. I was part of the back end team. As well, managers, relation team, the marketing, so, lots of we had to take on board lots of different opinion to do the platform and as we were, we had this system, two weeks, so we had forty sprints, six hundred US's and thirteen hundred points.

With all of this, this was a project. One of the first things that you have to think about whether the project is big or is small is about the architecture, when you have a small project you can keep the proposed architecture of Django, with the back and the front coupled but a bigger project as the former speaker talk, will face a new challenges and bigger challenges and which will require a stronger foundations. For instance, in (NAME - INAUDIBLE) we are starting to develop the project and six months later, we were asked to make, mobile version applications, mobile applications, export the data to other applications and we were unprepared we needed to make a lot of changes to go the new requirements so the lesson one would be completely separate API and front projects. The API is stable and the front projects are changing every day.

Apart from the architecture, there are also places you have to be ready, so you need to make it ready for any new requirements for instance, we had to integrate the Google API, Google maps API so took this and isolated in our different group and two months later, we changing all the maps API because we can. So we just needed to change in this module. So, we thought of course that the effort was worth it. But, with great power it comes great responsibility. The everything, factor in everything, in the way of the dead lines of the project or in the weight of being paid working with the team. So the lesson two would be ready for changes but don't over-engineer. It is difficult and useful for these to use some rules and, my rules refactor and abstract the second time you have the chance. If you are working, you have to, within the method you are working on, but the next time, refactor the method. This allows you to refactor.

So quick survey, raise your hands all of you who know about the loss of thermodynamics so, laws of thermodynamics, now rise your hands if you know the fifth law of thermodynamic yes, I was expecting that, it is common to forget these very important law which says sooner rather than later you will make a bug! it is not me, it is the dynamics.

Being that true, the lesson is quite easy, test, test, test. The key is that if in a big project many people are committing to the call. So you need to be sure that your changes won't affect the behaviour of the application. So you need test. You need to know that your future changes won't affect the previous behaviour, so you need to test.

In, we wrote tests but not many. Not disaster, but with some of the changes something or adding new features because there are dragons, we didn't trust ourself, which is normal. So says you have in Python you have a specialised framework you have to know about knocking, stabbing, spies you need lots of resources to make your application. Testing, you need to test the application.

One of the most powerful resources when you are working with Django is the community. In the community you have lots of libraries, big and small. You have I think I counted more than a thousands. So, some of them are probably famous Django framework, request, CMS we use all of them. In as well as in smaller libraries. All the resources help you achieve one important goal which is not reinventing the wheel. Sometimes the library is too big and, very little part or sometimes the library doesn't do exactly what you need or maybe you need the absolute control.

So in those cases, you can take advantage of the adjustment of the flexibility of the power of the Django, and Python. You just need to know the internals and do it yourself whenever it is needed. It is difficult to know when to use the resource or when to do it yourself. In the map, we need everything with the map we didn't found any library who could deal with the front with eight thousand cards which were points of interest. So we needed to class them in the back end and then show in the map the, for instance, the literal content was made with Django CMS as it is open-source,

we used at the base and made some widgets and we changed some functionality, there is no one answer but just to try to be aware of the context and change the approach.

Besides the, there is about, the project when you are working in a big team, for me, it is 5 people is a big team because everyone is changing. You should follow some good practices for instance, this, use a distributed source control manager. If you have been working with, you know this, but if you are starting you don't know you really need this., (NAME - INAUDIBLE) so ...

But, whatever you are doing, you need to use it. For us the lesson was not to use the source control but to add the widget to our work flow, so we started to use requests as the basic request to have procedure, so we created a brand, then develop anything there and then when we felt comfortable with the code, the test and the recommendation we made a request. This request would be reviewed by another colleague who would so, this colleague needed to understand, to test the code and make a widget or maybe accept a request, at least two people reviewing the code before adding to master brands which means like going the production.

So use pull requests as the basic procedure to add features to master. We thought it would save time in overheads. We, we were making better code for the very beginning thanks to this rule.

Very often overlooked issue is documentation. But in a big project you, even in a small project but a big project needs to be documented and well documented. But the lesson, so, you need, in my opinion in my experience it is important to have two different interpretation, one is meant for the developer who is are going the deal with the deployment and the API for those consuming the data and they are so worried about requests and responses, and something like that.

But the lesson is not only documentation which is important but documentation driven development. The first thing we do, to write the documentation to agree with the front team which are the first steps consuming our API and then once agreed, we could do the implementation, similar to the sign by contract but focused on a higher level and it worked very good for us.

Then finally the ten lesson, when you are working with Django, you are using and you are clever because you know about thermodynamics you are using the open-source resources. These resources make your code better, make your life happier. So, it is important to give back something to the system to keep it alive to keep it growing. You have lots of many ways to make as much open-source as you can. You can report a bug. You know how to report abnormality issue now. You can fix a bug, you can write a tutorial, you can just help someone, you can release a library and for instance in Kaleidos where we work, all the customer haves to agree that we are going to release code. From the development, from the project. Also in Capacity we have our own project and we are very gad and proud to share with the community. If we want to share it and not be ashamed of our code. More important we want the community to work with us, we want to make it easy, for that we need to follow those rules and many, many other rules. So anyone can run a test and they know they are doing things okay, they can read the recommendation and they know and they can understand everything, so, so it would be easy to make the system grow.

[baby crying], he doesn't like to ...

Yes., beside the ... you can think about many, many more rules, maybe you think all of these apply to a project and they do, think about Django and the community and the research and the recommendation, you can maybe approach to make the platform or the project or whatever much better.

I think that is it. Any questions?

(APPLAUSE).

FROM THE FLOOR: How do you decide that a piece of software can go open-source for a project?

YAMILA MORENO: From my company point of view, two ways, the first is if I am using Django's framework I can then use my working time to improve the Django's framework on so this will be the first thing. It is in the my spare time, my project is taking advantage, so I can put my effort my working effort and the second one is we prepare the code to have libraries. So I can make an agnostic library, my project can use the library but not release it. If you can't release on the project it is, the most important sorry, the most big, the biggest way to share.

RUSSELL KEITH-MAGEE: Actually following up on that, can you, have you got any experience with, any advice or experience with convincing clients that it is okay to work with the code., it isn't hard to convince someone to say it

is a free library, when we build your project we are going the open-source a large part of it. A lot of companies get nervous you are going to give away secrets.

YAMILA MORENO: The contract for us is mandatory, you are working with us, if you don't agree to this, this point because we are an open-source company, so, it is a basis. We, the code that we make belongs to the customer, so, the critical points of the business logic belongs to the customer, but we are making, need a special authentication and we make a special library for this or we make some extensive library that is not about the logic. We release it and they agree, they are very happy with this because usually when you are going the set a contract with us, they know us, they know what we do and they really like to be part of it.

Maybe coolness for them.

NEW SPEAKER: Was going to ask, so when you have got deadlines looming and you have to work faster and faster, and fleshing things out, what gets cut of the best practices, documentation, practice? What can go?

YAMILA MORENO: Thank you for that question! (LAUGHTER).

NEW SPEAKER: I am the awkward client.

YAMILA MORENO: We call it the hour of the brave where we stop doing tests. Thank you!

No really, usually we just come to agreement with the customer, okay we will have this, this next release for you but the next two weeks are for us. We have, we are writing rubbish and we need to rewrite it. So that really is, we will fill the, we will accomplish the dead line but then you owe us two weeks or whatever to rewrite properly. So, it depends and documentation it is for us, for me at least is the most difficult thing to do because I don't like it!

NEW SPEAKER: So the technical dep, but you need to – find any issues, explain that to clients sometimes?

YAMILA MORENO: Yes. Not only that because we also have developed a little project also in Django, which we call ... (NAME - INAUDIBLE) whatever we increase our technical depth we put a monster in our, in the dungeon, so the client can see how it is growing this. So it is a nasty thing, if you want to clean this dungeon, you need to give us some time but the customer also has a word in the decision.

NEW SPEAKER: How do you deal with situations where you will only need to use some of an external API for example, so you build a small library that could be open-sourced but it doesn't do all of the things that other people might want to do. In my experience a lot of developers are nervous to open-source that handle 10% of the API, they think they will get complaints from the community it is not good enough.

YAMILA MORENO: I know, I mean you can go into ... maybe the 80% libraries that you need for once or twice and hopefully someone can take advantage of this and whenever it is possible we try to make it agnostic and useful but it is not always possible.

NEW SPEAKER: You would say release it rather than sit on it?

YAMILA MORENO: You don't need to take with us, if by chances, from anyone, take it.

NEW SPEAKER: How do you make sure that your internal documentation stays up to date with the code you are writing?

YAMILA MORENO: What happened with you? This is my first talk ever ... (LAUGHTER).

There are some ways, you can know, I can be sure but for instance, we also do refactor documentation, we take the time to review the documentation, for us, a very important resource is to add a new member of the team so, take the documentation and see how far can you get? So, when he is, from he or she is a start okay, my documentation starts because always it is the documentation, it is not the developer. Most, 90% of the time is that I wrote something not clear enough or something like that. But, you have to refactor also your recommendation.

DANIELE: Thank you.

So, just a reminder that lunch will be available until 2.30. Stay for the next talk and then we'll take a break for lunch from here. Unless you have an appointment booked with a workshop soon after lunch in which case you might need to go now. Lunch available until 2.30.

1.2.10 Raphaël Barrois: Cyberponies - Django talks to machines

NEW SPEAKER: Hello everybody. So, our final talk on this section, please don't go away for lunch, there will be plenty of food left. So we're going to have one more talk by Raphael on cyber ponies. Thank you very much. {Applause}.

RAPHAEL BARROIS: I have one option one is I go very, very fast and everybody will be ready soon or I can try to present things in an able manner which I'll do. Today we're going to talk about {inaudible} cyberponies, Django and machines. What do I mean by machines? Where I work we are entering car sharing system in Paris, Lyon, Bordeaux, soon in {inaudible} ... some are cars so they're good with some systems and units and some are charging points so you get a big {inaudible} ... so devices I'm thinking of, you can think of all devices, you can think of traffic lights or small {inaudible} to know what temperature is so what I'm going to talk about days vices that are full of sensors around them that you can promote {inaudible} in terms of traffic light green ... and local interaction like swiping your card on a point. And all those 3,000 cars and points with Django and I'm going to explain {inaudible}.

What do we need to do? First you need to manage {inaudible} - your inventory, what devices I have {inaudible} when did I buy them {inaudible} new devices as they come out of the factory or when they come in. Hopefully you will have to push from {inaudible} new features, basically because we want to {inaudible}.

Monitor devices. Sensors, things coming up, record everything that happens, everything, road, or ...

And you want to detect issues, sometimes you'll lose a device, drops off or perhaps the {inaudible} is broken or you want to know about it and you want to be able to tell the officer oops there is a problem you cannot transfer this car it's not available right now.

Actually use the device, {inaudible} on your board, it is there, shiny and you want to interact with it.

So you might want to aggregate information from all your devices and put them to your end-users and you want to be able to convert user request from your web-site from your API, from your internal application to actual {inaudible} the devices.

Inventory ... OK the device will {inaudible} inventory,, once a year when replace port ... but you've got serial number, model, manufacturer, mac addresses, registration number, lots and lots of numbers, versions and information to keep, and it has components and subcomponents, when you have a car, hundreds of items in the car and you want to track everything to know perhaps if one batch is malfunctioning.

So just a model, kiosk, that's my internal reference and it's maybe a new kiosk model ... {inaudible} small kiosks, {inaudible} then you've got your actual {inaudible} you install somewhere ..., it has a model, belongs to a customer, it's somewhere in France. Basically you want to describe your objects Django will be pretty efficient for describing whereabouts. Put them in and you already have your whole database. Just with Django.

So if you want to {inaudible} perfect. If you want history you can use various libraries for instance I think Django reversion keep track of changes, you can use libraries for recording building nice graphs. You can build a simple APIs {inaudible} information system can access that data. Your devices can push inventory database and request for API.

OK a few things Django cannot solve. How do you registering inventory. If the board is about to scan all parts all serial numbers and push - {inaudible} provided you have device and a huge {inaudible} ... have to scan them manually, and when you want to push on upgrade you want to push up-grade with Django, perhaps you say with Django I want the device to be updated to that from a version but actual {inaudible} on to the device is going to be something else so yeah you have to do something else.

I'm looking for some very, very important thing when you have lots of devices ... internet... {inaudible}. Security, you want something that says when I am here - it is the right device but it is giving you serial number, someone trying to fake it and when you send data to your devices you want to make sure it's the right data and it's not been altered and it's basically your proper data and someone is not {inaudible} ... that's a few important things when you will have to rely on other tools via Django for instance useful for your place, lots of tools are coming in recently. So that's more what it is what Django is, the perfect tool because it's {inaudible} devices, you don't want to run Django on your device.

You want also to monitor devices. Let's say for instance device crashed OK it crashes so {inaudible} it's not working. OK. What happened? Why? I need to get the last 5 minutes usage or things like that. I just push a new {inaudible} to all my cards, are they performing better, worse? I have to know that. Actually the devices are same source with the charge points, how do we know whether a car is parked in front of charge point? We need to get that information to know whether we can {inaudible} the charge point or not. We don't have eyes on the thing. Our sensors our monitoring - only way we can know what is happening in the interface. So yeah be prepared to get a huge amount of monitoring traffic. Sometimes if you get a message once every 5 minutes but 3,000 devices it means you get 10 updates per se second. Say you wanted to have one once every 5 seconds want to switch to {inaudible} some sort of system so take that into account when deciding your network and system. But actually if you just want to spread the load of {inaudible} images, send them to the database no problem, {inaudible} application servers running Django because Django is designed you may put several {inaudible} database so if your database is big enough you can {inaudible} ...

A table {inaudible} around {inaudible} lines right now, we can query it pretty fast, postgres is amazing {inaudible} ... elastic, choices to choose from.

That's what I advise is to get all your requests through Django that may if you want to get for instance information about how do the latest engine perform you can look through {inaudible} recorded, {inaudible} then ... monitoring through {inaudible} through Django ... put that in your report.

A few important things to keep in mind when you are distributing system. You will lose messages. At some point they will be full or have to miss messages or some will be broken, you will lose messages. Some messages will arrive late because while the car was parked in another {inaudible} then it has to send 5 hours of data at once. Sometimes you'll have some issues because your devices have clocks of their own which aren't synchronised so one says this happened 2 p.m. oh actually it was 2.30 when it happened. You won't be able to do anything.

So tips. When send message to database just put {inaudible} that may if you have lost an event well, you'll catch up later on next message.

Send the time since last change for all binary sensors. I know what happened when it happened and I can fix my data. And what we found very useful is to keep a last known state cache in the database which means we have one line instead of in? Device so it's much more manageable and have access to all the current state very, very easy. Good, reports ...

OK now we've got our inventory and we want to use our devices at this point. Yeah one problem you have is your users they want information about for instance {inaudible} home. Sources in their home. So data to provide a global view of the situation and they have lots of devices to consult that are on. Mobile phones, tablets, they've got their web-site, so you want to put all that information in an agreement so you use {inaudible} proper {inaudible} for err users where Django helps and you want to send commands to your devices when someone acts on its interface, participant, or when your computer needs to reboot or reset some device. Well, here it helps with Django because for instance let's say I want to reserve a charge point I want to make sure it has {inaudible} received monitoring message recently and it's not used but can create reservation but keep in my Django database for anything else then I send a manual command, blue, that's the logic to charge point and here use the users, I want a reservation, you do that from Django, check the form and you just code your process, {inaudible} user swiping a card.

Huge changes to solve. You cannot send request from {inaudible} to devices ... here it is more complex - split brain effect which is what happened last time it received message from device. Partial view world has changed. You cannot know what has happened you have to guess and have to design and keep that in mind when designing your apps.

That's all so do you have any questions? {Applause}.

RUSSELL KEITH-MAGEE: Are you running Django on both ends of this on the server, devices or just on the server...

RAPHAEL BARROIS: Just on the server. {Inaudible} on the devices but Django {inaudible}.

NEW SPEAKER: How do you send out {inaudible} devices the software permit of it.

RAPHAEL BARROIS: For now we're using the {inaudible} packages so {inaudible} should be now using version of that much of our meta package so it starts a new package with all occurrences of all versions. It's not perfect but it works for? ? And we're looking at oceans like {inaudible} snappy which is going to have {inaudible} devices.

NEW SPEAKER: How you deal with tests like for instance you have a device that sends its state, how do you make sure if you use {inaudible} for instance that the device doesn't change over time that you're using the wrong {inaudible} for instance?

RAPHAEL BARROIS: We are building tests where we are running basically the whole ward on the {inaudible} we basically send fake messages to fake device {inaudible} sends a message to the app and back again and so we can {inaudible} like that and build for complex scenario but it's kind of tricky and so we've had project to design simple ways of running full integration tests. We don't have to run the actual device code which will send {inaudible} want them check the device runs properly and we use some database working to ensure we don't have {inaudible} for the same thing at the same time.

NEW SPEAKER: How do you detect that some sensors are broken?

RAPHAEL BARROIS: Broken sensors for instance we detect that a sensor is changing states too fast, for instance when a charge point says hey people have been connecting disconnecting 1,000 times in one day you think it's not physically possible so it's broken. {Inaudible} at all in the time where it should have because we have a few different {inaudible} for instance got to open the charge point to get access to the cable so never opens while you connect disconnect the cable probably is it's broken {inaudible}.

NEW SPEAKER: {Inaudible}.

RAPHAEL BARROIS: Well, for our next generation vices using web {inaudible} which allows us to send messages direct {inaudible} registration time for comments we want to send to devices. {Inaudible} more connected state. It can work for charge points for the cars. It's going to be slightly harder. Good. Contact information and if you have any questions about this or perhaps {inaudible} I'm making and perhaps {inaudible} feel free to ring me any time. {Applause}.

NEW SPEAKER: We're going to break for lunch now. Be back with the timetable by the time you come back.

(APPLAUSE)

(LUNCH)

1.2.11 Árni St. Sigurðsson: Data-driven democracy

ÁRNI ST SIGURÐSSON: My name is Árni St Sigurðsson and I come from Iceland, I have a Bachelors Degree in computer science from University of Reykjavik. I have worked at 2 of the 3 largest newspapers in Iceland doing work on their websites. I am currently working on my own start up doing some consultations and freelancing. My background is mostly developing websites and APIs using tastypie and I think of myself as a programmer although I have become more intimate with bash and operations than I was comfortable with at the time.

I'm going to tell you one level of the developing story of democracy in Iceland; there has been an initiative during the last several elections to get candidates for elect office to fill out a standard questionnaire and using that data to fit a voter profile. The last decade has seen turmoil sweep Icelandic politics. Economic catastrophe rocked the country in 2008 and led to a popular protest which ended with the resignation of the majority and an election that shifted the political landscape severely culminating with a historic indictment and conviction of the former Prime Minister for not observing laws on - I don't have a translation for it. It's laws on good government that are supposed to ensure that ministers do their best.

It would not be an overstatement to say the national psyche has been going through a schism. There was a general feeling that a change in leadership would not be enough. The faults were fundamental and systemic and nothing short of a total system reboot would fix the problems. Calls for constitutional reform were answered with an Athenian style assembly of 1000 randomly picked citizens from the national registry. The document was to inform a Constitutional Convention to grant the aforementioned reboot. So this is in Icelandic I couldn't find the graphic in English but this a word cloud of the document produced from that meeting.

Traditionally we have had 2 sorts of elections in Iceland. {Inaudible} watered down by party affiliation ... oops ... one should not get too technical! ... Yeah so it's usual party affiliation, you mark X for a party and it's a simple sort

of election. The constitutional convention also posed to be non-political, not partisan acted, so mostly they didn't interfere, but we had 523 candidates for the 25 positions of the convention and this represents the problem, if there is nothing like the elections they had participated infer for, after complaining of lack of choice for decades too much democracy was perceived as a problem and caused confusion. In essence it came down to making an awkward choice of up to 25 individuals from the candidates and we just scrolled through the short bio and position on the constitution and image of candidates of those 523 candidates so you can imagine you know about 50 people, about 20 people very well, you are about to select people who are going to lay the foundation for your democracy, you want to make a good choice but 523 candidates, that's a lot of choice.

DV saw an opportunity to field an experiment. Being one of the most popular news web-site in Iceland known for hard hitting journalism and having no political slant in its coverage made it a good choice to run it. You have probably had to fill out Likert scales - this is an example question. They come in varying forms but usually have an odd number of options, for or against a proposition. If you have data in this format you can measure distances between any 2 profiles by summing-up the difference of each question.

A couple of things helped the project along. There was no competition in the space and there was a general consensus politicians should not interfere with anything pertaining to constitutional reform, the size of Iceland being only 300 thousand made it easy to publish the solution. 447 candidates answered. That's 6 out of 7. That's a very good return I think.

If we thought about averages then you would assume that about 7 of the list of 25 would not have answered the questionnaire but it turned out that there were only 2. So, it would seem to be a good strategy for anyone putting forth their name to actually participate.

Another assumption we can form from this data is that also I ran a comparison what should the - I'm sorry I lost my trail. Best I can give us why people's votes were not similar to our recommendation is name recognition proved to play a strong role, most of the people elected were very known to most Icelanders. The results of the election read like a who's who of political activism, with wide range of opinions among candidates and most people agreed those elected would represent broadly the will of the electorate. There is a separate story to be told about the aftermath but the short version is the work progressed quite openly with relatively little interference from the political class resulting in a document which got a positive response in a referendum. The draft was submitted to government and has been a hostage to political class since showing change does not come without its fair share of work.

DV ran another such project for the presidential elections in 2012. It's rather unusual for Icelandic politics to see a sitting president being contested. There were 5 challengers and 2 of them strong contenders. 24 questions focused on the powers and image of the presidency 36 thousand readers participated while 163 thousand ballots were cast. The results were more predictive but there was still this pattern emerging that people would rather have a vote counted for a winner than vote their issues purely. There was an anticipation that the 2013 parliamentary elections would be historic, the mixture of apathy hope and fear was palpable in both conventional and social media, among punditry, politicians and the population. Instead of the usual 6 parties to choose from this election saw 13 parties put forth a slate of candidates. Each party has to turn in a list of 1890 sponsors to qualify for each district that's 25,000 politically active citizens out of an electorate of about 240 thousand. Any party receiving less than 5 percent of the vote nationally will not gain a member. This rule has been criticised for being too exclusionary. You generally don't get regional slates unless they're recognised as a splinter from a faction with a national relevance. It poses a challenge to any new party. We saw indication in presidential election numbers that people would rather have a vote count than vote for their political positions. All these issues illustrated to us the importance of an election app. This time there would be challenges we have not tackled before. The first logistical problem was data gathering. elections in Iceland are rather short compared to US a little longer than in the UK. Liaising with established political parties would be easier and infrastructure is in place and political operatives seasoned. A multitude of novices however could be hard to tackle. With those data we would still need to run a campaign among the candidates to answer the questionnaire during the height of their campaigning when they were busy doing other things. We featured 63 questions with the option to write additional comments on each question. We gave the candidates an opportunity to write posts that would be featured on the election front page and answer a more casual set of questions. Building up the drumbeat we also hosted some of the more recognised candidates in our editorial room making a news item out of who now had been added to the pool of answered questions. We also have battle fact that some of the established politicians refused to take part to you due to the media coverage done by DV. Remember hard hitting journalism - that's something that actually worked against

us there.

The stark takeaway here was the currently governing parties are the ones least likely to participate in the project and the only parties whose leaders abstained. The Pirates were the first party to have all candidates supply data and only new party to get members to Parliament; 11 per cent of the population voted for a party without getting a representative. Of those 8.6 per cent voted for parties that had enough national following for a representative if not for districting and 5 per cent cap. It's very ironic more democracy led to less democracy.

Instead of discussing the municipal elections of 2014 I want to give a brief overview of my architecture at that time. If you want to replicate a project like this it would be a good place to start. Django preaches reusuable apps and delivers. you can do pip install, edit some settings and some URLs and build whole products mostly from the command line. Messiness comes about when apps become interdependent as they often do in media. I elected to make an app that abstractly dealt with a list of questions, questions with attached answers and texts attached to questions. This was necessary to be able to have question span municipalities, be fluid in the event of having to change question wording and to be able to build a separate representation layer for the questions. The last feature was not used but any bilingual country would need that. Each model contained an app field so separate apps could be built to represent each election. It would be relatively easy to build a new election web app for each election but if I were to take one more run at it I would build the election app were making an election website would be a push button affair. We made a single change in the algorithm from the year before. One prankster decided to check which party best fitted a known nothing approach and filled in all the answers as "don't know". It also worried me that it would be a good strategy for a politician to answer only a few key questions but have the majority be "don't know". I fought for a default distance of one for any "don't know" answer irrespective of what position the voter held, penalising not taking a stance. This resulted in a candidate calling our office furious with a question: how come I don't agree with myself? I had to summon my straightest poker face explaining to him 2 people who don't know anything about something are indeterminately far from each other but the closest approximation I could come to without being harsh was to say that they probably don't agree.

This is probably more politics than any of you expected but most of the time it's the subject matter that makes the tech interesting instead not the other way around. Voting in a secular democracy is a sacrament. Voters are being drowned in data. For democracy to empower the citizenry, they need tools to see through the fog. I encourage you to light a lamp and show someone the way.

Thank you. {Applause}.

NEW SPEAKER: Can we say thanks to Arnie and has anyone got any questions about this? Yes?

NEW SPEAKER: You said that the position of the newspaper within Icelandic politics was very neutral. Is that all you used to make sure people could trust it in technical aspects...? Did you do an audit what you were doing to slim down a bit more?

ÁRNI ST SIGURÐSSON: Well, it's more of a culture thing. At the time at least, this newspaper would have been trusted to not take a stance on politics. People were mostly unhappy with coverage of rape cases where they sometimes named convicted people. There was this big case several years ago where a man living in rural Iceland committed suicide on the day that his name was on the front page, so it's hard hitting in that way. They took a lot of flak for it but kind of vindicated several years later when a lot of men who were boys at the time came forth and confirmed that the guy really was a sleaze ball. We didn't have to worry about people not trusting the result. There were a lot of spin-off sites that were using our data to do other things, to do things like rating each question for a political compass and then doing a graph of where each party was.

NEW SPEAKER: So the data was opened is that accurate?

ÁRNI ST SIGURÐSSON: Semi-open. You could actually browse through each candidate's position. We made it a small booboo as well (with respect to openness). For anyone that shared the test the first couple of days the Facebook sharing actually showed how they answered the test so we quickly quit that - and I was kind of amazed that the government agency for privacy didn't actually contact us and do something. But that was an honest mistake.

NEW SPEAKER: Anyone else with questions?

NEW SPEAKER: Do you see technology as being a good force for use for creating a digital town hall?

ÁRNI ST SIGURÐSSON: Excuse me for a town hall?

NEW SPEAKER: Yeah a way for the public to directly engage and drive decision making in a democracy?

ÁRNI ST SIGURÐSSON: Yes, yes we need this, we need all sorts of projects on democracy and we need funding for people to actually just go ahead and do these projects and publish them not as a vassall of some company, not as a vassall of - not as a wage slave, so that more journalist outfits are going to run things like this. In my opinion technology actually opens a whole new avenue of investigation in democracy. If you think about representative government we're actually using the worst form possible at the moment. We're sending someone to Parliament or any elect office for some amount of time and we have no say in what actually he does. So, we need to be able to make a recall, you know, say obviously this representative is not representing us and I actually have an idea. I mean think about it. Think about elections. Think about Facebook. Why aren't elections Facebook basically? why aren't districts groups of people that collaboratively pick somebody from the group who go to Parliament? If the representative is not delivering on his or her promises, then the group can just recall it. I mean, it's relatively simple to do something like this. You could have a layer. My favourite idea is something I've been calling an emergent congress, where at the first layer, you have to convince - you'd have to sign up at the national registry for the election, I want to be in Parliament. And you have to then - you are randomly assigned into a district and you need maybe about 10 people and you have to convince them that you are supposed to win this round and progressively you let people from each group meet people from other 10 groups where you also have to convince everybody else that you're the guy who goes to the - and then you finally have something like final decision or something that the whole group can then vote between, I like this guy, I like that guy. So, if you couple this with recall option what's the probability somebody is going to get chosen who will not do the will of the people? That's very low. That's actually you would say non-existent because this will presumably be a prestigious position people will want to keep and they're not going to keep it if the ones that elected them aren't - well, if they're going to issue a recall and that's the end of your term because the probability of you maintaining the position amongst the people that sent you there when you didn't do what they wanted you to do is low.

NEW SPEAKER: How do you think this can happen in other countries?

ÁRNI ST SIGURÐSSON: Something like the thing I was doing.

NEW SPEAKER: Yes in Iceland right.

ÁRNI ST SIGURÐSSON: The tech behind this is simple. There are obviously things that are going to bite you, if you do something wrong for example, but basically, the first implementation of this was just a model with each question being a field and form model to collect the data and then it is a matter of having an infrastructure to compare, so, you know, after this talk you should be able to pull together in very simple terms something similar. So if you want to do it locally, give me your e-mail, I will be in contact.

FROM THE FLOOR: Thanks.

FROM THE FLOOR: What about hacking and security, that is usually the main thing when we talk of politics, there seems to be no solution.

ÁRNI ST SIGURÐSSON: So you are worried about some of the spoofing candidate for example?

FROM THE FLOOR: Yes.

ÁRNI ST SIGURÐSSON: Well it is not relevant here, it is relevant in everything we do, we should of course worry about security, we should worry about access controls. But, it is not a part of this discussion. It is a part of what we have to do generically for example, we have a pay wall, I am no longer working at DV so, they had a pay wall. If the pay wall doesn't work, you know, if people can just smuggle themselves inside the pay wall we have no product. That is where you solve the security issue in the first place.

FROM THE FLOOR: Special problem with security and boarding, you can't mix security and anonymity. So it is not the same problem as the pay wall. Because the pay wall I can look back at my payment and I can know if there is my payment but, by default, definition, if you make them the voting public.

ÁRNI ST SIGURÐSSON: I am not advocating e-voting. Nothing about this is e-voting, all of this gathering data, filtering data and doing something with it.

DANIELE PROCIDA: I think we will have to stop there, thank you for your questions, thank you very much. (APPLAUSE).

1.2.12 Adrienne Lowe: Coding with knives

DANIELE PROCIDA: We are going to give folks some time. I am Adrienne, I will be speaking, thanks for coming to my talk.

Starting in about 5 minutes.

All our technical crew in place, ready to go, you are already to go with the projector, smashing. Welcome back, I hope you had a nice lunch. If you didn't have your lunch you have got just a few minutes to run up the road and get some. Really pleased to welcome Adrienne from Atlanta Georgia.

(APPLAUSE).

ADRIENNE LOWE: Thank you so much for coming to my talk. I am Adrienne, thanks for cutting your lunch short. It is an honour and a great pleasure to be here, like Daniele said, I will tell you about what it is like for a professional cook to get into coding. How cook and coding are not that different.

My goal, if you are new to get started with Django, to tell you a few lessons.

Before I get started I want to say thank you to Jamie and for the foundation that enabled me to come from Atlanta means so much to me that I get to come over here and share a bit of my journey, I am proud to be part of a community that values diverse perspectives, believes some of us who get started with Python and Django, have a unique and meaningful things to say and given a platform to say them.

I hope you that feel you feel the same sense of respect and welcome, if you are coming for the open day. It is something we are working hard to engender, as you are here, please feel invited to get curious to ask us, to share your valuable perspective, we really deeply value the input of the new coder, whether you are here for years you are welcome.

Something I will go into later, it is remarkable how often talking something through with someone new helps me gain a greater sense of the subject at hand. I think this applies to cooking and programming.

Thanks for having me here from Atlanta Georgia. This was supposed to be the point where I mentioned my southern accent but, this morning, on the way here, I got a cab because I was running a little late. I didn't want to miss the first talk. The driver was like, so, cab driver, where are you from? Canada right? Oh I was like oh, I am from Atlanta Georgia, maybe I should give you a sense of what I sounded like when I was small.

So, maybe I will read the PyLadies motto, back home, I am the co-organiser of the....

In my 7 year old accent? Fair. Good.

Our mission is to promote educate and advance a diverse Python Community throughout reach, education, conferences, events and social gathering. (APPLAUSE).

That is really what I sounded like when I was little.

But I did drama in high school in college and ...

NEW SPEAKER: Can you teach us how to do that?

ADRIENNE LOWE: Yes, I did a production of Romeo and Juliet and I had to do a Cockney accent.

So I mentioned back home, co-organiser of a PyLadies HDL and if you don't know what PyLadies is, I will give a quick plug for it, an international mentorship programme. Helps women become leaders in the community.

For the first time we will be Django Girls to Atlanta because of me, there are a lot of people. I am an advocate for the group.

We are here at Djangocon, maybe curious why I am using the word Python., a refresher, Django is a free and open-source. ... programme that is use Django that you may have used already include the social science pinterest and ... ... discuss, we here like Django because it enables us to build applications rapidly, cleanly, securely with the minimum of

repetition. Also superfast as the Django Girls workshop proves time and time again. Even someone without experience can build a Blog about a tutorial in a Blog in about a day.

So now that we have got some of the terminology down I want to take a few moments to tell you about myself and share what it is like to come from Django, from an entirely untechnical background. I grew up on a family farm in Georgia where we always had a huge garden this was a source of a lot of contention for my parents because they are about 250 ochre plants in the image, no family of 4 need 250, but my dad thought it was a good idea. That is what you see, 250 plants. We all love food, I started cooking at a pretty young age, went to college for history and philosophy. I started working in kitchens full time, working up to 80 hours a week. I moved to grad school, I went to a Quakers Seminary. Frequently cooked for one hundred person community mill and wrote my Masters on the different interfaith conceptions of hospitality and how different faiths think and share about food.

Grad school was where I used free and open-source software. Our lab was ... which I think was really cool. That is not common at a lot of American Universities, most are Mac everything, so cool we were using to (NAME - INAUDIBLE). Since finishing, I love this image from this year's ... since finishing grad school, worked as a personal chef, doing many consulting for restaurants, lots of restaurant related stuff private and group cooking practices. I came through coding through the passion for open-source software as a user, I want to help women in a nontechnical backgrounds get started on the journey with programming. Along with the rest of the folks hosting this conference I believe that you do not have to have a degree in computer science a background in math or engineering or really any experience coding to get started with Django today. But there are a few things while I have getting started I want to share with you today, might save you a little bit of time.

So first, I want to emphasise how helpful it is to source your learning materials, your tutorials from a wide variety of perspectives and to remember, this is very important, to be very general with yourself if it seems like one tutorial isn't working out for you. When I started learning Python, I did myself a tremendous disservice by using one tutorial and one alone. When that tutorial wasn't really working out for me, I thought, I was the problem and that was very discouraging. I started to doubt my ability and to ask myself if I should even be doing this?

Let's contrast that with the experience of being in the community. So you are learning to cook and you have to use recipes, don't listen to anyone who doesn't tell you they use recipes, anyone who is serious has to use recipes. You want to learn to cook a particular dish. You get a cook book, what happens when you read it, you realise you don't really understand the way the recipe is written or what the authors are saying and in general it doesn't make sense. Probably just get another cook book right? That makes a lot of sense. This bakery book didn't work, so I will try this one instead. That seems like a sensible thing to do, as a new coder, I thought I was the problem. One teacher's method wasn't working for me, and since he is the expert right, like I am must be the problem?

Wrong. Just like you had you may have had a teacher in grade school who you really connected with, who really inspired you, who helped you understand material that you found really difficult. You will find that some resources you connect with better than others. That is totally okay.

You are not wrong. You just need to find out what works and you will. There are really too many great resources outed there for anyone to drag themselves over the coals because one is not working.

You get another cook book. The second suggestion, please let us know if something doesn't make sense and reach out when you are struggling. When I struggled with a particular tutorial, I made a sat tweet on twitter, it was the first time I had ever done anything like that. I was really nervous about it. I didn't want to seem pathetic or didn't know what I was doing. But the truth is, like, I was sad. I was anxious confused, doubted myself. That tweet happened to get noticed by my friend Anna, incredible friend she is an inspiration to me and a leader in our community. Co-organiser of Django Girls and Djangocon, you should follow her, there is her handle.

She is a chapter leader of the new PyLadies remote. Which is something we are working on. Runs a, shares interviews with women whose stories of using Django, sometimes from a nontechnical background, if you want more experience, check that out.

Anna is also self-taught from a nontechnical background. It reminded her of how she felt when she first got started. Wrote me a long thoughtful e-mail, it is okay if one tutorial is not working for you, shared her stories and shared a long list of new resources to try.

So if I hadn't taken to twitter to let off steam, perhaps this interaction never would have happened, our friendship wouldn't have an opportunity to be deepened. Maybe I would have ended my studies.

Come on in, you are welcome here.

Hope lunch was good if you had it.

But I tried certain things, I watched "Amazing Python talk" signed up for a course, already done it through Georgia tech on music technology, we used Python at the end. I didn't think to look into more of Python. She gave me some good suggestions there, I finally started reading, how to think like a computer scientist, I got the help to move forward.

The reason I want to remind you, when you are first starting out to code, it can be really hard to reach out to you the more experienced developers and in general to let people know that you are struggling. Sometimes you feel like the risk of exposing your vulnerability or who you really are much greater than the potential reward.

So, as a new coder you keep to yourself, you say, oh no, no, I am not going the bother my more experienced friend about that because I have received that you are so much smarter than I am. We minimise ourselves, we say, oh this is probably some little thing that I should already know and, oh man, why don't I already know it? Maybe I shouldn't be doing this?

So like the example of hitting a dead end with a particular tutorial, not reaching out, you can feel insular and isolate yourself and it will cause you doubt your potential and purpose. It will keep you getting from what you need to continue on your journey with Django.

So, I want to share that I found that in every case where I have taken a risk and shared when I was struggling this community, you guys, have poured yourself out to me, you have been there for me, you have showed up in incredible ways. It wasn't in my Blog post in coding with knives, that got noticed by Django Europe, finished up in my invitation to speak here, if I hadn't taken the risk, perhaps I wouldn't be here today. I can't think of a more powerful personal example to share where you are and ask for help when you need it. Who knows what marvellous opportunities await you? Please take my word for it, speak up, say what you want to make do, dream, especially when you are struggling.

If you do feel that you are burdening someone by asking questions, the better you get, the better he/she will get and the better altogether. This is because we are asking what feels basic to you, gives someone an opportunity to flex their teaching skills. It is mutual learning, you get the answer and your teacher learns more to hand from their interaction with you.

I, as experienced cook, I am so far removed from the experience of being new, I move too quickly through my explanation, take for granted that someone understands, it is great to ... never feel like a waste of my time. Help me rethink my processes and procedures and help me to be a more effective cook and educator.

My third piece of advice, have a goal for yourself, sometimes this manifests as a project you want to work on or build. It may be a more general goal but, you can use it to refer to and use as guidance during the very inevitable periods of discouragement and de-motivation.

The thing about teaching yourself to code and from what I understand about the experience of programming in general, professional programming in general, is that you are going to have moments when you feel like such a genius, like you are going to do this tutorial and you are like, oh my gosh I got this. But I think, also be in equal measure those times when you feel like you are not getting anything and nothing makes sense and maybe again you will start to doubt yourself.

I just like this. Somebody else used it in a presentation, it resonated with me. You may think your goal looks like that, but in the middle there is tangly bits. Likewise cooking professionally is hard you guys! there are lots of opportunities that through your spoon go in your hair and storm out of the kitchen. I am not sure what the working conditions are in the UK, in the US, kitchen work is notorious for 4 things, long and inflexible hours, bosses, no health benefits and really low pay. So, it is not very appealing work.

Yet, millions of us still do it as servers, line cooks, food runners and yes even chefs. I think it is because whether you cook professionally or for your friends at home. The most satisfying, we simply love seeing the delight on your face when we put something we made down in front of you. I spend way more time cooking for people – I won't feel my

best if I don't serve you a simple meal. So you can prepare yourself for these periods of frustration and encouragement, being mindful of your goal, help guide you through the times when you want to give up.

For me, my goals to continue to open-source projects which I want to talk to you about later and to help get more women into Python and Django, the two projects I use, I feel you have given me so much and I want to give more back, getting more women into computing is precious to me. I had to fight for it myself and I would like to make it easier for women like me.

When I start to feel down and out, I just remember these goals that is kind of like setting a big plate of food in front of somebody I love. Like these lovely people at a dinner party I have.

There you go, examples of my cooking and coding are not that different. If you can cook a simple meal for yourself, you can start with Django today. Sourcing materials from the library, don't take it personally if one doesn't work, you are not the issue, you just need another cook book, there are plenty out there.

Let us know when you are struggling, don't be afraid to reach out for help and to bring your full self to the enterprise.

Third and finally, having a goal and purpose will get you through the tough times.

So that was the recap. This is me, my cat, named after the programming language, I have another cat called pearl.

If you want more pictures of my cats you can follow me on instagram and I am also on twitter, follow me on there. Sometimes I tweet about coding and sometimes I don't. If you are interested in coding with knives, I haven't done much in the last two weeks, it is cool., lots of people referred to and said they have found it really helpful. This is the first of two talks, this was the open talk, geared towards newcomers. Giving another one on Wednesday that will maybe be a bit more salient for experienced developers.

Yes, be in touch and just thanks so much. It is a pleasure to be here. (APPLAUSE).

DANIELE PROCIDA: Thank you very much. So we have got a few minutes for some questions. So, if anybody from the floor has a question for Adrienne?

Funnily enough, Russell.

RUSSELL KEITH-MAGEE: Thanks for talk. You mentioned at the beginning you hit a wall with the tutorial. I was wondering if you could, share which tutorial you hit the wall with, and what the nature of the wall was, what got you over that?

ADRIENNE LOWE: Sure. Best possible question. No offence to other potential questions, one that makes me nervous, I don't want to upset anyone. The tutorial was Python the hard way. I don't know Zed, but I am sure he is a lovely person, but I struggled with his tutorial.

Where did I start to struggle? I think part of, I think part of what caused my struggle really had to do with imposter syndrome. Had been invited, received sponsorship to give my first talk at a conference, in the southern US, and I really worked myself up for that. I felt like I had to complete learn Python the hard way in a certain amount of time. That schedule was ultra-condensed and it didn't work out for me. I think I got to some of the later chapters in the 20's and 30's, he would say like, oh you need to spend a week on this or two weeks on this. Then he would have others, oh you can do this in a day and that actually took me much longer, then the things he said would take much longer, was more condensed for me. So some editorialising in his tutorial. It is hard to teach. Just like we all had our favourite teachers in school, like some you will connect with more than others, he was not my favourite teacher, he has done a lot for all of us. But his tutorial didn't work for me. So the thing that got me over the hump was reaching out. I had been documenting my process of. My friend said, you are flying, you need to slow down. She reached out to me and said that. If you see somebody doing that: Dang girl! Feel free to say Dang girl! It must be what they need to hear? Anything else?

DANIELE PROCIDA: I have a question.

ADRIENNE LOWE: Thank you.

DANIELE PROCIDA: What kind of difference are there in the cultures of the programming communities and the cooking communities that you have encountered?

ADRIENNE LOWE: That is interesting, I would say that among us programmers we tend to be free spirited and we have lots of things we want to work really hard and I think we want to make a difference, I think you also see that in the kitchen, people will consider that a plate, to be creative and work very hard. I will say, I haven't experienced this much in this community, the kitchen is highly regimented. We have the chef who is at the top and then we have the Sous chefs and Salad chefs and food runners you get a sense for that kind of, the way it is very regimented. Very much stay in your place, don't step out of your place and be respectful. While I feel we have respect in this community, I don't think we have the sense of you are this and you can't do that. That is freeing especially coming in from a marginalised group. Does that satisfy?

DANIELE PROCIDA: One more question.

FROM THE FLOOR: Anymore projects that bring your coding and your cooking together?

ADRIENNE LOWE: Still learning, I am hoping I have a couple of little things maybe I could chat with you privately about. I am still new. That is why it is an honour to be here in this community.

DANIELE PROCIDA: Thank you so much. Thank you (APPLAUSE).

One thing I have not mentioned, if you haven't already seen in your conference bag one of the items is a signed numbered, limited edition print by a local artist of the animal wall next to Cardiff castle so we approached Cardiff print workshop, asked them to produce their interpretations of the sculptures on this wall a very famous Cardiff landmark so look in your bag look after the print because it's a lovely piece of work and the artist will be at the conference on Wednesday.

I am very pleased to introduce Árni St Sigurðsson from Iceland who is talking about data driven democracy and his work round the Icelandic general elections. Thank you. {Applause}.

1.2.13 Žan Anderle: Mistakes and lessons in developing user experience

Our next speaker is Žan.

I am pleased to welcome Žan Anderle who is going the be talking about mistakes and lessons learned, that is a common thing in our field. So. (APPLAUSE).

ŽAN ANDERLE: Thank you for that. So, hello everyone. Before I start I would like to get a feel of who is in the room. So, how many of you here are students?

Okay. A couple. How many of you do back end?

Okay most of you. Front end?

... okay.

Also where you work, does user experience, is user experience something that is taking care of versus something that just happens on the way? So who works in a company or a place where it is taken care of?

Okay some of you. About half.

Okay thank you for that.

This talk is mainly about user experience, it is not specific to Django, where I work we use Django, will be happy to talk about that later but it is not going the be a part of this talk.

My name is Žan, I come from Slovenia, I studied maths science and Bachelor Degree in mechanical engineering, I currently work at datafy.it. I am in charge of user experience.

I don't want to make assumption, let's get that out of the way. One possible definition, it is basically how a person feels when they are interacting with the system. In this talk I will be using the term broadly, like sometimes it is better to use usability or user friendliness but we are not going the dwell on that.

So, say you are a developer who has little to no knowledge about user experience or you work in a team where, a team that doesn't have money or time to devote to development of user experience. What do you do about it?

That is basically what I am here to talk about. It is what happened to us at datafy.it and it is a start-up. We managed to do something about it. Let's have a look at the mistakes made and the lessons learned along the way.

But first let's get some context.

So datafy.it is basically is search engine for business contacts.

You get the country, what you get in return is business contacts. We are four developers and two sales people and because we are a start-up and a small team, this is what begs the talk rather than as we face this, this issues of not being able to devote a lot of resources to user experience.

So before we got the version of the app you saw on the slide before, we had bunch of versions before that. We started with an app in command line and the first public interfacing app was this piece over here. It was very complicated. I had tonnes of features that shouldn't be there, yes, it was really complicated. Complicated to maintain, difficult to explain how it works. So needless to say, the user experience from that first version to today changed drastically.

Basically, at first it was a lot of users were using to ... because it was difficult. For every new users the sales had to go through the process of how it works and explain everything which doesn't really scale because how are you going the sell to a lot of people if you have to explain it to every single one personally?

We got a lot of calls customer call related., we spent a lot of time to explain how everything is working.

Now fast forward to today, the one thing that people point out, time and again is the use of our app. Basically, do no more teaching of how to use it, it is self-explanatory, yes. Any calls or e-mails are just bug related and not user experience, that is good I guess.

Yes. There is still a lot of, a lot of stuff to improve but, we have learned a lot on this way and we have messed up a lot of things and through messing up we have learned a lot of things and this is what I will talk about.

I mean, this talk comes from all the mistakes. A quick think about user experience is, I notice it is, it gets ignored. As developers we seem to forget we are developing for the users. I don't know if that is us being lazy or ignorant but, it is not good because it matters. Since even if your app has killer features, if it is unusable painful to use, people don't understand it, it doesn't matter.

Why is it that we tend to forget about it? I think it comes down three reasons, lack of resources, lack of knowledge and ignorance.

Now through the mistakes and lessons I will talk about the first two, as someone coming from a small team, but just quickly about ignorance. I feel that a lot of times developers see their users as like that. [on screen].

Yes.

I think it is important to realise that when a user struggles with your app. It is your app's fault not user. If you blame the user all the advice I will talk about now, they don't make any sense because, there is an issue elsewhere.

Also I think it is really important to realise that developing pretty good user experience is quite easy. Now, getting to a phenomenal experience that is hard, that is not something we will talk about today.

So finally, this is the crux of the talk. Let's go through the seven mistakes we have made and seven lessons we have learned.

Now some of these may seem obvious and they are after you have learned them but, until you have, they are not obvious.

So as I mentioned before, what happened to us was that as a start-up we have to focus on getting the product out there. We have to focus on so many things and we don't have the time or money to have a single person work solely on user experience. Because of that, what usually happens is, teams don't assign anyone to user experience at all. Because someone can't be there 100% of the time. The thing is, if this is the case, you can't expect results.

For us it was kind of by chance, I can't really retrace how it happened just one day I took over user experience and you should take that with a grain of salt because I still did, I still did and I still do front and back and full stack. I just became the person to talk to about user experience.

Ideally someone would be assigned to user experience, focus all their time in reality there is no way usually and what we have learned is that you get great results regardless. Even if I still worked on other stuff, this proved to be very helpful.

So, in our office when we discussed the features or stuff we want to work on, we have sales people, developers, CEO, there is no user involved in that, in that debate. So when they talk, they talk about okay how we going to solve this from the technical side? What it means about the big picture. All that stuff yes, but no user. That is not really good because users should always be involved in that, those debates. Someone should represent them and someone should be there to defend their point of view. So whatever you are considering a new, a new the feature, new feature, does this make sense for the user, is this user friendly?

This came as a second nature once we assigned someone to be in charge.

As developers I feel we are often content in our own little bubbles, and we don't talk to customers because it isn't our job. Like the boss has moved, we don't talk to customers. When it comes to user experience you need to have input from the actual users. If you don't, you will just guessing and speculating what, what experience is for them.

This for us, this was, I remember one day, I heard my boss talking on a phone with a customer. I heard some complaints. Just by hearing that, I immediately got, I immediately realised, that is not okay, we should change that. It was 5 minutes work, if I didn't hear the input I wouldn't know that. He didn't think of passing that on to the developer's team. So, I think it is really important to, to include developers to customer support. So a really good advice I have is that all, any feedback you get from customers which is usually forwarded to sales or other, or what is the word? Other parts of the office, developers should be included in that too. What we do is, whenever we get feedback, they toward it to me, so, they forward to me, so I can see if the feedback they gave and what they are struggling with is part of a bigger problem.

This was one of the most beneficial things we did. Through that, realised, error messages were worded in a way that confused the users. We had for example, when you wanted to export the search results you would get a comma separated value format. Which made perfect sense to us as developers like you can import it to anywhere., to every customer that was painful. Didn't know how to convert it to Microsoft excel.

Also we realised for every single user, they have to like as I said, at first we had to explain how to, how to get started with the app. I simply created an intro to our app, which took less than a day, now everyone just gets it which is awesome.

Now, complicating is always an issue because especially for the developers I think we always have, we are working on something and we always have ten different ideas what else we could, ten new features, ten more ways how to improve something.

That is not always good because then you end up with, with so many features that it is difficult for users. It is difficult for you as a developer to maintain everything, basically nobody has a good time.

For example, we had a really complicated export which contained all of the data we could get. Which sounds like a good idea right? Just give them everything but, in the end it just confused the users.

Also we would offer pausing and stopping of the searching and scheduling and all kinds of stuff which you don't really need for a search engine.

So, simplify.

You have to be very critical of what features you are going the add and keep. Basically, in the office we have a rule that we are not going, we are always going the assume that no one is going the need a certain feature and move forward only if we have evidence that it is something that is going the, some kind of use to someone.

Now this one is tricky. Use of technical language. We don't even, we are not even aware that we are doing it but let's, I will just give you an example. So, we are doing a search engine and at first when the button or like start search we call it "start query," when we mentioned a search, we said query. Of course it is a query, why would that be complicated? Almost no one understood what we meant by a query. So simply by changing that to search, it made things a lot easier for everyone.

So yes, try to use plain English, it is so hard, since we are involved in that language, we are deaf to it, we don't realise the word might not be natural to someone else.

Yes. So, not listening to users. Listening to users means many different things and it means listening to them when they give feedback but also means testing new features with actual users. Sitting down them next to a computer, seeing how they interact with what you design. In ideal world you would do user testing but this would be very extensive you would have to get different people in the office and in reality you can't always do that, but we have learned that if you bring someone from the next office or someone from just the different person doesn't have to be an actual person and get them to use your app and observe it, you get so much information from it, it is incredible. I am surprised every time.

So always test new features with actual users. With actual people.

Finally, sometimes it is just failing to adopt basic principles. So, in user experience there are like ten basic principles if you follow just two and follow them consistently, you can get better user experience of the app than most other apps.

It is kind of surprising how easy it is to ignore this or forget about this. But yes, consistency and feedback meaning consistency, do your buttons in your web app always represent the same thing? Is a link that is pointed out wards, is it always styled the same way? Whatever action a user might take, do they get a feedback for it?

So on.

For example, with us, it was the first version was when they started the search it took like a minute to get started and in that minute nothing happened so usually just I mean obviously like, if I was there, I would do the same, click until something happens right?

It is really frustrating but, just a simple tweak was that, so yes. This was what happened after you read a search. Nothing. Even like, even after a, it went after last ten searches it was easy to miss.

Okay. Now you get a notification and it is separated from last searches so it is clear you started a search. It is a simple thing but so easy to miss.

So basically starting taking care of user experience because it is not that hard. Regardless of your knowledge, it is simple things you just have to be consistent and critical of yourself.

As I said. That is it. Thank you.

(APPLAUSE).

DANIELE PROCIDA: Thank you very much. We have got time for a couple of pretty short questions and short answers I guess.

FROM THE FLOOR: Hi, you mentioned that you assumed that you don't need new features or your users doesn't need new features unless you got evidence. What evidence do you need?

ŽAN ANDERLE: It can be many things for example, getting a person to request it. So getting a person to request it and say they are willing to pay for it. So sometimes you often like we will create a landing page, selling something, and we haven't even developed yet, but to see if there is any interest, if there is, we will develop it and otherwise probably not.

NEW SPEAKER: My question is what about why first and did technology you use have or doesn't have to do {inaudible}.

ŽAN ANDERLE: Sorry can you repeat that?

NEW SPEAKER: My question was about the mobile first applicability to such experience and if the general technology you use helps or make it more difficult to implement mobile first.

ŽAN ANDERLE: Yes from mobile first design we use boot strap from everything we doth if you use boot strap properly it kind of forces you to do mobile first. That's been working great for us.

NEW SPEAKER: One more up there.

NEW SPEAKER: How do you measure good user experience like for instance if you are a data driven team and implemented something and wanted to measure the effect you have any advice on that?

ŽAN ANDERLE: I think that is one of the controversial topics when it comes to user experience because it is very hard to measure. For us it was very obvious by the amount of user support related questions we got. So before even though we had a lot less customers we were dealing with customer support, support all the time, things they didn't understand, things we had to explain, that kind of stuff, and now we have a lot more customers and that just simply does not come up and that can be one indicator that user experience is better but other than that it's, yes, through user testing so when you sit down with a user to see how they use your app you get to see where they struggle and that can also be one way to tell if it's good or not.

1.2.14 Alasdair Nicol: Ponies and moustaches, or Templates in Django

DANIELE PROCIDA: Thank you very much. {Applause} next stop Alasdair Nicol on ponies and moustaches which are in fact technical terms in Django. {Applause}.

ALASDAIR NICOL: Good afternoon and I am going to talk about ponies and moustaches the more silver templates. I'm Alasdair Nicol a developer at Memset hosting concern south-east England. Started using Django in 2009. Like Adrienne who had the slot this afternoon the first time I used Django tutorial it didn't work at all and I came back a bit later and it worked the second time so it's good advice to try again if it doesn't work first.

In 2009 Django 1.0 came out, we've seen a few releases since then and in April 1, Django 1.8 came out. It's a long-term support release so if you upgrade to it you get security releases for quite sometime and some of the highlights API for the model meta class, there are lots of goodies if you use the postgres database and the thing I'm interested in talking about support for multiple template engines.

So, for any visitors who come on to the open day and don't know what a template engine is I'll start simply. A template is just a text file with a special - and the template engine can take advantage of special syntax in the template so in this case we've got "hello my name is" then 2 variables name and subject. Then if we have a context so the variables, name and subject, then a template engine can render a template and merge the context into the template. So even if you haven't used Django before you might have used mail merge in office or you might have seen an email where you got some these current in braces which means someone has messed up somewhere.

Before Django 1.8 if you wanted to - your choice of template engines there was a single choice of template angles Django template language. This is perhaps a bit surprising there was only one choice - not surprising there was only one choice when you get other parts of the Django stack there has been lots of choices for some time. Mark this morning was saying that with Django you have a choice of operating systems, choice of servers you employ Django with then what I'm looking at is within Django there is different layers and you have been able to have different choices before so when you come to Django and are choosing database you had lots of choices in database. Postgres SQL very popular with distributors in Django, my SQL which is very ubiquitous what we use, SQ light which is built into python which means you can get up and running very quickly, oracle and there is third party databases you can use as well.

Similarly sessions the cookies or the other way you store data about the user accessing your web-site, with Django you can store sessions in the database, use signed cookies or store on files though probably not a great idea to do that.

Finally one more example part of the Django stack swappable components caching where you take a page or part of your page and save it in memory so to speed up your web-site. There is mem cached support built into Django and third party support for Reddis or you can do caching in other ways as well.

Templates in Django - before Django 1.8 there was only one template language you could use. There were occasionally proposals Django should replace the Django template language with something like jinja. I wouldn't have liked that because we have hundreds of templates that work and if I had to change them to something else it wouldn't be fun.

It was a crowd funded campaign launched by Americ Augustin, it's one of a few Django features which has used Crowd funding and indigo {inaudible} seems to be working well. Any way the 3 planks of the campaign or the 3 goals were to keep support for the existing Django template language add support for jinja 2 and API so other people could swap in any template language they wanted.

So for the rest of this talk I'm going to have an example web page which I'll show you in a second and go through the 3 bullet points to see how we implement with Django template language with jinja 2 and another engine.

Sorry this is so basic, my skills are round the back end rather than front end development but since Djangocon here is a table how we moved from 2009 in Prague to 2015 here today in Cardiff.

Django is model template view or also other frameworks use module view controller so before we get to the template we have a model which describes the database layout in Python code. Got a very simple conference model. Going to store the location and country as Charfields {inaudible} country would be better. Years into the field. The view is where you describe what data you want to display on the page resident Python. A very simple view which grabs conferences from the database then renders the template. And finally here is the template to - or section of that template in the Django template language. So, we've got conference year, conference location, conference country, the double angle braces or moustaches, including variables and then the other couple of things this template has is we've got a forward, that's using a template tag where you've got the angle of brace and percentage and we've also - we're using the cycle tag so that we can display {inaudible} even as odd not shown very well on the slide but you can imagine there is {inaudible} extracts going through that column.

So, moving on to jinja 2 which in Django 1.8 there is support for out of the box, jinja 2 syntax is very similar to Django's, very fast. Used by flask, a micro framework in Python and I have been exploring Ansible recently and it uses jinja 2 for its template as well. Enabling jinja 2 in 1.8 is easy. There is a new templates list and by default it will come up with one entry, the Django template engine and you just need to add the second dictionary to enable jinja 2. Then when you write your jinja templates you put them in a directory with jinja 2 and when you use the short cuts Django will take place of everything else. So here is our template in jinja 2 which works very, very similar to our original template in Django template language. I think the only difference in this template is that instead of using a cycle template tag we've moved our cycle instead and jinja allows you to cull functions within the language which Django doesn't. I think that's a philosophical language, {inaudible} thinks you should keep language out of templates and jinja allows more logic into templates.

Now on to the third part - I had a go at seeing what the API was - see what the API was like to add support for another templating language and I chose moustache so I could use the title I talked about. The other reason I thought moustache was interesting was that there are so many languages that have support for moustache and we've had couple of talks today where people talked about rest APIs and perhaps the need for doing templating on the server side isn't as much as there used to be but maybe there is a future where you're doing templating on the server side and on the browser in which case something like moustache where support for Java script and...(inaudible)...is potentially interesting.

Yes in Python there is a module pystache which {inaudible} I used.

Here is my template engine. You need a template object which knows how to render itself and most of that was taken care of using pystache then you subclass {inaudible} base engine. You set what you want to keep {inaudible} moustache directory and then we implement 2 methods, one which will render string and the other render template style from directory and render it moustache template engine.

Here is a template written in moustache. Again using double angular braces for variables. Looking through conferences looks a bit different. There is not a 4 conference and conferences syntax like we had before. And the one thing you might notice is that I couldn't work out how to - I couldn't find an equivalent for the cycle tag in Django template language although you couldn't see it on the slide any way so never matter and you could do it with CSS selecting even {inaudible} any way.

So, we've got the 3 layers, the model, the template and the view, and in order to get the view to work with moustache had to change it slightly and add dictionaries instead of conference who ran objects to whatever was rendering the template so I cheated slightly but I think that I have shown them that you can for the most part keep your models and views as they are and swap in whatever template language you want.

So, I am going to declare success. My pony has grown a moustache. If you want any further reading, I am not a {inaudible} don't worry. The weekly updates from Amaric about decisions he made and process are interesting it's Django open source and it's great you can see the collaboration decisions and?? Getting contributed. There is a new process called Django DEPs similar to Python DEPs and I think this multiple template engine is the only DEP so far that's been accepted but it's interesting again to see a new way of how decisions about the future of Django are

made in public and by the community. The Django design philosophy page on the web-site says a little bit about why Django thinks there should be less - about some of the decisions made about Django template language, I touched on that briefly when I said there is not as much logic in the Django templates as the jinja templates and there is 2 scoops of Django which is a really great book and the new edition has a chapter on jinja and some good hits if you were going to be seriously considering using it in production rather than just a fine exercise.

So, yes that's my slot. Thank you very much. {Applause}.

DANIELE PROCIDA: Thank you very much. Not least because your helpless claw back lost time. A coffee break waiting for us. Thank you very much. I already saw a couple of hands moving in the audience so who would like to go first with a question? OK I'll start. So, why might one not want to go down this route of exploring additional templates? Why not just stick with the completely built in out the box Django template language that everyone else uses that you know has been widely tested and will do pretty much every tutorial documentation at work? {Laughter}.

ALASDAIR NICOL: I think that's a very good point and I personally work anyway - I don't think we're going to be switching any time soon. If you had a performance reason you might want to switch to jinja although many people say there is other areas like caching or database where you get more performance rather than templates any way so I like the Django template language, I'm not advocating we switch to something else but I think it's really good we formalise the API and given people the ability to swap it out if they want to and I think there is some of the even by making the API more generic it's improved the way the Django template language itself interfaces to me to some extent -

DANIELE PROCIDA: It makes me nervous when I feel there is an implication I might be expected to learn something new.

ALASDAIR NICOL: I agree like a couple of talks today suggested that the future is REST APIs but actually it is very nice that with the Django template language at the moment you can get your tutorial or your log in the Django girls tutorial up and running and you don't need to learn yet another front end single page app.

NEW SPEAKER: By reaching engine {inaudible} template packs?

ALASDAIR NICOL: Yes I think that is probably the biggest - so the templates act yes I think you'll need to re-write right the templates act. The filters - I think say if you use jinja, I think you can register filters - a filter is just a python {inaudible} takes a value and one other and returns another value. I think in jinja 2 you can put filters but tabs if you switch to jinja that involves quite a bit of re-write.

NEW SPEAKER: Is there currently the ability to choose a template to run in {inaudible} time so if you wanted to render may be an Ajax view slightly differently with a different engine {inaudible}.

ALASDAIR NICOL: Yes so the example that I used was you do - templates by configuration so you stick your jinja templates in jinja directorate and Django will look for them but when you use rendering in Django 1.8 it's got engine key word so you can have - I have got example template that can do most of this stuff and the way I've shown the view 3 different times is with a get parameter which the engine equals Django or engine equals moustache gets passed through but yes at one time you can choose engine.

DANIELE PROCIDA: Very quick one.

NEW SPEAKER: If I understand correctly the reason the logic is taken away out of Django template languages is so you get better design principles. If you follow jinja is that taken down a darker path? {Laughter}.

ALASDAIR NICOL: Ah ... so I think perhaps you'd be in danger of going down that dark path because one thing jinja has that I would like Django template language which is to be able to look up and attribute by variable which you can't do, you can use it basically it looks like dictionary look up syntax in jinja which you can't do in Django I think that would be nice in Django, I don't think the world would implode if we did that. I think the Django philosophy of trying to keep logic out of templates I side with it to an extent.

DANIELE PROCIDA: Smashing. Thank you very much. {Applause} we have a coffee break now, so go and have some refreshments. We'll come back, try and be here for in 25 minutes time so at 10 past.

(Break)

1.2.15 Christopher Hunt: Arduino sensors, mobile apps and virtual reality

Hello and welcome to the final session of talks today, we are going the kick off with Christopher Hunt, the lead developer, research assistant and ... at Plymouth university, he will be talking about Arduino sensors.

CHRISTOPHER HUNT: Are we all caffeinated ready for the last session, tried to Welsh cakes? Hopefully you have.

So welcome, today I am talking about we are going to talk quickly and speed through, try and scare these people doing a lovely job in the corner here.

So, come in people, you have missed the start.

So the Django and the data system. Objectives for today, understand the concept and come points of the data system, a metaphor for the things I have been using, the different processes and systems.

Discover we, i-DAT is using Django, try and provide two live demo, only one now, because some of these didn't survive the train journey up here.

I am from institute of digital art and technology, we are an open research lab for playful experimentation with creative technology. Very fancy. I didn't come up with that, my boss did.

We exist at Plymouth and support the digital art and technology undergrad course, research, environmental things, social stuff, domes and different environments, biological and lots of other key things.

Key line from our description of our strands tat bottom. From data to code to experience to behaviour.

This is the process and this is what the systems and what we can do enables. So as we go through, keep that in mind and keep thinking, what kind of things, what stuff you can enable, tools and technologies we can play with as we go through this stuff.

So, the data system. This is my model with lovely ASCII art I am proud of this. Start at the top. Circular system because it goes around in circles.

At the top we have sense and collect. This is sensors, these are these devices, things that you know, like sensors, a few other things, they transmit to somewhere we can store and process that data. This is where Django comes in and that is where I will talk about that. Then that enables interpretation and understanding of that data. This is visualisation, this is oculus rifts, and cool stuff at the end.

Hopefully through all of it. So first step. Sense and collect. Which transmits to somewhere we store and process.

In here, this set of transactional props comes from PhD projects this is done with myself and one of my colleagues, John and in here, we have these tiny little formally sparkled now particle. So I have got the re-wipe that bit of code in my head. Renaming things it is annoying.

These are little Arduino light devices, programme them using the same languages, wired directly into the device.

So, what is going to show that but because, in the interest of time and with a few network issues, what happens with this project two people sit down. Then, these devices all have different sensors the funnel has an fsr, you squeeze and then it, certain cups vibrate and little nozzle tips light up and do things. The idea you are using the sensors and exploring the relationship and the system beneath. So behind this, there is this Django page which is receiving all the data over something called message cue ... transport.

All these devices sending up to here, we can see this going on, this goes to another back end, we can explore and see the data then different stuff happens there., I may need to ask whether somebody has a soldering iron.

Pretty standard code, except extra bits for connecting to a broker, connecting to NQTP. It is a publishing subscribe protocol, nice and easy to connect with., here we are connecting to our NQTP broker, then publishing our information our messages, so, the green glass, what happens here in its loop, checks if it is connected. Every time I tap that, that would then send that message, that fault it received over NQTP then make it vibrate, but we will show you that later.

So that is sensor, those are Arduino's and different devices but, at the essence of that, that is just data, data can be any kind of thing, we have done a lot of project and research with arts organisations around different, about how you

gain and manage feedback. Traditionally someone would pull, go around an art gallery, someone pull you aside with a clipboard to answer lots of boring questions. What comes to your mind when you think of science?

That is, then you have got this open ended, this event did not match my expectations dry and not interesting boring things. You start to see lots of ways to get around this. This is a happy or not sensor, the question here: Please rate our check outs.

You may have seen these at airports as you go through the security gates. You go through, then slam on the button say, hey I enjoyed it or this went badly.

I can imagine sometime next week you might see something like this, dotted around Cardiff! just in case you need to voted on Zain, say you are a one direction manager, find out who next to put on the bottle of fragrance, we will get back to that later.

This is developed into our quali project, this is where the mobile apps come in. This is the same kind of thing, an app for the Cheltenham Festivals. As you go around the festival, as you explore and find out stuff about it you get to rate and leave feedback. Say what mood you are in or did you enjoy the event. In the micro-interactions, instead of having a big long essay of things to fill out. Tiny bits of data, as we collect more and more of that, we build up a better picture of what is going on.

So, devices, sensors, mobile apps all cool. They all transmit back to somewhere to store and process the information then enables interpretation and understanding.

So, we could use fi base or pass or all the platforms of the service thing, we are at Djangocon, I would encourage you, why not build your own? A good question. You could save time with those, but you want to learn and, boar reed this quote from my boss's talk mike. I will give it my best Tom Hardy impersonation.

"take control of your city. This is the instrument of your liberation!"

I really should have been an actor not a developer clearly.

But that is it. It is for us, it is being able to control a platform rather than rely on someone else or worry about things degrading, things being unable and falling apart so, patch ... cosm and xively, lots of stuff out in the field broke. Great. Amazing that.

So you learn and that ability to control and build stuff really is valuable. This is an old screen shot from digital ocean, but the cost of computation, the cost of running the server is so small you know, you can easily do quite a lot of big project work on just a tiny VPS, easy peasy, – got 5 minutes left.

Dive through the last few bits, why Django? Why use it? Because it is awesome. Why else? Obviously you need a few more reasons than that.

So top of my list, models and migrations, that makes it easier to prototype the data. Here is my model our happy or not button, ignore the code, only for demos, I have forgotten to remove a certain someone from the member choices, that is depressing, the value of Django, that makes it easy for us to prototype, we can iterate, get stuff out. Get people playing with stuff.

REST API libraries, allows us to enable different things so we can talk to tasty py and the rest, there is the data from earlier, of course you have got Python. The only way to describe the power of that, well with that. You have got all the sorts of things, as you have been listening today, you have got numpy, scipi all sorts, takes the work away from you, you can get on with doing amazing cool things. Of course, the kit most important thing in Django, is the community, is you lot. Everyone who is really friendly all the core developers, or core committers are so friendly and helpful. That makes a big difference when we are teaching and working with people, then aren't afraid to get involved, mix in, ask questions, that is brilliant.

So lastly, so we have stored it somewhere, built our Django back end, now we need to interpret and understand the data. Because we are working with arts organisations we can do something like this, just the templating language, so build basic views on demographics, feedback, we can look at hot spots and send Norwegian puppeteers where people are building the apps. Create systems to create questions, get interested and involved in the data we have the immersive vision theatre which is our planetarium, here we can fly you to the edge of space or through the bosses colon, which is the weirdest experience I don't want to experience again!

Or take you into data scapes. So this is something we have been developing for one of our conferences and lastly, I can dive straight into another demo.

Fingers crossed. A ha.

Fantastic.

So, interesting effects, so in here, I can now can't see you obviously. You are all just floating bits of data. But, if I look up there is some pictures been down loaded through one of our back ends, here some blobs of data, bits of orange there. You can come along and experience that later.

But that is all about engaging and getting people involved in the interpretation and the exploring what is available in that data.

Okay. How much longer have I got?

One minute.

Cool. Fantastic.

That means I can do the last few examples, from i-DAT's work, we have a building, we have sensors in a building including a vision system. Can you see this thing moving? Should now start going backwards because the video is looping backwards. So this is our Slofbot, using the whole model. So the sensor is at the top of the atrium, that is looking for detecting people going across the room, as you go along to your lecture, there is suddenly a wall in the way. The idea of that is to kind of get you thinking about the space because you don't think about this corridor, then something is there, I must realise what is going on.

Or, this is very old video now, but or you, or you will look at how people use the space and the building. Then you decide as a long term intervention you put a random duff button in the lift, to get people exploring through the data.

That is the data system, that is all the elements, I will open up to questions before I get pulled off the stage, thank you very much. (APPLAUSE).

NEW SPEAKER: Can I take any, Chris take any questions?

So you mentioned briefly at the start of your talk, how is your work explaining – do you want to explain?

CHRISTOPHER HUNT: So, the internet thing was, when you say interneter things, you think a smart house, smart car, smart things going on. I don't like using that term, so that is why I use the models with the students and the people I work with, to break down the system going on. It isn't just remote controlling something, but it is interesting things going on with that data.

FROM THE FLOOR: So obviously Python and Django on the server side, but the nasty language in the ... side of things, is there a way of fixing that?

CHRISTOPHER HUNT: That is a good question. There has been some attempts at porting, having Python then generate that C code for you? But, unfortunately, you can't just run Python on these sadly.

RUSSELL KEITH-MAGEE: Is it a hard way limitation?

CHRISTOPHER HUNT: It is a so, on here, what that C code is done, compiled down and then run on a devices. What these devices are, they are essentially dumb. All these are doing is sending their data up to the Django server and then waiting for messages to come back. So we can add all the intelligence, all the interactivity server side and manipulate that and change that without having to change the code on these devices.

FROM THE FLOOR: There are slightly more powerful devices that can run.

CHRISTOPHER HUNT: Yes, these are simple ones here, raspberry pies or eagle bones, it is all becomes enabled and available.

FROM THE FLOOR: How did you make Django talk in PDP?

CHRISTOPHER HUNT: Django is using the pay, ... library, then a written a management command which is supported by supervisor, then runs in the background, listening for messages coming in, validating them and then to the models. If you are interested I can show you the core stuff that is going on.

NEW SPEAKER: I think we have time for one more quick question? If not, then we can thank Christopher Hunt again.

(APPLAUSE).

NEW SPEAKER: While the next talk is getting set up, there has been a slight alteration to the time for tonight's meal, aiming to be at the restaurant for quarter past 7 now.

NEW SPEAKER: Could you repeat that?

NEW SPEAKER: Change in the time for the meal, aiming for quarter past 7 now.

1.2.16 Katharine Jarmul: Data wrangling with Python

NEW SPEAKER: So the second speaker of today's section is Katharine Jarmul and works of customer data analysis using Python and today is talking about data wrangling impact.

KATHARINE JARMUL: Hello everybody, nice to be here, I have prepared a talk that is probably after seeing the last hour or so talks a little bit novice for this audience, however hopefully you can use something from it and if not then feel free to use my slides and tell your aunt or cousin or mother how to do data analysis with Python and yes if you want to talk more advanced we can hopefully chat later.

I am Katharine Jarmul, I am at Kjam at most type of tech things; I was originally from Los Angeles, I live in Berlin. I hope everyone is familiar with Pyladies? Yeah? So the original chapter was in Los Angeles and it's really exciting to see now how much it's grown, it's really amazing and makes me feel very warm about the Python community. I've been coding Python since 2008, I started with Django at the Washington post when there was still ugly Adrian hats in the {inaudible} it's a good time yeah we can talk about some of the caching and I am self and mentor taught so I really hope for those of you who are new to Python that you can find and make some connections here today and throughout this next week or that you've already made those because I definitely wouldn't be where I am today if it wasn't for the mentors that helped me.

So, a little intro to what exactly I mean by data wrangling.

So, it's basically the ability to analyse as something with data. Everybody here probably has done way more advanced data analysis than this but the really good news is that you can run any type of reports using Python and data analysis so some of what I do currently is running marketing reports and user analysis, site visitor analysis but you can also do sports analysis with statistics and {inaudible} quite a large open data {inaudible} a great tool to use.

Why use Python it's a scripting language with some real power. There is advanced scientific stack of course which I hope some of us are familiar with. This is a really friendly community. It's really easy to ask questions and get help. And of course it is named after Monty Python and here in the UK that makes it obviously superior to all other languages.

So why do I even care about Python or data analysis? So the way I approach it especially when I'm talking with people who don't know at all about code is do you ever have wrote boring awful tasks and would you rather never do them again? So learn Python and that's what I do. Sadly cannot fold your laundry yet but talk about that sloth balling where do you go, yeah, we'll see if we can get that working. So again Python allows you to have statistical power without necessarily becoming a statistician. You can easily automate things and you can never ever use excel again and hopefully some of us recognise a nice Welsh dragon burning Microsoft excel!

So the first step of any data wrangling is getting hold of data that's interesting for you to use. You probably already have some. There is probably something you do regularly whether it's logging into a utility bills or whatever it might be - you can use that data, another great source is the very large scale open data movement, so data gov UK has quite a lot of good data sets and of course Python supports any number of formats, CD CSV, Executive, PDF, XML, Json, Google docs and I'm sure many of you are aware of supported natively. For those not supported natively there are

lots of really useful tools. I recently had the pleasure of using G spread which is a Google spread sheet user which is not the Google spread sheet reader API and it's quite intuitive and easy to use, XLRD is my favourite if you have to output excel text for people in your company and PDF miner has been something I've been recently working with to mine PDF documents.

So data basing. For a quick start especially if let's say you're working with people on your team that don't know SQL yet I would recommend data set, it's developed by a guy named Puto. It's not his name but his handle so to speak and he's part of the news foundation and lives in Berlin and we're grabbing beers next week I'm helping work on some of the bugs so if you have any things on that send me stuff - there is also relational databases, {inaudible} et cetera and non-relational databases so you can use mongo, couch ...

So APIs application programming interfaces are a great place to get data and you can get Python direct with twitter, Instagram, Facebook and tons of other data sets. Let's have a look. Yesterday I had the pleasure of sitting in Cardiff and scrolling through twitter and I wanted to search for {inaudible} did I butcher that - who speaks Welsh? Caerdydd ... if my family were here they would be very not proud of me. And I just wanted to see what people were tweeting about, a gorgeous day yesterday in Cardiff so you can see that we have just some tweets that we were able to get about 10 lines of Python code.

Other APIs that you can use, Google analytics. Google adwords, these are things that are really useful if your customer data that you need to integrate say with a back end so say you need to pull all this analytic data in integrated with the back end and say how long are return customers spending on site etc. that can give you more information. Also, plenty of open government APIs, I've actually been interfacing quite recently with a lot of the open Africa data looking at some of the conflict mining stories there so there is plenty of data sets there, translation APIs again if you need to do anything like that, stock market APIs recipe APIs, million APIs to interface with.

So a little bit on web scraping, if there is not an API and you need to access the data you can build your own API so to speak with a web scraper so Python is uniquely situated in that it is a scripting language and therefore gives you really easy access to read something like HTML H or X document and then allows you to use the information really quickly with the analysis so I find it's uniquely situated. I don't think that's something necessarily that the community has yet so that's really nice reason to know Python.

So if you want to take a peek at a page I recommend LXML it's tremendously fast and has great syntax. If you actually want to click around and use things you can of course use selenium but something recently {inaudible} interact with ghost driver which is a little bit better I think in some ways and then if you really, really need the whole site you can use something like scraping, scraping is tremendously fast and useful and worked on with a great team of developers.

OK so I took some time to just scrape the Django talks for this programme to scrape the content for simple stuff and returns what's on the page {inaudible} - OK big deal everybody here can read a web-site or use a browser that Laos them to read a web-site so imagine if you couldn't so for example right now I'm using PowerPoint because I don't have internet connection so if you don't have internet connection, if you are travelling, if it can't be retranslated by Google translate or if you wanted to run data analysis on it this is a good reason to scrape data off the web. I did data analysis on the talks most common words are "the" and "and", I don't know what's up with these words and we can start to see Django "discusses, describes" and there is a lot of "nice, useful" depending on how interested you are in natural language processing there is a lot of useful APIs out there for stripping out things like these site words, occurrence words; average character length for titles is 35 characters and 7.4 per cent of talks mention Cardiff.

So, another thing that Python allows is allows support for "big data" - I don't really know many people who do require big data but if you do require it you can integrate it with hadoop and pandas numpy are some of my best {inaudible} and anyone not familiar with them it's great to get Wes Anderson's book and start working through some of the examples using panda's you can automate reporting again an allow you to generate reports on the fly or allow you to create normalised generated reports that run every week and allow you to hopefully move some of that front work off your task and then you can run statistical functions, generate graph charts find and remove if you need to outliers or find news outliers, normalise data and perform data clean up so quite a lot of clean up libraries I highly recommend if you haven't used it before taking a look at fuzzy wuzzy is one of my favourites for if you have to do some language processing and it's not always the cleanest.

OK and then of course visualization which is the point so you've ran all of these things together, the data that you need, you've done some statistical analysis and now you're moving on to visualization, Python has great ones, Bokeh

is one I have been playing with, matplotlib is standard and pygal has pretty cool SGE related ones and then the ability to easily share coded charts with iPython notebooks so a client I work with currently there is a lot of non-technical people at the company but they can easily run the code that generates the reports they need they can down load the excel document directly from the notebook and I recommend it for teams that need to interface with teams that are may be a little scared to code Python and I found that over time they become a little bit more accepting to play around with it if you just change the variable we can run it for a different data set and that's been kind of exciting.

This is bokeh. If you haven't played with it. Pretty cool visualization just from the grailer(?).

So if you want to know more I know this one lady who writes about Python so have a book coming out with another one of the folks that I worked with at the Washington post it's an O'Riley book and yes if you get it today there is a free Brains pint in it for you!

OK so you can ask me questions now or you can ask me questions later, so I will be here sadly until Wednesday when I have to get back because I have an intensive German course in Berlin but yeah feel free to reach out and thanks so much for listening.

{Applause}.

DANIELE PROCIDA: Thank you very much. So, do we have any questions from our audience?

NEW SPEAKER: What is the hardest data process you've probably faced?

KATHARINE JARMUL: 100 per cent data clean up it's the biggest pain I don't think anyone really likes it but I think that there is quite a lot of powerful tools down for it but I find it to be still the most manual of processes. If it's clean data and I can import it from a database or a clean source that's great I can immediately start using pandas or whatever I feel like using that day but yeah clean up particularly when there is no normalisation of data say like a matching non normalised strings things like that it is just kind of one of those - so may be one day we'll solve that problem, I don't know how but have a pint of Brains and talk about it.

NEW SPEAKER: Do you have a favourite toolkit you use?

KATHARINE JARMUL: Yes I mean the NLTP toolkit the standard one is the one I've most played around with but I think there probably will eventually be one that's maybe somewhere in-between that having to have that entire stack and having to learn so much about NLTP but allowing people to kind of use some of those tools within just a small library, I think fuzzy wuzzy is useful for using that talk analysis.

NEW SPEAKER: Do you know of any way to get data from say film files?

KATHARINE JARMUL: From film files?

NEW SPEAKER: Yes it's made of audio then many frames -

KATHARINE JARMUL: Does anybody have any ideas? I haven't worked with film before.

NEW SPEAKER: FF {inaudible}.

KATHARINE JARMUL: ?? ... Yeah?

NEW SPEAKER: ... Testing your analysis against a known data set? Making sure you run the same {inaudible} reproduces design output always.

KATHARINE JARMUL: That's pretty essential. I think one of the problems that you run into with this and some of why it can't always be tested is that you have to take into account say your handling a standard deviations or out liars right and do you have a normalised or non-normalised data set and that's one of the hardest things is figuring out OK identifying is your data set normally distributed or not and maybe taking different paths depending on that so I think like testing your data a little bit first and getting toe it is an essential first step before you decide OK this is the report I can use with it. If not you are going to find your reports become really skewed because of one particular outlier or a few outliers, I think that's essential and needs to be done more often and I think that determining different pathway depending on data distribution is another key part of that. Thanks so much. {Applause}.

DANIELE PROCIDA: Thank you.

Okay a little announcement, two little announcements whatever you have heard from our website or written down in your programme booklet or in a hand-out or anything else, I am telling you now the correct time for being at dinner tonight. Ignore anything else, aim to be at whichever venue whatever it is, if you have a ticket for the Vegetarian Studio or the Clink at 7:15. From here it takes about, you can amble to either of those destinations in about 20 minutes. The aim to start eating by 7:30, if we aim to be there by 7:15 that will be helpful. If there are sponsors who would like to get their stuff moved to City Hall ready for tomorrow, we will put it in the van and take it down to City Hall which is a short while.

ADRIENNE LOWE: If we purchase hello web app book, Tracey Osborne's book, deliver it to the registration desk.

FROM THE FLOOR: When does the ...

DANIELE PROCIDA: We will put a board up for you to sign tomorrow. There was another thing to say, it slipped my mind, I am sure it was really important.

Oh if anybody fancies has a spare pair of hands to help put any of the conference stuff into the back of the van when the talks are over, that will be very handy.

Ah yes, so you should already have your tickets, should have a printed out ticket, don't worry if you don't have the printed out ticket, but either have purchased the ticket or had the ticket from us in one form or another. If that is not the case, and you expected to be at one of the restaurants, see me up in my quality room office thing. You can still buy tickets for the VFS they are £15 for a good vegetarian meal., all the Django Girls ... you will have a chance to go the different restaurants.

So, yes?

1.2.17 Rivo Laks: Django and the real-time web

So, you have got 20 minutes including questions. So, Rivo Laks travelled from Estonia, on the real-time web with Django, thank you very much. (APPLAUSE).

RIVO LAKS: Thanks, I am Rivo, I come from an Estonian product development agency, we use lots of Django in our products and we try to change the world. Here I am here to talk about Django and the real-time web and how the two link together.

So Django is ten years old and it is certainly a stable and mature framework but when you think about it and the web has changed so much during the last ten years, in 2005 we didn't have small personal computers in our pockets, mobile web didn't exist, Gmail just came out. It looks like that, paradox 1.0 was also something new. So the question is, is Django still relevant with all the change requirements that today's web puts on it?

Or has the pony become the dinosaur?

No I don't think so. Because, Django is actually really modular and really flexible and you can use it to you can use the parts that help you and will show that you can just add a real-time functionality into your application really easily and keep your application up to date so that it hits today's needs.

So, a small disclaimer real-time web isn't obviously something that must be used everywhere. My Blog for example doesn't really need real-time updates but, on the other hand it also has some really good news cases where it makes the UI more flexible and fluid and makes the user experience better which is what matters in the end.

So, probably the easiest way that is you just need to push some updates from the server side into the client side when some data changes. This is commonly known as the Pub Sub. It is really easy to do in the sense that there are many external services that provide this functionality for you and it is always good use insisting services or libraries where possible you don't really want to reinvent the wheel, but you want to focus on your application and use whatever building blocks are there.

So some of the services include pusher and pusher is what I am going the show today. They all work pretty much the same way so it doesn't really matter that much which one you want to pick.

You just basically go the pusher website, sign up for a free account and first thing you do is you need to add some code to your server and it is really easy. First you create the pusher instance, give it your ID and credentials. Next you can already start sending messages with the trigger function. The trigger function has 3 important parameters. First we have the channel which is kind of like chat channels in the sense that when you send a message to the channel then all the clients which are subscribed to the channel and listening to it will receive this message. The second in this case show message is the message type itself or the name. So if we are building a chat we could have a show message we could also have join and leave in the names. Then we have the actual data that we want to send in this case, the message itself and so, with less than 20 lines of code we already have some way of proactively sending updates into the browser whenever something happens.

The next obviously we have to actually receive those messages on the client side in the browser. This is almost as easy, again we create the pusher instance then we say that we are interested in a certain channel or the message is sent to that channel and then, event handler which reacts to messages of a certain type in, in this case the show message.

Here, in simple demo we just show the model with the receive message.

I want to show a live demo here, there is one, I will give you the link and you can try it afterwards.

You probably also want to have the other direction you want to send the data from the client or the browser to the server and the easiest way to do that is actually bios requests, so you don't need anything fancy there, when the client or the user types something and wants to show a message you do the usual post request to the Django and then Django can push this message forward to all the clients listening.

So, as you can see, it is really easy to add some real-time functionality into your application, talk about the 20 lines of code would probably take about 5 minutes to integrate it and the external services are good again in the sense that you don't have to have your own infrastructure, you don't have to build the building blocks. You can instead use existing libraries to build your application and focus on what is important for you.

But, sometimes it is not really enough what the external services provide. For example, you might want to have faster messages or the full control over the messages that you send. This is where web sockets come in. So web sockets sees a protocol standardised in 2011 supported on every modern browser. What web sockets is give you the connection between the browser and the server and it is very low overhead, it is very fast. It also supports both text and binary data. In fact we used this binary web sockets where we used to it to work with street lights. So it is versatile.

Let's also look at how web sockets can be used and integrated within Django, using asyncio library, we will also be using web sockets package which gives you again the building blocks or primitives that you can use to just focus on sending and receiving the messages instead of implementing the web sockets protocol yourself.

Because the web sockets should run on a separate port from your main application, we are also going the need a custom server process for that.

I am skipping some of the code. But the important part is this, define a handler that gets called whenever a client connects. Then you can use the web socket to send and receive messages.

Again, very simple. You can expand on that to basically create the loop that reads messages from client and then processes them in some way like printing them out to the standard output.

On the client code web socket has pretty good standard API so you don't really need any third party library, you can use ... JRS, but also use simple rapper library, I am using something called sawkit.

When the connection is made, it sends a message to the server and waits for incoming messages and reacts to them. Again, showing the alert.

So I hope I have managed to show that Django and real-time web can relate really easily., easy to use external services like pusher. Also simple with websockets. It is not difficult but often some third party service will also get you off quite well.

There is also demo up at Djangocon.thorgate.eu. There is code. We also have a stand at the front, please come and see me if you are interested in more. I will tweet the location of the slides thank you. (APPLAUSE).

NEW SPEAKER: Does anyone have any questions?

FROM THE FLOOR: Not really a question, more a remark, there is a new ... called web bush and ... (INAUDIBLE) we have made an implementation that is called web push and there is an implementation, ... so if you want to ... you can use it. Basically, when the clients connect to the server, ... you can post ... web socket.

RIVO LAKS: Okay thank you, that is really interesting to know.

FROM THE FLOOR: What sort of things would you recommend if you wanted to have a mobile native type app.

RIVO LAKS: I think pusher has made ... it depends on your needs, for something pusher might be enough or you can use websockets.

FROM THE FLOOR: How would you handle anything other than a client ... you could (INAUDIBLE) something like a mobile app, so is it just for websites and browsers? Or can it lead to anything?

RIVO LAKS: Use it between any two, in the project we are using it to communicate between server and not browser, but many devices which controls street lamps. So, street lamps, so yes, it is quite versatile.

FROM THE FLOOR: Try out ... dragon, which is a library that is made for Django, using websockets.

RIVO LAKS: I have heard of it, but not tried it yet.

NEW SPEAKER: Okay, let's thank the speaker again.

(APPLAUSE).

1.2.18 Jamie Hannaford: Making our spaces more inclusive

So today for our final talk of the final session we have Jamie Hannaford who is a software engineer with rack space, and will be talking about making us more inclusive {applause}.

JAMIE HANNAFORD: So our industry is defined by its ability to solve technical problems. We produce software, tests, documentation, sometimes of immense complexity as a result. But, let's not fool ourselves the most technical problem in tech is not technical it's social. People are more difficult to work with than computers and to day I want to talk to you about diversity but also about inclusiveness.

So the first diversity I want to talk about is diversity in the workplace, the challenges we face and some of the techniques and this we can do to mitigate and overcome them. The second thing I want to talk about is open source communities and how we can make them more inclusive for everybody. Thirdly I want to talk about physical spaces, conferences and meet ups and how we can minimise exclusion there. So, I think the first question we really need to ask ourselves is what do I mean by diversity?

For me, diversity is not just about hiring more women. Gender is only one part of the picture. We should aim bigger by incorporating a wider spectrum of people that covers race, ethnicity, sexuality and religion, age, disability, class, educational background and bodily appearance. These are all truly important aspects of what it is to have a diverse team. And to answer the question of why is diversity important the simple answer is it makes your teams better. Fresh ideas and new perspectives are introduced. And we begin to think differently and make better decisions. And as a result, we produce more innovative products which drive business up for the overall company.

In fact there have actually been studies that prove this. For example, the Kelup School of Management a few years ago commissioned a study where they got 200 people and placed them into groups of 3 and they gave them a simple detective problem. They were given a set of interviews and they had to determine the likely murder suspect. Now each person was given the interviews and they basically had to come up with their own ideas but they were in groups of 3 from the same social background so the groups were fairly homogenous. After 5 minutes a fourth member joined the team. 50 per cent of the time it was from somebody from the same social background so 4 people who effectively thought the same. The other 50 per cent of the time it was a stranger an outsider which was effectively composing a diverse team. The results were startling they found because the diverse teams guessed the correct murder suspect with far greater accuracy than the homogenous group. And this wasn't just because of the influence of new ideas and fresh perspectives, what they found was it actually triggered more careful information processing, so there is a real, real compulsion and reason there for diversity in terms of what it gives our teams.

So, we know what diversity is and we know why it's important, so what concrete tings can we do in our workplace in our team to incorporate diversity? Well, unfortunately saying we care about diversity isn't really enough. Action speaks louder than words and one of the ways we can actually solve this problem, take a first step, is to reform a hiring process.

The first way, the first important aspect of when we start to think about how we can reform our hiring process is to look at the way we right job postings. Now the job posting is the primary interface of your company. It is the lens that allow prospective candidates to glimpse your internal culture. So if we want to write a really good job posting that's inclusive and will encourage diversity, what things do we need to do? Well, we need to focus on the tone, we need to ensure it is friendly, accessible and completely assumption free, we should avoid all the irrelevant clichés like rock star, hero, ninja, we're all tired of them now and they don't contribute any knowledge to job posting itself. If you employer actually is willing to have a commitment to diversity make it explicit, there is nothing wrong with being open about it because it encourages people to realise what type of company you have. And lastly you really need to focus on offering things that people would genuinely want. So, not everybody cares about having free red bull and coke in the fridge or a football table. Things that people might want is flexible hours or remote working, childcare assistance, accessible and wheelchair friendly facilities, mental healthcare coverage, trans inclusive healthcare coverage, generous parental leave. These are all truly important things people look for in a modern diverse company.

The second aspect of how we can reform hiring is to look at the way we interview people. Now the technical interview has traditionally been a major arena where implicit unconscious bias enters the picture because we tend to favour people from the same background who went to university, studied computer science and as a result we kind of enforce homogeneity. We don't promote diverse thinkers because we have a very monolithic way of approaching this.

Now, when we ask people to code on a white board or prove to us that theoretical knowledge or algorhythmic integrity we really have to ask ourselves whether me want to be part of an engineering team that was primarily chosen by its ability to write code on a white board. For me it's kind of irrelevant. That's not really what I do on a day to day basis as a programmer, I don't really know anyone else who does that. So outspoken confidence and having an in depth and theoretical knowledge of algorithms is not the most important skills a team needs.

Also think that we shouldn't be afraid of trying to redress the balance a bit because when we start a conversation or a discussion about changing hiring practices it can be really difficult with your team because it actually has a tendency to reveal potential internal problems with your own culture, culture and the teams culture itself.

So, what are the things we do to make these things better to make them more inclusive? Well, let's be conversational, let's not be adversarial. Nobody should be set up for failure. Instead we need interviewers that are more self-reflective and less alpha male, we need more effective sensitive inclusive interviews. Also let's give the candidate something that will accurately measure their performance on a day to day basis so for example pair them, give them a problem and run through a - let them run through how they might solve the problem, can they debug? What is their code review process like? Can they fix an elusive issue? can they optimise a performance bottleneck? How will they re-influence something if they have a chance? These are all relevant questions that will accurately measure their performance of your team not ask them to scrawl stuff on a white board and actually Google they found zero correlation between technical interview performance and job performance so it just goes to show how meaningless the white board interview truly is.

So, if we move beyond hire itself and start to think about some of the challenges that are still left in our teams, one of them is a culture of fear. What I mean by that is the fear of asking questions. The fear of thinking you're going to look stupid if you do, fear of thinking you don't know enough, it's a real fear of pretty much everyone in the technical industry and the fact is our industry is evolving at an extremely fast pace, we often expect ourselves to keep up with this constant onslaught of new technologies and knowledge but there is a constant threat of being left behind and out of our depth. We all face a knowledge deficit and people who start to {inaudible} are beginners, they often get frustrated really easily because they see other developers using really could technologies and they question themselves and say why bother? What can I do? I can do nothing. But guess what, we've all been there, we've all been beginners and we've all had that self-doubt. But it's OK, it's OK to be a beginner and it's OK to have these feelings of inadequacy. Learning to programme is difficult but so is anything that's worthwhile.

So, apart from beginners we also have a problem of fear with seasoned developers, experienced developers. One of the very common things we're starting to talk about in our industry is the idea of imposter syndrome which if you are not aware is the psychological phenomenon which arises from an incorrect assessment of one's own abilities so for

example if you are an 8 or a 9 out of 10 programmer you yourself only think you are a 3 so it's an inability to really understand your own merits basically. And how does it manifest itself? If you are on a team with somebody who has imposter syndrome they often won't share knowledge because they don't think they have anything worthwhile to contribute, they don't speak at conferences, they don't submit talk proposals, they don't write blog requests, they don't collaborate contribute to open source and they don't apply for jobs. But beginner's fixer these problem is more than personal responsibility, we can't ask people to solve it by themselves. I think it's endemic of a wider communal problem that we all have a responsibility to help with, we should all help each other.

So, Sasha Mundy gave a great talk at Python this year about how to give and get technical help and came up with a great line where she said: if people are afraid to ask for help it prevents from them stretching, if you don't stretch you don't grow. If people don't grow your team stagnates and your company doesn't build amazing things.

I think that perfectly summarises the struggles we all face when asking questions saw for people with the little voice inside their head telling them they're not good enough we want you to fight it, we need to have the braveness to overcome it and having that struggle of trying to overcome it too and for mentors we really should be aware of it, so if we're on a theme and we're mentoring people with no self-confidence or people who are constantly questioning themselves be aware of it, reward question asking and realise how brave it is and if they are giving and asking too many questions shape their behaviour, teach them how to fish instead of just giving them fish.

So to switch focus now to look at open source communities, the most relevant and pertinent question me need to ask is why don't more people contribute to open source whether it's Django or PHP or ruby? For me the answer is that not everybody is aware of the full range of contributing options so for example you can contribute documentation, you can write tests, you can write blog posts, you can speak about something, you can design an interface or just give feedback. These are all equally valid ways of contributing to open source but people tend to get hung up on the code slinging code they tend to think it's the only way to contribute and it's not.

Another vastly underrated way of contributing to the community is organising conferences and events because it often takes months of very careful hard work to set up and it's as much of a solid contribution as writing a new feature for Django.

Another reason more people don't contribute to open source is often people don't have the luxury of time and opportunity. Women for example are far more likely to be a primary care giver not only for children but also to ageing or ailing relatives. On childcare alone for example they spend more than twice as much time per day as fathers do and there are other reasons, some people might have medical conditions. Some people might get paid very low cannot sacrifice any contribution to do external things, some people have long commute and when they get home are so exhausted they can't contribute to open source. So I was reading a blog post a few months ago on modern new culture and the question asked was why don't you contribute to open source? And Anna who wrote the book post she said, well, I tried and people were unwelcoming and even cruel.

So this is us line from Julie Pegano also at Python where she talks about imposter syndrome and she was referring to very bad habit we have in our industry of elevating certain people to God like status, we think they're inscrutable and their code is flaw less we think they make incontrovertible technical decisions and think their status is untouchable but the direct consequence is that we doubt ourselves negate our own contributions and shy away from getting involved so her message was skill kill the heroes not necessarily that because she would violate the code of conduct but she said we need to leave the person behind and see them as a realistic person and it helps us equalise our own contributions and helps rationalise our own feelings of ourselves. Another really sort of insidious thing that happens in our industry is meritocracy the idea that power is bestowed by technical contributions but for me that's deeply flawed because it tends to rule one type of contribution. I said about multiple different ways to contribute. Meritocracy recognises code and that's it and another flaw is it assumes everyone is on the same level and has the same level of access to opportunity, time and money and we all know that's not the same, people aren't equal. So, what can we do to enable people and give them confidence in open source?

For me the biggest tip I can say is write a good contributing guide that formalises the transparent process that people need to abide by to get a patch merged; if you write it down and make it explicit there is no knowledge deficit there, people on the same level and know exactly what to do to get involved. We also need empathy and patience with issues and bug reports because if someone has taken the time to report a problem they have done that because they want to help the eco system and want to help strengthen the product itself and we should have honest intact, we should be

appreciative of the fact that somebody has taken the time to write a feature.

Another thing I do with all of my projects is I deliver certain feedback really pedantic feedback through computers so if I have a very strict policy through syntax for example so in Python with indentation that kind of pedantic feedback is a lot better coming from an automated system or computer because if somebody thinks you are rejecting their contribution because they're missing a comma, they're probably going to walk away and say I don't want any part of this ridiculous charade but if it comes from a computer because they're stupid it's OK like I will accept that feedback and change it and re-push and everything is gravy but if it comes from a person then I will probably doubt that a little bit more.

Another thing which I think is really important is to stop denigrating peoples doling interests. I am a PHP developer and every time I introduce myself as a PHP developer I can see the eyes roll back there is an implicit judgment on me and I don't understand it I don't understand how we can judge somebody's choice of tool aunt language based on our own assumptions, I think it's wrong, we need to be a lot more open. Another example is word press developers, they get a lot of flak because they have very strong opinions about the way the code is structured but the way the code is structured is completely irrelevant, they're using that tool for a specific purpose and if it's good for them it's could enough for me and it should be good enough for you too.

And the final thing to really, that is essential to help improve inclusiveness is to document well. We had this idea earlier of documentation driven development and I completely agree, documentation is at the heart of all projects so we should make potential and assumption free knowledge sharing keep component of what we value nothing is ever obvious or easy and for extra bonus points you can do things like provide non English translations, you can focus on accessibility for visual impaired users, these are all really cool things you can do to make your project as inclusive as possible for as wide spectrum of people.

Lastly to focus on physical events themselves, what can we do to tackle exclusion? Now, I think it's very important to reason Django is pretty much the shining light in our industry of how to do it right, they have a code of conduct which basically helps define and make transparent what kind of behaviour is deemed exclusionary or threatening so everybody is on the same level and can no longer plead ignorance. Codes of conduct help formalise the support system for marginalised people and allow them to contact organisers directly and also one of the things I regularly hear about the reason not to have a code of conduct is people say nobody else before has reported harassment but just because that's happened doesn't mean harassment doesn't happen, it could mean somebody didn't feel confident enough to report a bad thing.

So to wrap up, I think we should remember the 4 points I talked about today. We need to change the way that we hire, we need to remove the culture of fear, we should formalise the transparent process of contributing to open source projects and we should make an attempt supported events that value diversity.

We all have the ability to change ourselves and by association we have the ability to change the communities we all belong to. Bell hooks once said dominating culture has tried to keep us all afraid to make us choose safely instead of risk, same risk instead of diversity, moving through that fear finding out what connects it, revelling in our differences this the world that brings us closer gives us a world of shared values a meaningful community. I will leave you with that quote thank you. {applause}.

DANIELE PROCIDA: Thank you very much I think that was the perfect way to end today so thanks.

Just before we all leave the room a couple of very brief announcements.

Actually I'm so sorry, Jamie, sorry, I didn't ask - there must be some questions for Jamie. I'm sorry I didn't mean to - I'm sure there must be questions for Jamie come and stand here so - yes please.

NEW SPEAKER: Hi Jamie thanks so much for that great talk. I get from the sense of your talk that you are deeply into diversity because you think it is the right thing to do but I was interested in how you kind of led the beginning of your talk with the idea that - I mean I kind of promulgated it in my talk too like when you have diverse teams we make better things and so I'm wondering do you ever kind of use that line to kind of finesse diversity to some people instead of - because I come from an ethics background I believe diversity is the right thing to do. Can you talk about the way you play with that sometimes?

JAMIE HANNAFORD: So, I think that it is very important when you believe so strongly in something and you are

trying to make changes to be sensitive to the fact that not everybody is going to have the same ethical framework as you. They are going to think differently because they are going to have different moral and political beliefs so I think it is important not to hammer my framework on somebody. I find the presenting facts and psychological study is kind of a neat way out because our industry loves numbers, they love stats, they love very technical things that we can't argue with, if you give them that, how can they say no? It is a better way of trying to achieve change than just trying to talk about ethics but I think both are important.

NEW SPEAKER: Truly tremendous talk, really well done. It was really refreshing and I think it is a real testament to Daniel's vision that we have got so much inclusivity at this meeting.

(APPLAUSE).

I am a research astronomer and we, there is a group of us who crossed over into this sort of dark world of software writing and the some of the lessons that we brought back are including things like codes of conduct in our meetings and it is really, really important to us that you guys are leading the way in this. Because we can benefit from this. The epidemic community for a long time tried to a lot of the hinges, like you said, nobody has said they have been harassed, why? All those sorts of things so, that was really what I wanted to say that it actually it is wider than just this community. It does trickle back into other areas and it is really, really important certainly to us.

JAMIE HANNAFORD: Some of the problems I talked about some doubts and voices, they are applicable in every industry, I think it is great that our tech industry can be seen like an archetype of how others can act, we all have a responsibility because we are on the front lines, we are the ones defining future generations of other industries too. We need to pitch in and make it better.

DANIELE PROCIDA: Last one.

NEW SPEAKER: Thank you for your talk, so my question is, if you have managed an open-source project, ... a guideline not just for the coding and the patches for doing research, ... and all sorts of contributions.

JAMIE HANNAFORD: Are you talking about personally? Do I – have I seen others.

FROM THE FLOOR: Have you seen?

JAMIE HANNAFORD: Not that I can recall, I can look into it. The types of projects I maintain I make it key priority when I start them to focus on unfortunately just on code contributions but now that you have raised this issue of different types of contributions I think that is something I will start thinking about with my projects too. I can be on the lookout if I see ones that are of particular interest.

DANIELE PROCIDA: Right thank you once again.

(APPLAUSE).

Okay, it is the end of our first day of our open day. So I have got a lot of thank you'd, firstly thank you all for coming especially if you weren't part of the conference and came to visit our open day, we are really glad you are able to be our guest here today, maybe we will see you at some other events.

Thank you to our speakers and the workshop leaders today, who have, I have seen most of the talks, not seen all of them. The workshops were all really busy, so thank you for everybody who put so much effort into that too.

Our volunteers, are any of the volunteers in the audience? No, well you will have to thank them, look, yes there are, Damien has been yes, you Damien, you will have to represent the volunteers they have done a fantastic job and grateful for everything they have done.

The people who provided technical services, our speech to text reporters, people doing the filming, people looking after the AV for us and people looking after the network for us, so thank you very helping keeping today running.

Once more, thank you to Cardiff University if you can send a little thank you to Cardiff University that would be nice, they have put an awful lot into this event. Who is a sponsor here? Representing a sponsor, come on I know you are, thank you very much because you are also one of the, you also make this possible it would not be possible or not possible like this were it not for the involvement and active participation of sponsors, thanks very much to all of those. (APPLAUSE).

I hope you have enjoyed today, I hope that you will be able to come to one or the other of the meals that we have got on tonight. You do need tickets for those, tickets for the VFS are still available from the tickets page on the website. Whichever of those venues, be there by 7:15 this evening, if you are going the the VFS, they don't serve alcohols but you are welcome to take in your own bottles. They don't serve alcohol at the clink for entirely different reasons! see you later or tomorrow!

1.3 Talks day one

1.4 Talks day two

1.4.1 Introduction

DANIELE PROCIDA: Good morning everyone for day two of Djangocon Europe, welcome back if you have just arrived welcome to Djangocon.

I just want to remind you about the wellbeing support sessions that we have arranged. The appointment cards are on that little partition at the back just pull off a post it note and you have got your appointment with the room and the time. So help yourself to one of those. The counsellors from the university will be here between 10 and 12 and between two and four.

We had two or three very strong and candied statements made by speakers from this podium yesterday about mental health and wellbeing matters that had affected them personally.

I want to say that I was having a really awful time a few years ago; when I was working for the university I was able to make use of that same counselling service. It didn't solve my problems for me. But, it did give me a bit of space and time and a way to think about them which helped, helped me find a way to deal with them without being maddened by them, which is what you can expect from a counselling service. Nothing is going to solve your problems by itself, but I am glad I did that and if you have been affected by any of the things that other speakers have already talked about in the last couple of days, then, feel free to take that as a first step.

I hope you had a good night's sleep, I did until I had, until my dream and I dreamt that I was running a conference and that there was a code of conduct violation caused by Mick Jagger! Who was writing down lists mocking other people's names and Python code and then I woke up (LAUGHTER).

So maybe I am becoming maddened in a slightly different way.

We talk a lot about community and you have all got a special membership card on your lanyard for the community now, I hereby invest you with the power to induct anyone else into our Pythonic sister and brotherhood, into the international society of Djangonaut, before you go home, get the cards, slip into their wallets, so when they get held up at a border or held up at a nightclub, they can show this, yes please, you are very welcome, so we will scatter them around, take some of them and we are the most inclusive society in the world and let it stay that way.

I am sorry, I meant to show you, there we are.

Take advantage of that.

If you can't find one, come and see one of the volunteers.

I mentioned very briefly that we would like to try to get together people who are working with Django or in Django or and in some way have a project that is aimed at increasing social value or social good. We will have an informal meeting in here at the back of this room towards the end of the lunch break, so we have got half an hour, tomorrow at 2:00 o'clock, so if your company is doing something, if you have got an idea, if there is a need that you think could be solved with a Django project somehow, we will come and talk, maybe we can set something up that, some kind of umbrella, some kind of framework for organising ourselves and see what comes out of that.

I don't know what will, but at least we will know who we are.

The first thing of course that we might be doing is returning the favour to the counselling service of the university by collaborating with them during the sprints to see if maybe we as Django developers can create something that solves a problem for them, we don't know what that is, but if we talk to them, we will find out.

About to welcome our first keynote speaker of the day, I will let Ola introduce her, thank you, enjoy the day, see us if there is anything that you need. (APPLAUSE).

1.4.2 Ola Sendecka: Into the rabbit hole

OLA: It is good to welcome Ola today, she organised Djangocon Europe, in Warsaw together with me, also Django core developer, started Django girls, a huge organisation that is now changing the way this kind of conference is being run. So, she is going the give a wonderful talk about getting into rabbit holes and it is wonderful story and how to get out of the rabbit hole which is probably the most important part. So yes without further ado, please give a warm welcome to the sweetest person I know, Ola Sendecka.

(APPLAUSE).

OLA SENDECKA: Good morning everyone. Do you hear me well? Okay.

Amazing. I am extremely happy to be here, I really appreciate that so many of you came, it is like the second day of talk and there was a party yesterday, so it is really amazing so many of you are here. I really want to first thank Djangocon organisers, especially Daniele who convinced me to give a talk, turns out it is a keynote, I am extremely excited about it. You can hear that in my voice, I am superhappy to tell you my story; I have so much stuff to say to you. So. Yes. It is Tuesday morning you probably had a great time yesterday. You are probably sleepy; some of you have coffee and some of you not. So, the good news is that I will start with a story.

This story will be about Alice in Djangoland obviously. Little disclaimer, if you don't follow all the code examples, please don't worry, because they are not the most important part of this talk. So, if you are lost then it is fine. Actually there are a lot of bad code here, so maybe it is even better if you don't follow it.

So are you ready?

Alice was beginning to get tired of sitting by her sister on the bank and having nothing to do.

I am bored, I want to do something interested, everybody is occupied, I half hour and I have nothing to do.

She was having this little project and tempted to start coding even though she was visiting her family, she felt a little bit guilty about it because yes, she was not visiting her family too often wanted to spend some time with her sister.

But she had this free half an hour and her project that was not finished, so, she decided to herself that, well, nobody will really notice and she will be done before afternoon tea.

She was working on this project for some time already. It was quite big, tightly coupled, lot of legacy code, she felt very confident about the code. She knew that quite a lot so she knew that she can do everything very quickly.

One part of the project was basically tracking time so she could have some projects and then she could lock time, how much time she spent working on the project.

The project and many, many other models were very special. They were special because you could archive them. So, by default, you just work with active instances but the archived ones are still there and you can still in some scenarios see them.

And it was done like this. It was not maybe the best way to do that but it was already there implemented so she had the class, the class called archive bubble that was saying if they had the flak saying if something is active or not. Then all the models she wanted to archive inherited from it and override the objects manager and filtering out archived model instances.

She could track time, logged the time, how much hours spent on which project and after the project was done, she could archive it. So later on when she entered new time, she didn't show the first project with the list at all.

She had this list where she could look what she did last month or last year. Recently she looked at the list and she saw this line. Maybe you haven't noticed what is wrong with this line? But Alice noticed this very quickly because she hate typos, there was typo in the description. So she clicked the edit button, this is what she saw, where is my project she asked?

What is wrong? She felt a little bit and she realised that the model form she used is just taking active objects active instances so obviously it does not show up in this select. Because she edit something that is pointing to the archived model, archived instance.

So now editing that was not possible and she really wanted to edit this. Alice started to think what she really wants and what she defined it this way, that if she wanted to by default work with active instances, so wherever she had the form with the select, she wanted to have list of active instances and the current one. So if the current ones was archivable she wanted to save anything in the form and save it. She didn't want to change many, many forms she had. She started to think about it and decided she will create this function outside of any forms and she will call it preserve existing choices that will take a form and she will iterate over all the fields and do some magic, so no matter what is the field, no matter what is the projects, is it a project or something else inheritable from archivable, it will do it for her.

She thought about query sets well, most of her forms were model forms and Django is doing all the magic for herself so, she felt yes, what I want is to have current query set plus the current value of the instance.

So, she started to draft in there that and she decided that in her function she will just take the field and override the query set with extended query set that is taking current query set plus current value.

She started coding hoping that she will be done very, very soon. After a while, she had her first sketch ready. This is how it looked like, extended query set inherited from Django query set. Passed the value to the init and then saved it, then after iterator and then after returning everything that is there, yielded the value. Didn't check if the current value is not already returned but just wanted to check if it works.

Yes, she was happy with that. She was yes, convinced that she will just go to her sister and speak with her and very soon. But she refreshed the browser and this is what she saw.

Something was wrong, yes, that was for sure. She started the bugging. She looked back; she went deep into Django, step by step, PDP and printing different kind of things. The things didn't work as she supposed they will. Alice tried to override different parts of query set but it was retrieval. Fixed one place and another thing was broken. After a while she knew that it is not the way to go.

This slide is very Polish!

(LAUGHTER).

During her trials she noticed the error happens someone in model toys iterator, what if I, if I override model choice iterator. Why not? Had this idea that she wanted to try but in this very moment her sister came asking Alice do you want to have tea with me? Alice said, I need to try one thing, I will be done in 10 minute cans you wait for me? Yes sure. Sister left, Alice started to code and minutes passed and after some time it was ready.

This is how it looks like; it is very similar to the query set. She passed the value to the init and then after everything was done yielded the choice that was, that was the current choice, once again she didn't bother to check if it is actually active or not. Just wanted to check if it works.

Then in her preserved existing choices function took the field and changed choices and create extended model choice iterator. She refreshed and superhappy. Now it works. Her project appeared here and she was supersure she can go to her sister and have nice evening with her family.

But then she tried to save the form. She realised she was wrong. There is a validation error. No! She cried.

She felt frustrated, angry, and her sister came again asking Alice, can you just come? The tea is already cold. Alice was really irritated and she really wanted to fix that. So she said, just give me some more time. I will be soon.

So her sister went away once again and Alice started to think and debug again.

She finally reached a definition of model choice field. She decided that the error must be somewhere there and without even thinking she decided okay, I will override the model choice field too, I can do that, I already override the iterator,

why not do it with the model choice field?

She did. This is how it looks like. At this point she didn't even bother to change, to make something smart just copy pasted Django code to model choice field, she checked the value is the current value of the instance and if so, she returned that.

Then in her function preserve existing choices one she extended, basically, she created a field and put that inside the form.

Now, Alice said, it must work! She refreshed the page with the form, save it. It worked. Alice felt it is over now. There she had overridden the choice field, iterator and model choice field. Kind of ugly but it works, she tried to convince herself and she is sure she is done.

But then, she notice another thing, she wanted to change one of the fields pointing out to the archive bubble object and instead of this; she ended up with this thing. Oh no, no, no! Alice started to tear her hair out of her head! What now! On top of that her sister came, she was annoyed, you came here to visit, why you code and code all the time?

Alice just she was superfrustrated and superangry, said, just go away, I need to fix that, it is very important. Her sister left angry. Alice felt pressure and she felt very, very guilty. But she continued. She looked into the code again and she understood that yes, creating a new field it is not the way to go. Because she was trying to do that outside of the form and she had no idea what was the widget used or any other thing. When she looked into definition of model choice field and in it, she was thinking maybe I can pass it somehow, take it from the old field and put that into the new one? Well it was not way to go, so many things were there.

I can do that. I can, Alice was sure, there must be a way, I need to fix that somehow.

So she thought a little bit and started to think in a weird way and decided what she wanted was different to Python not the whole field. If she could just take the different implementation and inject it into the existing field that was created by Django, yes, I can do that. So she did.

So she took the whole Python method from her old extended model choice field and she put that into function custom to Python. Then in preserve existing choices she did this, so she took field and put the new cast into Python, using partial to Python.

At this stage it finally worked. Alice decided to leave it as it is. She couldn't spot anymore errors and felted really frustrated and angry and her family was angry. Spent hours and hours of her free time looking at that. She was sure that it was supereasy at the beginning like, half an hour job.

The code was ugly, weird, confusing and partly working, just like a house of cards, have fragile and ready to collapse any time.

She committed that repository and felt superashamed every time when she looked back and others could see that too.

She was not even sure what she tried during this way everything was blurred, had weird errors and felt frustrated so much, every step, so many unexpected things happen. In the end she spent hours making something that was not working. It was, it seemed easy at the beginning but, in the end, it was band of horrible code through repository.

What happened? Asked Alice.

How I end up doing all, all this horrible things?

Exactly. What happened? This is very good question and yes. Rabbit, Alice was in the rabbit hole and I know that many of you have intuition about what rabbit hole is, but let me rephrase it for you, or define it. If you are not familiar with it. So, the rabbit hole is the act of getting so caught up in what you are doing that you spend precious minutes, hours, or even days on doing that and it negatively impacts your overall productivity.

When you fall down into rabbit hole with the certain goal in mind, you get side-tracked by various events and you change direction many times along the way and eventually you end up somewhere unexpected and very often without satisfied the original goal.

So the rabbit hole is kind of metaphor of entering, disorientating, confusing land and a moment ago you could follow the Alice steps following the rabbit into this strange world of query set, model choice field, and the strange thing is

nothing seems to work as expected and he got more and more confused and more and more frustrated.

Some can argue that rabbit hole could be a good thing, that you are in a flow and you discover many unexpected things and you can learn from that.

It's true, but it's very individual. And I must confess that every time I end up in a rabbit hole I end up feeling frustrated, angry, helpless, and stupid.

Next time I look into the code I know I would do that differently, like I have no idea how I end up doing all of this horrible stuff, like it was wrong at the very, very beginning. And I know that many of you already have some cool ideas on which Alice went wrong and how she could do that better but really it's not the part, like it's not the main thing in this talk because believe me I really - yeah I had so much stops and it pained me when I wrote this presentation and wrote this horrible code for you.

But this is the point. Once you are in a rabbit hole you don't really think and you're so confused and so frustrated that you don't see obvious things at all. And for Alice it was model forms. But for you it could be different library, something that you don't know yet, or maybe you have hard problem to solve and you have the worst day.

To understand the rabbit hole, I tried to read things and, yeah, I wanted to somehow deal with it much, much better, and there are 2 things I want to discuss with you briefly and I believe it will give you a better understanding what's going on when you are in your rabbit hole.

The first thing is Zeigarnik effect. I don't know if I pronounce that correctly. According to Balmeister and Bushman, the Zeigarnik effect is the tendency to experience intrusive thoughts about something that was once started and never finished. So even though you try to pursue new goal you cannot stop thinking about what you left behind something unfinished. In a way it's reassuring because it kind of means we are built to finish things but how many times you end up abandoning your project and not finishing something and you feel guilty and you keep thinking about it and you feel bad about yourself. Experiments in this area suggest people remember unfinished tasks and puzzles much better tan solved ones so we can say people are built to finish things.

However there are times we need to realise that we have invested too much. It's very, very hard to objectively judge it use especially when we feel responsible for the project. And this brings us to the second phenomenon which explains rabbit hole which is sunk cost fallacy. I expect some of you know the term but if you are not familiar with it it's a phenomenon that your decision or you are influenced by the emotional investment you already accumulate. And the more you invest in something, the more time, money or resources, the harder it becomes to abandon it.

So, if you work on a project and you go nowhere and like Alice you spend a lot of time that you should do something totally different you just feel you cannot abandon it because then everything you invested, the time you sacrifice it's not always your time only but time of your family. It would be like sacrifice for nothing so you don't want to waste it and you keep going, keep going and the damage is bigger and bigger.

One of the features of sunk cost fallacy is that it's connected to personal responsibility and it appears to operate mainly on people who are personally responsible for investing the resources money or time. And this irrational escalation of commitment is a very dangerous thing and it can make you miserable. You will end up doing unwise things and you invest more and more resources on something that has no chance to succeed. And you will feel worse and worse about yourself.

So, being in a rabbit hole is possibly very, very dangerous and you waste your time, effort and wellbeing for it.

So, let's talk a little bit how Alice end up in a rabbit hole in the first place?

First of all, she assumed the problem she's fixing is very easy and because Django was doing everything for her magically she assumed that this must be easy, she expect easy solution and therefore she did not see all the implications at all.

She was also fixing symptoms and she didn't look at the problem as something bigger. She had no bigger picture of the problem.

She also continues even though she failed so many times, she tried override query set and failed, tried override model choice field and failed and then override model choice field and failed. She should have realised something is not right much, much quicker.

And all this time she was super, super sure she will be done in 5 minutes, just give me 10 minutes I'll be done and she was sure solution is one step away and she had something else she didn't expect and she was confused again.

Finally, she couldn't give up. She was very persistent and she really want to solve the problem. It is possible that Alice didn't want to admit that she cannot solve it on her own. She was too proud or she want to prove that to herself. But rabbit hole is very dangerous place and maybe it's fine when you do that, like that you are in a rabbit hole working on your side project to discover different things, but if you work for your client or you work on open source and then you commit something that is horrible and people see that then, yeah, I'm not sure if you'll be very happy to see that out there, people commenting, so I think rabbit holes can make people not contribute to open source that much.

So, how one can fight with rabbit holes.

First thing is to realise that you are in one and believe me it is not very easy. So many times I'm asked are you stuck are you in a rabbit hole? I said no, no I will be done in 5 minutes; I just need to fix the error. I was in rabbit hole obviously.

So I wrote down some symptoms that I have, so it's not very extensive list and I believe you could have different symptoms too. But here is the list:

You don't know when time flies. 2 or 3 hours feels like 5 minutes. You are always almost done. And you are hungry and thirsty because you forget to eat anything. You are so fixed on solving the problem you ignore everything around including your body and its needs. And when you finally try to make a break because you are starving or you really need an extra coffee you cannot stop thinking about it you go to the coffee machine, starting to make it and you think, what if I do ... if I check his barrier, or maybe this is wrong? And you do this debarking session in your head and cannot stop thinking about it and then you go with your coffee and try to see if you are right.

If you haven't spoken with everyone for a longer while you can be in a rabbit hole. It is tricky though if you are a remote worker as I was for a long time but yeah if you haven't even chat with anyone, yes it's one of the symptoms.

And finally, you feel extremely unhappy when anyone dare to interrupt you. A colleague, your boss, neighbour, or even a food delivery. You are cranky and very, very irritated and you want to kill them.

So if you suspect that you are in a rabbit hole what can you do? And getting out is not easy. Once again it's not trivial.

And this list is also not very long and I still try to find the better ways to get out of the rabbit hole but here are some of my tips.

Ask yourself how much time you need for fixing the next problem you face and be honest and say, if you don't know, if you don't understand what's going on just try to think how much time you would like to spend maximum and just set the timer and be very strict about it. If the timer - if {inaudible} just make a break.

Make a list is a good idea because it force you to think what could be wrong and to think about it not as a solving error you face but to think what you try to achieve?

So you will have a bigger picture and you will not go in obvious bad direction like Alice did a couple of times.

Obviously rubber duck works too so if you say to somebody or rubber duck, you can also paint rubber duck in paint if you don't own one, so if you try to describe what's going on and try to explain to yourself what you try to achieve and what's happening may be you'll notice something more.

If it's still not working take longer break. But make sure that you are not doing the bugging session in your head so, yes, an hour swimming, or I know climbing, or just meeting with friends, going to the cinema, making a longer break like one or 2 days, it is really, really helpful.

And finally and I think this one works the best for me, is asking for help. You might be proud and you would not like to show to somebody that you spend hours on fixing something weird and not understanding what's going on. Honestly, being in a rabbit hole is potentially much, much more dangerous and it will undermine yourself confidence a lot. So it's better to ask for help as soon as you feel that you are stuck.

And if you don't work with people who willing to help you or will judge you because you do weird stuff sometimes in new code you probably should change your employer. If you don't have colleagues because many of you are self-

employed or freelancers you can ask your friends or ask on internet. Sometimes even explaining things to somebody who is not a programmer is very, very good idea.

One more tip is to have somebody who is not doing their own debugging sessions in their head so they are not in rabbit hole because you can end up convincing them that you are doing something smart. It happened to me.

Rabbit holes are scary but there is one positive thing and good news for you if you ever found yourself in a rabbit hole. It means you are very, very curious and very persistent and really want to finish what you started and it's a good thing, but being in rabbit hole is something really, really horrible and it can impact you in many different ways and make more damage than you expect. It can make you feel bad about your code about your skills, you can feel dumb and worse than everyone else and just because you have the worst day and had nobody to ask for help you are just fixer and you are in this strange confusing world. So don't be afraid to ask for help when you're in a rabbit hole.

I'm Ola Sendecka; I am Django girl founder, Django cofounder. Many years' experience; I work for amazing company called Potato but still I find myself in a rabbit hole from time to time. ; They make me feel bad about my skills and undermine myself confidence. That's why I am super happy to hear about your tips and tricks, how to become unstuck when you are in this horrible funnel with only one way out through the rabbit hole.

I would be glad to hear about your stories and your ways of avoiding noticing and getting out of the rabbit holes.

Thank you. {Applause}.

NEW SPEAKER: So if anyone has any questions or suggestions there is a microphone in the middle of the room. I can start. So you said you are working remotely for a long time so where are tips for remote workers and how they can make sure they don't fall into that?

OLA SENDECKA: What worked for me is to communicate with a team very often and if you cannot have hang out open all the time with your team you can try chatting with them a lot. It's tricky though when your time zones are different. In his scenario I think like taking longer breaks here and there and having something that distracts you and you cannot just put that away helps. It's very tricky though and I am not sure I am doing that very well so that's why I'm talking here so I hope you have some amazing ideas and I can learn from you. But yeah taking breaks and asking for help, writing an email sometimes helps so if you have somebody in your team in a different time zone and you try to write in the email what's going on what's not working for you then you can during writing the email you can realise what you are doing and that it's insane basically.

NEW SPEAKER: Is that on? Good. Thank you for your beautiful talk and beautiful storytelling as well. I appreciate the artistry in telling a good story.

I am not really - I feel a bit of a fraud being here because I'm a research astronomer and I consider my code to be hacky and really like tinkering, but I am a remote worker as an astronomer and I have to interface with a lot of professional software engineers and I feel like a bit of a fraud, so I don't ask questions when I probably should. But also a lot of them are Java programmers and can't help my Django questions so because I feel a bit of a fraud asking them for help and they can't help, I don't really know where to look for other help, so I write even hackier code. Do you have any questions that will help me stop going down the rabbit hole?

OLA SENDECKA: I am not sure if I understand correctly because I'm stressed - no sorry I'm excited. This is my trick I'm excited not stressed! So you ask where to find help if you don't know where to find, ask to help.

NEW SPEAKER: It's knowing what questions to ask so you don't -

OLA SENDECKA: Yeah so I think like asking the internet is a good way to start or starting to explain things to even to someone who is not in the field. So, it's good to have a mentor or a colleague even if that's like remotely so make connections with somebody and have a network of people who are willing to help you. I know it's not a perfect solution if you don't have anyone to speak right now, but yeah. It's hard and I still struggle with that. So I would suggest trying to find somebody in the internet even. Try to find person who is doing something similar and try to contact them and I don't think a lot of people will bother if they don't want to talk with you, they will just not respond to your email, but maybe you'll have, yes, end up having nice mentor or nice peer to talk about things, even the emails or something like that. NEW SPEAKER: How would you suggest finding those people? Just by looking at this sort of community and - people don't have little like floating bubbles with their contact details but I try to talk as many people as possible but it's always tough.

OLA SENDECKA: So I didn't hear that well but about finding people I think it's very nice to go to the conference and meet ups and be part of the community so if you do something and you are alone with that you are like - it's very possible that you end up in the rabbit hole more and more likely. If there are any meet ups and or forums or mailing list, just be active, if it is like, if you don't have a question now, maybe you will have in the future, if you are active and try to help others, it is likely they will help you once you are in a rabbit hole. It is nice to have a network of supporting people, I encourage you to speak with people at the conferences and meet ups. Write to the people who are your role models, like make your community bigger.

FROM THE FLOOR: This might just be me, I don't particularly mind the rabbit hole thing, when I am in the middle of it, I am not happy about it. I find at the end of it I have learned something, do you ever think you have learned something from the whole process, dug into model forms code, otherwise would never have looked at and understood how it worked.

OLA SENDECKA: It is not that the rabbit hole is bad, but if you are wasting time like you wanted to spend with your family or do something else or you are working on the task that your clients want to have something done, but, you estimated that for one hour work but then you spend one day and you wasted clients' money or you wasted your money if you are a freelancer, it starts to be not worth it. So it is about the balance and it is, I think it is very important to know that you are stuck and you are in a rabbit hole, then decide, is it really worth to dig in and to find the solution? Or to like experience all this unexpected things in wonder land? Or are you have no time and you don't want to follow that. In case of Alice she was frustrated not in the flow, not happy with what she is doing but felt guilty there is pressure and she didn't want to be there. So, I think it is, it is important to realise that you are in one and to consciously decide if I want to follow this path or not.

FROM THE FLOOR: It is as much about time management type of thing?

OLA SENDECKA: Kind of. But also, it is like you are, in your project in your life, I think that often it is that you are not realising that you are doing something bad; you are just fixed to do something, to finish the task. You don't really pay attention to anything else, so then in tend you have something that you are ashamed of and you feel bad about it yourself and it undermines your self-confidence and then you think you are stupid but you are not, you are just following the wrong path and you are superconfused in the way.

FROM THE FLOOR: Thank you.

FROM THE FLOOR: Okay, all right okay, so I have got one thing I was going to say, more a comment than a question which is that for anyone here who is as old as me, and used to programming in CVS back in those days in the rabbit hole, made 15 commits, it was difficult to come out of it. With the horrible things you have done and put that in there and start again from the beginning of the problem you were supposed to be solving in the first place, quite often forget that the other branch was there at all. Sorry that is not really a question.

OLA SENDECKA: So we are running out of time. Thank you, that was amazing.

(APPLAUSE).

1.4.3 David Gouldin: Django's role in the polyglot web

OLA SITARSKA: So, our next speaker is David Gouldin, he is using Django since 2007 so quite some time. He is currently working at start up in San Francisco so a long way to Cardiff. He was going to talk about Django's role in a polyglot world so give a hand to David. {Applause}.

DAVID GOULDIN: Let's get started, if you want to follow on with me this is actually a hero: app as well, let's see where this is going ... sorry about the technical difficulties. I may have to go without my speaker notes here ... of course going to give up on the speaker notes. That's fine.

Yeah so let's start with just a brief history about Django. Django is 10 years old as of this month. It was started in Laurence Kansas in July 2005.

Django was originally built as part of a CMS project called Ellington. The Laurence journal where people decided that they would split out the code base into 2 parts, license Ellington the CMS and open source Django as a more generic web framework.

So Django's mantra is perfectionists with deadlines. This kind of sets the trajectory of the entire project and you can map all of Django's strengths and weaknesses to this sort of ideal.

The web is a very different place in 2015 than it was in 2005. Do you remember the days when you could describe your entire application stack using a single acronym? You could say I'm a lab developer and people knew what that meant, usually had one day a store, usually relational, your service side code encapsulated all your business logic and views on the server were used to render your entire HTML pay load which you'd send down to the server. A very simple design.

So we've moved from a web that looked like this which by the way Django is developed to solve, to one that looks more like this. Weight, that's not right. Something like this, yeah. OK.

Much more service oriented where we have public and private services that are talking to each other via http, we can also use something like pub sub when we need to communicate one to many, one to many other services but most importantly our client in this model is just another service so web browsers have advanced to the point where we can treat the browser as a service consumer from our public services provided by our server.

This changes dramatically people's expectations of what the web is and how it acts. People now are used to dealing with web applications like native applications so there is an expectation of responsiveness that comes from these interactions and server side N VC just does not adequately fulfil these expectations any more.

So let's talk about a few of Django's strengths. First of all, Django is written in Python. I don't think we can really under represent the strength that this provides the framework. The Python community is super deep and super rich and provides all kinds of libraries for us to use that other less mature less longstanding communities just don't have. This is a huge advantage for Django.

This same perfectionists with deadlines mantra means Django is built for getting stuff done which I really appreciate and you really appreciate when you work with frame works that weren't necessarily built with that in mind. So Django makes some compromises to achieve this. One of its most famous compromises is in its orm. Django's orm is kind of blamed by some people for not being entirely expressive or inclusive of the entire SQL standard but I find I can get most of what I want done in it pretty quickly and easier then the orm allows us to build other tools on top.

Ola spent her entire talk talking about how powerful and expressive Django's admin is truth is nobody really wants to build this stuff from scratch, the fact that this is included with Django is a huge advantage.

The Django rest framework makes it easy to build on top of the orm and get cloud working for resources very, very quickly and then allows you to continue from there to create customer actions.

If you've heard me talk before you know a live celery. I feel it's a vesting class application of synchronous work cue which pretty much any non {inaudible} application needs in a function. The fact that you can have this pretty much out of the box with python and Django puts it leaps above lots of other solutions.

But for all of Django's strengths it obviously has weaknesses. The most I guess talked about weakness is the C 10 K problem. For those who don't know what it is it's handling 10,000 concurrent connections to a single process. And Django just does not do this very well. Python or at least Python 2 wasn't built with massive concurrency in mind and Django was built on a Python 2 world so your alternatives here are to use something like A sink IO, or twisted which gives up a lot of the library support which makes Python strong in the first place or it's monkey patch the world with something like G event which gives you all kinds of edge case problems. You know it's patched event loop.

So with the proliferation of client side tools comes the necessity of proliferating Java script as well because Java script is a de facto language in the browser so when you are developing on client side Java side tools you enter this community that expects Java script everywhere, expects Java script on the server too to take full advantage of the tools so you are hamstrung if you are not willing to run Java script on the server.

Look at problems you could run into trying to run the ocean with Django. Pushing the web clients is a very comment problem for modern web applications and Django doesn't have a good solution to it. You can try to use long pulling but during the time your connection is being held by the server you're consuming a precious Python web worker and this is a really good way to deal less your server really quickly.

However if we look to the wider community we see there are lots of options for solving this. Socket IO is the most

famous one for web socket communications. There is also built in web socket support with go, which uses not blocking IO. Native solutions like web sot {inaudible} hold the web socket open. Things like using tornado where they sink OI or twisted like I mentioned before.

So, let's say that you your IPA has a point that issues a celery task but for your clients who continue with its interface it needs to get the results of that task before it can move on.

This is common problem in like search engines. Let's say building a travel search site you want to execute a search but you can't show anything to the user until you have the results of that search back. How do you solve this problem with Django?

What I'd do is I would organisation meant my API response with a list of celery task IDs that were launched as a result of that end point and then create a separate end point to allow me to get the status and result of each of those tasks then set an interval in the client to check you know check over and over again for the results of those tasks and move on once I have the information I need but there is a more elegant way to do this.

If we build on the ability to push the clients, then we can treat a web socket server as a bridge to pub sub and if we are standing up pub sub on the service with something like Reddis our ancillary workers can push direct to it using {inaudible} as channel name then our client can using web socket's subscribe to tasks they receive through the same augmented API end point with task ID then have information streamed down to them as if they're a pub sub subscriber to that task and then all of a sudden you've got this complete push loop going on where your client can in real-time know updated information from asynchronous processes on the server.

As I mentioned before there are a bunch of Java script specific tools which are helpful when building rich clients not least of which is server side rendering of your initial page state. This became possible with reacting has been adopted by ember and some other web frameworks but to do this you pretty much have to have Java script running on the server. In the early days of react Eric was telling me he tried to sub possess out to node in order to render react components to grab the HT mode back and send it down through Django and this is obviously just not an optimal solution. So, I would not recommend doing this, it's very slow.

If you on the other hand are willing to run no JS to serve directly to your client as well as Django API side by side, then you can just use the native reactor render to string and have your node web server actually serving your client application. Remember you are treating the browser as just a service consumer so it doesn't really matter where that application pay load comes from, it just matters that it's responsive, it matters that it is fast and you can work with different services side by side. So in this world you would probably want to use something like cores to be able to have a node on may be a dub dub dub sub domain and your API on a separate sub domain and have node render the initial pay load of your page, sent it down with all of your static assets then treat the Django part as simply an API provider for the client.

So as soon as you have more than one service serving web traffic you have the problem of shared authentication with the client and there are 2 mechanisms you can use for statefulness in a web browser, there is cookies and local storage.

You can refer your off to Django contrib and just using it willy-nilly happily. You can go to an off load in page for your client, get a set cookie back in response, set your session ID on a domain that again will go to both your other public services and to your Django service and then any requests you make to your other services they can pull that session ID out of the cookie header and actually defer over to Django to ask who is this user and get an authentication response then be able to have authenticate user for their service.

If you don't want to use contrib auth because you don't want multiple methods for KPI you can use it directly same sort of mechanism, issue and access talking to your client, your client either stores it in cookie local storage sends it up on API requests and also request to the initial page though if you are using local storage you can't take advantage of authenticate service side rendering because you need {inaudible} pull it out of local storage and send to it node that's something to keep in mind.

Another option is defer authentication to a separate service which is the {inaudible} SLA approach because all of your public services are treated equal and but there are moving parts involved and it's more complicated to get this up and running so you may prefer the lighter approach.

In conclusion, say no to full stack by python I know a few years ago there was an idea we should be pursuing ass a

community full stack Python, Python everywhere, this just isn't feasible for the web and it will cause you pain if you try to solve all your problems with Python. Choose the right tool for the job. Not all nails are - I don't know what - mice, no, not all nails are mice.

HTP is your friend. All of your web services know how to speak it how to listen to it, use it to your advantage. Talk your services in http and when you need something more flexible than direct one-to-one communication use something like publisher subscriber model for your letis(?) for your post stress for your cofca(?) whatever.

And think about web browsers as service consumers rather than as special displayers of content or something. Web browsers are sufficiently advanced enough at this point and the tools are sufficiently mature that you can build much more modern web application architecture by treating your client as an API consumer or as a consumer of your public services rather than trying to make it more like a dummy terminal.

So a little pitch, heroku is hiring got employers all over the world, if you are interested go to the careers page. Now I will take questions if there are any. {Applause}.

FROM THE FLOOR: Hi, so, the polyglot stuff is really cool and great but screams often over-engineering to me, how do I go about working out when my website is big enough worth, to be worth bothering to do all of this. Writing the API's, all the services, front end, is a lot more work than is necessary to whack up a single Django only site, do you have advice on how to get, when is it worth starting to moving a service ...

Dave: If you are building a Blog, you don't need the real-time responsive stuff. But if you are building Google docks, then obviously you want something that acts more of an application, to what degree does your project look like a website versus a app. There are shades of grey in in between, the more it looks like a app, if more people expect for a native application.

Be thinking about is, my product also a platform play, like is it going to need an API anyway, if it needs an API then you get a lot of advantage of dog feeding that in your web client. You are using your API all the time, you will feel the pin points and make it better.

RUSSELL KEITH-MAGEE: I would like to get your details on the assertion at the end, Python can't be used for that right now the tools are not there, not anywhere near as mature, do you think that is a fundamental limitation of Python that it shouldn't be like that or never develop the tools or something inherently good about java script, I can't believe I said that word.

DAVID GOULDIN: You are writing java script on the client, if you want to write V eight in Python, I don't want to.

Python can do all of this stuff, Django itself is not particularly well suited to things like you know, web sockets and A syncI o, there are a lot of ..., weren't built with the con currency in mind. A sync io are promising and it is possible we could get to the point that the only benefit of doing the ... is for isomorphism.

FROM THE FLOOR: Making dynamic responsive applications is something that is obvious for us but for example, in my company, each time we are trying to do something new, right front end ... java script, somebody from a department that is responsible for (INAUDIBLE) comes to us, no, no we need to render everything on the pages, needs to be traditionally.

DAVID GOULDIN: I am having trouble hearing you.

FROM THE FLOOR: Closer to the mic? Better now okay, so, my question is, how do you like handle the situation when you have to choose between making responsive web app and like classic website where everything is rendered from the controller view template and stuff like that for this co purposes?

DAVID GOULDIN: Are you saying when you would choose which?

FROM THE FLOOR: Yes, Google robots don't have the capability for now to grow the web size that are rendered by java script, I think.

DAVID GOULDIN: They have (INAUDIBLE) I see the point in that, you are departing from web standards right and so, like you are going to lose some of that goodness that comes along with like really simple just like full responses in the original page, is that what you are talking about yes., I would point to isomorphic java script, get the goodness of

the client application, ... fully rendered server page, have things like push state to keep track of your urls's, the user fields like they are using a regular old website even though like java script is powering it on the client. Copy, paste ...

Shipped down fully in html to the client and then the client uses the Django API as a consumer, does that make sense?

FROM THE FLOOR: Thank you.

FROM THE FLOOR: Thanks for the talk, my question is, how would you go about like integration testing in an environment that has lots of services or you just have like, the unit test level things where you test like the service in isolation and you maybe use marks and how do you make sure that the mark for a certain service stays or has the same API as the service, it doesn't change over time, and you are testing something that is not there.

DAVID GOULDIN: This is edging in on the world of distributed computer, where service contracts are extremely important and they should be versioned and they should be enforced. I know a lot who use the Json schema to document the ..., as long as you have strict "contracts between one service and another., you can test a service in isolation, know that the integration will work, you always want to have integration tests, to run those in like a sand boxed environment probably want to use something like docker or ... or something like that, to spin up virtual machines so run your services in is way that has party with ... I know ... launched with what they call dev cloud, the 20 something services they will spin up that number of services in order to test all of the integration points, the more services you have, the more difficult it is, as long as you have strong contracts most of the time you don't need the integration level and you can mock out or stub out calls according to the contract established between the services.

FROM THE FLOOR: Thank you.

DANIELE PROCIDA: Thank you very much David.

(APPLAUSE).

1.4.4 Stefan Foulis: Local development with docker

Our next speaker is going the need internet access, we are having a few problems with connectivity today, so, unless you are doing something life or death right now on your phone or lap top, just close it for the next 20 minutes that will be greatly appreciated.

DANIELE PROCIDA: I am specially pleased to introduce Stefan Foulis from DVO and it is I had the privilege of having him for a colleague at DVO, so a great pleasure to work with him and pleased to have him here speaking at Djangocon.

(APPLAUSE).

STEFAN FOULIS: Thank you Daniele, so today I will be talking about local development and local Django development with docker, so I will briefly introduce what docker is and what the benefits are of using docker. I will look at some basic docker concepts and then we are going to create a dockerised Django product. Then use docker compose to define our stack, so run the stack locally, as a bonus, if we have time I will show you how to use nice urls's with a proxy and docker.

Most of this works without internet, I think just to make sure, I think we maybe sacrifice something to the demo gods, for that I have kittens (LAUGHTER).

So let's delete or sacrifice one or. (APPLAUSE).

Hopefully that is enough!

(APPLAUSE).

I still have some kittens just in case, so, let's see if we can do this.

First of all, I am going the just want to show you a scenario, so got a new project just checked out the repository and you want to get this up and running. You are go the docker directory, the docker composer comes up. Start all the containers needed for the application, we are starting cob stainers and the web application, then the familiar run server. So everything is running. I can switch over to my web browser, open this IP and there we go. That is the web app

already running, in this case, just language containers running in, that is for debugging purposes and already using the cache, 3 second cache and a longer cache, so we see every 3 seconds it is updated so show you how the cache works.

I want to guide you through the way this. I want to find my presentation again, dam kittens!

There we go.

So what is docker? To make it really simple kind of like a virtual machine. If you understand how to make virtual machines work to understand the analogy. It is efficient, run them in a host. Not the overheads, you can launch new containers within milliseconds, not in seconds and minutes like with a virtual machine. The main selling point is the set of tools around it to build and share your applications with others and we will have a look at that with docker images.

The party between development and production. Because you can really run the same code bit for code, the operating system, every library that is installed, the development and production.

So some of the common things you will hear, combination with docker, here is the image, this is you can think of a template. To go back to the virtual machine analogy, a virtual machine, packages then make a snapshot then use it as a basis for a bunch of other machines you run, those will be containers in the docker world.

Then there is the registry, a useful place where you can share and distribute those images, those reusable images.

Here is some of the operations you can do in docker, at the right hand side the machine, you can docker pull an image which you would get from external registry for example, or build locally with the help of the docker file, once you have it you can run it.

On docker hub, there is ready to use images, like we are using in this example. Django image in docker that is just insanely easy to use. So, if you, images work in layers and that is how it does this with these different images so, there is for example, a debian8 docker image and then the Python image. Then there is the Django image that builds on top and adds it stuff. At the end your app which adds whatever is specific to your application and of course your source code.

A docker file to run a Django app is just this one line. In this case it is using the own built trigger, automatically detect your ... copy your source code into the image, there is not more needed. To see an example of how docker files work, I will make an example of using the Python ... so this is a docker file, start off using the Python image, install some packages, we want the mysql client, couple of things, create a directory, make this the default directory, copy in our requirements, install them. Copy in the rest of the source code, tell docker we are expecting to expose part 8,000., requirements file is simple. Just Django, now if you want to start a project from scrap, use docker build to build this docker file which is created dash T defines the tag, the name the image is going to have. Then use start project which basically we do docker run, IT means run it interactivity, ... – V mounts your local source code into the container and user is needed for permission reasons for mounting that folder. Then we run the general admin start project and you are all set.

We can run the container now with docker run, the same thing, the only thing is we tell it to map the port 8,000 on our local machine, to the port 8,000 on the local container.

... default command for this imam, but listed it here for completeness.

Docker compose, a command line tool that adds a yml file which describes the whole stack, a combination of multiple docker images and build them together to a whole stack. So I will switch over to demo again and show you such a composed file.

So, here at the top you see it is, we all our main container we call web, we tell it to, if we want to build it, build in the current directory, add this to your project. We do the volumes like we defined before in the direct command as well. Mount the directory for our up loaded media, so we don't lose that when we recreate and destroy the container. Map the port again.

This is where it is interesting, link other containers into this one, take the container and map it in this container with the name postgres., this is straightforward, pulls it from the docker hub and you are good to go already, all we do is define the password. The database here, the host is postgres. ... Makes the host and matches that name to the IP address of the other container using postgres.

We have extracted the most important ones here, of course you will have to update your settings a little bit to follow this convention, so here, you see, we are setting secret key based on the environment and using some reusable tools like Django database and create settings out of it.

I got the docker file is a simple one, just the Django template we have built. So basically we are ready to try this out.

This is the app I started previously, I will kill it. So all containers are stopped again. Actually completely delete those containers. So this is now wiped postgres container, wiped the data, starting afresh.

This time, I am going the do compose up with no recreate and just a web container to show that it will automatically detect I need these other images because I need the links. Start them up, no recreate, tells it not to rebuild the containers if not needed otherwise we will lose the postgres data every time we start.

So we have our up and running again. You see, Django is already complaining oh, we forgot to migrate our migrations, this is pretty straightforward, so here is the command, docker compose run, which will with the web components tell it, well I want to run this in the context of the web component and run the command. I will run, now I will end up in this container and I can run multiple commands so it is a bit faster than running, than less typing than always typing the whole thing, so I can just Python, manage, migrate. Here.

There we go.

While we are at it, we will create a superuser.

Great. So, let's switch back to the browser. See if it is still working? Yes, we are still up. Now we can also log into admin, with our newly created user.

So, let's say we have this extra little application, myapp doesn't have any migrations, you go about this the same way, every time docker compose run, web, your commands or step into the container and just do them one by one, we will make the migrations for this app.

There we go. And we will go ahead and run them as well.

So, another thing that is a bit different than when you normally develop in your machine, when you need to add dependencies, in order for dependencies be picked up by all of your containers you are running, add to requirements and rebuild the whole container, a trick there is you can also just add them. If you have a single requirement and don't want to rebuild everything, you can use caching mechanism to your build. Install that one package you need for experimenting but before committing this to code, you should probably put it in requirements. Py.

Once you have rebuilt the container so the command to rebuild it with docker compose would be build then the name of the container. To be really sure that it actually gets the new image, it's a good idea to delete the container and recreate it. It's also good to do that when you change for example environment enables because it doesn't always pick those up automatically.

Another slightly different thing that you have to take care of if you want to use tools like PPB(?) with docker move up you won't have an interactive console it's just a log output but you can run data in another way that it's because to have an interactive console so run docker compose run instead of up and you tell it to service ports extracting which means also maps your port into a container like it would do when you normally do up and when you start the container like this you'll actually have an interactive console here and can do PDP debugging.

There is another neat way you can change stuff, for now every time we do docker compose run it creates a new isolated container that is not part of the web container already running but there is a command to step into, an existing container. And that is docker exec. So I can do docker exec IT for interactive, I choose a container, let's take our web container. Run bin bash. May be have to sacrifice some kittens. No don't want to kill more kittens - what you do here is step into the processing space of existing running container then if you do PS for example you see the same processes running that you see when running the run server.

Another neat thing is a package called - or docker image called N gin X proxy. We run this on our developer machines on port 80 and port 1 1 3 so does all the proxy for {inaudible} run locally and this is smart enough to watch docker and see any containers that get started up and if you have an container that has environment virtual set like this it will run any request for that virtual host to the correct container.

So what this gives you is that you can actually run your web-site like this which probably won't work now. So you can have real URLs for your web-site. This alleviates problems you might have with cookies when you run stuff on different ports, makes things way simpler in that regard. It also has the functionality of doing SS L determination so for our older stack we have SS determination for older development as well and that allows us to really test the different services how they interact much better than if we have just http.

Okay I think I am running out of time.

Let's go to questions. Thank you. {Applause}.

NEW SPEAKER: How do you keep track packages installing your dock images and -

NEW SPEAKER: Can't hear you?

NEW SPEAKER: How do you try stacker images make sure all your images are to date with all the latest secret batches?

STEFAN FOULIS: Updated images on docker hub and make sure you get those locally?

NEW SPEAKER: Yeah or that images on docker hub web are updated with latest open SSL or mercury or ability -

STEFAN FOULIS: Basically it's to you if there is a dependent image you are using that has a bug you have to rebuild your application based on that. There are tools on docker that make this easier so if you have docker hub build your image as well a service they have you can have build triggers that will automatically rebuild your image if one of your dependent images changes which means you always have up-to-date version in that cases you still have to remember to employ that to your live system but that alleviates it a bit.

NEW SPEAKER: Hi there. Why should I move from vagrant to docker as a sort of development environment?

STEFAN FOULIS: Well you think it's a good idea or?

NEW SPEAKER: No - I never heard of docker much and I just wondered.

STEFAN FOULIS: It's much more light weight specially if you have a lot services that are different, if you have one part it's only a no dap or Django app and have different dependencies what they need on the system. It's easier to spend up 10 docker systems with different operating systems than 10 vagrant systems with different operating systems.

NEW SPEAKER: Thank you.

NEW SPEAKER: Thank you very much. Do you think there is any benefit of using Ansible with docker?

STEFAN FOULIS: You mean use Ansible to build docker images or -

NEW SPEAKER: Yes -

STEFAN FOULIS: I don't think so because the nice thing about the docker files is basically you don't have to maintain your image over time because whenever there is any change you just rebuild the whole thing and you redeploy a whole new artefact so what Ansible is strong on I think is keeping servers up-to-date with what you've defined to be your stack over time which isn't really needed with docker any more so the docker file I think is perfectly enough to build those docker images.

DANIELE PROCIDA: 2 very quick ones.

NEW SPEAKER: First question. Do you use docker in production. If yes how do you manage to keep user data and put database on this containers if you need to rebuild them? If not why not?

STEFAN FOULIS: OK so we do use docker in production. We make sure toe that the containers are state less so any up loaded files for example on the sites me host go into a shared storage back hand. The postgres database is running in docker but not running in those containers, it's running separately as a service and we're using that and with postgres for example if you want to get your data outside of the container you mount the directory and have the data part outside the container so you reconstruct them.

NEW SPEAKER: If I may, how do you manage different versions if you have to all {inaudible} container NEW SPEAKER: SF of the what?

NEW SPEAKER: For example postgres data 3, and 4, have to change data of underlying files -

STEFAN FOULIS: It's the same process you do traditionally with a server so make sure you migrate the data when you upgrade.

NEW SPEAKER: OK thanks for your talk. Say you have your own project and have gone down a rabbit hole writing your 5,000 line bash script to build everything up and only just heard of docker, do you have any advice for converting your existing projects over to docker?

STEFAN FOULIS: Really depends if you're having problems with your current set up. If you're not having any specific problems for them I don't see a good reason there but of course it's useful for new developers starting on a project because the only dependency they need to install is docker so if you have a lot of new developers working on your project probably until they have that set up it will take a long time if you don't have a good reproducible set up.

DANIELE PROCIDA: Thank you very much Stephan. I won't keep you but I've got an important announcement before we go into our break. Let me just quickly do this. OK so I'm about to put through the order for the sprints catering on Thursday and Friday. According to this, we have got 248 people coming to lunch on Thursday. 199 on Friday. But only 137 are coming to the sprints party on the evening of Thursday in-between. So, this is why we need you to give us accurate information because we don't want to have a party expecting 137 people then have 248 turn up because some of you will go hungry and go home disappointed so what you need to do is go to your teeto ticket. You have one of these on every single email message that came to you from teeto practically and just answer the questions. Hit change to do that and say yes I will be coming to this meal or that meal. This will be on your individual or corporate or sponsor ticket the one for the whole event the first ticket you bought. If you can do that straight away we'd be very grateful.

I'll let you get your coffee now. Be back promptly at 5 past, an extra 5 minutes because we have a special announcement for you of something that's been kept secret until now.

(Break)

1.4.5 Xavier Dutreilh: Web accessibility is not an option

VINCE: Our next speaker is Xavier who started using Django 2 years ago {inaudible} opportunity to get involved with Django. Talking about web accessibility. {Applause}.

XAVIER DUTREILH: So, hi everybody. My name is Xavier Dutreilh; I am going to talk about web accessibility during the next 30 minutes. I would like to say 2 things before starting my talk. The first one is I really appreciate different action on hearing are doing for the conference because otherwise I would not be able to understand anything of almost all talks.

The second thing I would like to ask you is much more a favour than anything else. So at the end of the talk if you want to ask some questions, please, try to be clear and simple so I can read them here. I am very sure that some of you have some questions to ask me. Some of you have been very honest yesterday about issues in life. And I think some of you may have some questions about accessibility.

Accessibility or web accessibility, we can ask both questions.

So, first of all I would like to clarify why web accessibility is not an option.

We are here at the conference on Django. It's all related about the web. So, is anyone in this room ever considered life without the web? And all the smaller opportunities provided to you?

Before this talk I never really thought about this too so it's not surprising you may not have think about it.

But still even if you've never thought about what the web did for you and what happened in your life, it did, you may have obtained work, studied, you may have learned about it, maybe make friends, from what I see the Django community is very friendly so people make friends thanks to the web.

May be today you do not see any issue with the web, you use the web any day and it's fine and may be for you it's perfect but this issue exists for people and the issues are very real. So I make this talk to explain to you what are these problems and what you can do to solve them.

One of the biggest reason why some people have real problems using web-site is because we do not design people for people in general - we design web-sites for non-disabled people. So maybe you are shocked by such statement but it is what it is. We do not include everyone when we design and build web-site and exclude a large part of our community and the world in general.

In particular, we exclude people with disabilities. So, may be the world of disability is for you very fuzzy but with this talk I think you will get a clear idea of what it is.

So, let me tell you a few things about web accessibility.

So, it starts with an inclusive practice of removing barriers for people with disabilities. In fact, we know that many web-site we'll be on every day may be difficult to use if we have some kind of impairment. This issue, this kind of issue keep all people with disability apart, on the side and they cannot use it. And such practice is very important because if you have some or no disability you get access to information on the web-sites and functionality. It is not linked to the kind of disability you may have.

So, disability may be hard to understand if you never read something about it document yourself about or impairment or disability.

So disability is the consequence of an impairment. And you may have one since your birth or you can get some disability later.

So if you get one from birth you can do anything but sometimes when it is acquired later, it, you can acquire the disability because of an accident, of genetic condition that start later in your live and affects your abilities.

Also can be progressive so it means that not all impairments have the same effect on you when you are five years old or ten or 20. Sometimes in only a few weeks or a few months from being fully able to being disabled.

Often we split all this impairment in to four main categories we can see other categories but it is much more important to, to focus on this one to start.

So, visual impairment is what you see with your eyes, you can be blind, so have issues with light and perceiving, understanding what you see. Blindness is not as we may think when we don't know what it is, blindness is not when you do not see at all. It is when you cannot see in fact when you go to a doctor and you test your eyes and what you can see, if your view is rated below a certain threshold you are considered blind. Even if you can still see something.

Low vision, or also called poor vision is something that a lot of you have in some way, if you wear glasses. So a lot of this people in this room, never consider yourself being someone with disabilities? Now you know you are.

Colour blindness is when you have some trouble making differences between colours, it is not always about making a difference between red and green colours, it is also an issue with blue colours sometimes, completely different. It is not only about making difference between the red and green colour. Sometimes a few people have issues with colour and they have issues with red and green colours and blue colours.

Impairment maybe also hearing and most of the time it falls under two categories; deafness and hard of hearing which is the category I am following.

When it comes to motor, I think most of you imagine someone in a wheelchair and that cannot move or walk by yourself, it is much more general, people with genetic condition that makes them having a lot of trouble to move or to coordinate the movement following this category, so that is why we call a limited dexterity. It is also a loss of limbs when you have an accident and you lose your arm or two and you cannot use a computer as you used to before.

Cognitive. It is not visible at first but some of us have issues about learning things. For instance, remembering the things on websites. This is two possibilities, this is category is very large and they cannot learn everything.

From there what can we do to make accessibility?

The first thing is simple. Every time you put an image on the website try to describe it because for instance people who rely on screen readers to read a page, won't get any information. The screen readers, scans the image up and then nobody might be able to understand what it is on it.

So, this example is very bad. So, as you can see the name, it should be in front of me. So, if we try to add a description or may use something like image of Xavier, which is also bad, better to produce this example, we add in description but it is bad because it is a lot of, a lot of redundant information, in fact, one way to describe as, there is an image, there is an image of Xavier, if you never use the screen reader on the website, you may not know, but every time you add a lot of text it becomes longer for people to read the page.

Before this talk I tried to do a screen, voice over which are still on macro six ...

To give you an idea of how long, if you put too much text on the website, it can make difficult for people to follow you and to browse your site.

So this one example is short. It is clean but tells what it should be.

What I do on the photo is important. You have to describe it. Because otherwise it is, there is no possibility that these people, the people who use this screen reader can understand what is on it. So only if it is important you have to specify the action on the photo.

The second, to use html but in a good way, so you have to stick with standards and structure your information. So some of you write documentation on a daily basis, so they are used to a structure of writing but it is not always the case with the web. To give you an example, this is an example from a oversimplified on the website. You would use structure all your page as such. So if you have header in the page and some part of the navigation you would use appropriate types like header and not choose what a lot of people use in their developments or in fact have no semantic at all. So every time a screen reader it does not know if it is a header, navigation or the main content of the page. It makes the things a lot harder.

If you write at all, you may try to use some kind of example. You have a data which h1 a subtitle for h2 ... screen readers pay a lot overflow clone attention to this type of thing, every time on the list for instance they say, nine elements and element one and it is, element two reads the text and so on.

Advice and guideline, build proper tables, so often when we build tables, we put data without presentation to the name of the elements. As you can see here, for screen readers it is very important because it helps when it reads the table to make the distinction between the header and the column in the table.

The fourth advice is about making proper validation from validation and error recovery.

Most of you use Django and most likely only Django, so this issue is not less, not sorry presented in our web apps, much more prevalent when we do a lot of java script in the front end, as you can see here, what is important is that we have a field name input and a label which is, which explains what it is to field before that. The label is attached to the field by filling the attribute, the form attribute, so when you say name here it refers to the input with the right name.

At this level it helps a lot of assistive technologies to make a matching, allowing people to fill forms and it is also good practice to use all attributes you may have in html file because if the field is required and for some reason the person is not able to fill out the web browser will help the user to fill in properly and then instead if we do not make, if we do not fill the, we do not add the attribute, the user may submit the form and then get on without understanding what is, what.

Also this is very important, not also for accessibility but also for search engine optimisation, every time you put a link on any page, try to be sure it is clear. If you have these kind of thing where you name, you name this link as clear, click here, it is very problematic because a lot of people may not understand what it does. There are a lot of assistive technology which scan the page, extract all links and leads them to the user. If you have a lot of click here link in your page, once they are presented to the user, which is any, no links, no links which are named, click here will be understandable by the user.

So if you are, the link which moves the user to the article page, just name it article. But even better, something like see all articles. So the user can understands where you want to move him.

This is guideline is offering ... I think it is just confirms what it means, you should have to every time you put some kind of audio or visual media on the website, you should put transcript. Transcript is what is here on the screen, tells you what the speaker says and eventually some information you may have missed, if you have a hearing impairment like people when they laugh and when applause, caption is not about providing subtitles to the speaker and the subtitles are often displayed at the moment the speaker says the words.

So, I don't think you need a clear example since we have one from the very beginning of this conference, but if you want the slide of the talk or the video on-line, you may want to watch this talk from, this is one of the keynote and suddenly not so much talk from Py Con that contain subtitles, I hope following, the videos will contain everything.

After video and audio, you should ensure like I did for my slide to, if you are unsure, make sure that all of your binary files should be accessible, to do that, most of you probably use some kind of some kind of software like Office or ... to build binary files you just need to use the same guidelines, just need to structure everything, for titles and put things where it would be, it would make things a lot easier, but if you can use just the web for explaining yourself, use the web, because screen reader have made a lot of around the time with web pages not always the case that where all the binary files.

This one, again, it is very important for people who have a lot of issues, a lot of motor issues. Sometimes we have very complex pages with a lot of stuff and we want to help people to move between part of the page really easily. One example is from one link most of you never saw on any website but even though the, you have this like kind of link. This is, it is this one. In fact, at the very beginning of your document you have most of the time a non-visible link which allow you to skip to the main content of the page. So you do not have to scroll to the page to go to where you need to go to get access.

You can do this kind of quick link for everything if your page, for instance, you want to switch directly to each part.

This one is about not relying on colour to convey information. Most of the time since I started to work with back end developers they would use things like label to convey information but as with the classes imply, it is labels which should be boxed with a colour and not extend it. So if you have issues to see the colour, you may not know and understand what it means. A lot of e-mail clients a few years ago were doing that. So labelling all the e-mails, if you cannot see the colour, you don't know what categories.

Simple information, put some label, so if you have trouble to see colours you may, you just may put the text in place.

This one is probably hardest guidelines to apply. Because even if you master one language, if all of you, I assume all of you, as native language in which you can express yourself probably easily, but writing for the web is often very complicated. Sometimes to explain things we use a lot of complex things and we often, sometimes people have to understand what we call nonliteral communication. You have to be explicit about what you say and you will not have to expect people to have all the information we have to give you an easy example to understand this.

If you learn a new language and you talk to someone from the country that choose in language you may not understand the expression, you may not have the same culture in order to understand this.

Sometimes it is also good things if you cannot find someone around you or, I don't know, if you cannot find someone who has some with this issue, you maybe work with someone who is mastering the language, ask them for help.

One advice when you have to do this, you can also try to in fact a lot of this work, I don't know a lot of this word before making this talk, so you have to be explicit, clear, so if you think some of the talk you have heard in the past few days we are expressing and clear about what it is I want to tell you, it means a lot of people have spent a lot of time on it to make it clear. It should be clear and positive, it means that, write explicit clear and positive, there is two way to have sentences, for instance, if I write a positive sentences I would say, I eat at, I ate at the city museum yesterday, it was really good.

So instead of if I will not choose a positive form I would say, yesterday, the city museum has offered me to eat free.

So this one connective issues, much more about visual issues, the first thing is to use clear fonts. Some people for instance live with a dyslexia and much more the font we use on presentation website are not readable by some.

So, if you try to make your web-site accessible, you may want to of course build your pages with one set of fonts but also provide fall backs because there are fonts that have been made a long time for people with some issues with

reading, a font we use most of the time.

Also use relative units. If someone may have some issues accessing web-site need to zoom in most of the time if you use pixels to size everything when you zoom in it become a mess and when using relative units make sure when you zoom in on the page everything stay at the similar position.

And strong contrast, well you see the slide, you have to be sure the colour of the text is very different from the colour of the background and it's not magical and a lot of web-site have been made on this and if you need a tool to help you choose colours I will provide you with one at the end of the talk.

And this one is something that has been removed from HTML and it should of course - it's not about toning anything, it's much more about you don't want your friends if you develop a web-site - you don't want your friends to make some sort of epilepsy quizzes during the - but it happens. Some of you are smiling but, yes, for instance we often say that sweep flushes in one second may start epileptic crisis so you don't want your user to have some kind of episode on viewing your web-site and also avoid to move things on the page here because if people who have some issues with co-ordination when things they move will have hard time on the web-site and will not spend a lot of time reading things. But OK the thing was here, now it's here, and will spend much more time in other things.

This one is no simple thing to fix this issue. Java script is accessible to almost all web browser, most work just above screen browser so if you think Java script does not work with a web browser or screen reader especially with a screen reader you are wrong. 10 years ago it was not the case but today it is. I think 98 per cent in studies it has been shown 98 per cent of screen adjust users have Java script so you can make Java script application in top of your Django application.

The only thing you'll need to be careful is that not overlook people using keyboard over mouse. So if you have motor disabilities or if you cannot use your arms or do not have hands at all, you won't use a mouse and most of the time people when they built Java script encryption over {inaudible} for instance they will pint all {inaudible} over mouse and they have no click so may be your app won't work at all. You have to be careful about that much of the time the key board part is still supported so you can stick with key board.

And the last one, guidelines. It's very similar to the guidelines about tutoring content. If you use HTML 5 do not put HTML 4 element into HTML 5 because you'll have deprecated tax and it will be a complete mess. Avoid in line styles, put your access into complete files. Do not miss some attributes for example images, the other attribute is not an issue - {inaudible} it makes your pages very inaccessible.

So, from there, if you at least implement those guidelines on the web-site most of the people will be able to board the web-site you will just start to exclude a large part of our community and people in general. And after that I would like you to spend sometime to first document yourself about impairment and disability. If you're not very - I think for a lot of you it's not very clear what it means in reality so you have to document yourself. I can't talk about it during presentation but you have to document yourself. This subject is very, very broad so you have to spend some time.

You also have to talk with people who have some kind of disabilities. You cannot just talk to one person. You have to talk do lot to understand what it means. Because not only one way to live with disabilities and if you want to improve your web-site you have to talk them and listen to them. It's very frequent in our community to hear from people who have no disabilities and asking people with disability to making more effort include themselves.

They do they already do, so may be at the moment, maybe you should be.

Once you get a lot more fine and comfortable with the disabilities you should also read some guidelines like WC AG, but if you understood some of the things I said you are done with WCAG, the other ones the 2 other ones ATAG and UUAG are about software that allows you to write a page. And there we are come up last once you must ARIA and it's very useful if you build {inaudible} - {inaudible} WCAG is sufficient.

Also I think a lot of you haven't tried, haven't yet any kind of accessibility tools and most of you have a smart phone or computer at your disposal day or night so if you have some recent computer with windows or I don't know 1, 2, any other Linux application activate things like the screen reader the colour blind tools and everything else, it's very important that you try them, may be at some point it will disturb you because it will change how you look at web-site and how you use them to interact with them.

But you should have to try them because if you never try one you may not fully understand what it means to design accessibilities on your web-site.

To get things here a lot of people have started organisation like web aim and they build a web-site which can call your web-site and make you some things that tell you which things work and what things do not work so we can tell you if you use tone contrast and if you have issues with contrast it can tell you why and pose some solution. It applies to the same thing for the links and in fact most of the guidelines I gave earlier is covered by this application. So, just try it. It's wonderful.

And you can try it on the Django web-site.

Also if you don't like using web-site and prefer some kind of extension for your web browser you can use this accessibility tools. Google provide a lot of tools including this one to help you in this.

I don't know if that is something similar to fire fox but my view you should try this one with chrome.

And finally once you master everything or at least a bit of it you should provide your skills to any accessibility organisation that you may have met before or you may meet today because most of the time they like the technical skills need to make great application and you can provide them with these kind of skills.

So in conclusion, web accessibility is not an option. We may think so but it's not. And none of us are optional. So we should get back to work and fix this.

{Applause}.

VINCE: Thank you Xavier. If we have any questions could you make your way to the microphone please.

RUSSELL KEITH-MAGEE: Thank you Xavier. Just something you said at the end there. Did you say that the Django web-site does not have good accessibility at the moment?

XAVIER DUTREILH: In general if you use screen reader for instance not that bad, some areas for improvement and if you want we can talk about that after that but for screen reader it is really good. It is longer to pass and to read so maybe there is some areas for improvement there because some pages were really long, but I think, yes, on the colours, yes, the colours are really - you may not see the issue between the text in white and the green for instance but - so, maybe you do not need to make, to rebuild all the web-site in fact, you can keep your current CSS on the web-site and provide a link which - off CSS which applies some kind of stronger contrast. This is something we can do.

RUSSELL KEITH-MAGEE: Absolutely. If you are here for the sprints we'd love as much assistance as possible to improve any accessibility problem.

XAVIER DUTREILH: We should do it.

NEW SPEAKER: Fantastic talk so thank you very much. I just had a quick add on actually as a person with a visual impairment myself, I don't use any specific tools, but what I find sometimes is the most useful thing is for developers and designers to get out of my way. The best example I have of that is using meta tags to lock the zoom on the view port because that means that when you are trying to pinch and zoom on a web page in order to read text, that optional setting that designers like to set means that you can't zoom in any more. So I think there is a lot that you've said which is fantastic about using tools that are available, but also some of it is about not setting options which get in peoples way so I just had that to add but thank you very much.

XAVIER DUTREILH: Thank you.

NEW SPEAKER: Thanks for the very informative talk I was wondering could you recommend a command line that we could run like unit tests that would validate accessibility. Could you recommend one?

XAVIER DUTREILH: I do not know any kind of tool. I think may be there would be something to build - in fact maybe we can have some library built up on top of, - for instance if we want to make some Django forms more accessible and may be build something above the Django forms to apply accessibility more easily but at this moment, no, I do not know any kind of tools you can run from the common line -

NEW SPEAKER: May be a good sprint project? OK thanks.

NEW SPEAKER: Thank you for your talk. A quick question, from your experience do you think building 2 versions of the web-site one with accessibility and one for normal viewing is better than having the same like version of the web-site respond depending on the accessibility of the user accessing the web-site?

XAVIER DUTREILH: From my own experience and a lot of organisations - accessibility organisation - I think you should deal with the same version and should not put too much like just stick to what I said to Russell before I think one version in fact, yes, and not building 2 version because you want to provide the same kind of information, sometimes you may want to mark some information on the page as optional so it may be skipped by some for instance screen reader but no it is easier and from a dev viewpoint it is important not to have 2 version to maintain. But this one is acceptable because there is no value added to provide 2 version, 2 different version.

NEW SPEAKER: Thank you.

NEW SPEAKER: Thank you. I have inherited a design have you project and I have 2 questions. Where does the responsibility of the designer come in in designing and picking up proper colour scheme? How would you work in accessibility features in a legacy project or in a legacy code base?

XAVIER DUTREILH: So the designer is really responsible. In fact most of the time most designer I worked with in the past are not trained in accessibility so they use funny colours because they like contrasting of colours. There are a lot of tools like wave I suggested before that can help you to, well, you choose one colour or a set of colour that you like and you want to use for design and then after that you can just check that for instance if it's about contrast that the whole contrast are still varied. After that if your question about colours is about dealing with colour blindness, I think you should make your colours optional. In fact you can still use red or blue or whatever colour you want but it should not be important to use your web-site so I don't know what kind of computer you use or operating system you have on it but most operating systems allow you to switch to a grain mode to see how your design your web-site will look like: about the second question which -

NEW SPEAKER: Legacy code basis, working in it -

XAVIER DUTREILH: Oh yes so on new project it's very easy because you can just - at the moment you design your web-site, includes all this stuff and all guidelines. On existing code base it's very hard. I suggest you may be to start using one or 2 guidelines for instance describing non text content may be in {inaudible} for instance using retinue needs things like that may be hard because your designers not been designed for that. I would suggest to take one guideline at a time and apply to your web-site. It is really hard because most websites when they have not been designed for accessibility are hard to integrate.

FROM THE FLOOR: Thank you.

FROM THE FLOOR: Question on colour blindness, could you recommend specific tools for validating our presentation slides web based or pdf's, so validate them on colour blindness, on all the types that exist?

XAVIER DUTREILH: I mentioned a few slides ago. This is this slide, mention WAVE and mention colour contrast checker, wrote a few years ago and you can, you can use it in fact to validate your presentation. Before this presentations on some of my slide I was used to use some different colours like on ... because of this tools I make some adjustment to ensure that if some people in this room had some issues with colour, they wouldn't be able to read. So.

FROM THE FLOOR: I have just tried ...

XAVIER DUTREILH: If you have sometimes ...

FROM THE FLOOR: I think WAVE is really good if you have web based slides, what about pdf's and images?

XAVIER DUTREILH: If it is just about colour, if it is just about colour, you can use WAVE and because you can provide a colour, in fact there is a colours as specified in the tool I use for building my slide can be taken and injected into. Because you have fields to type, colour code you use and assert you can still validate. It is still a lot manual, you can have to take the colours from the tools and import to the tools but yes. It is yes, about colours it is very complicated.

FROM THE FLOOR: Thank you.

(APPLAUSE).

1.4.6 Theofanis Despoudis: Django aggregations demystified

NEW SPEAKER: Just a brief announcement we are going the take a group photo before lunch in the lecture theatre here, if you want to be on the photo, please stay here for a minute or two, if you don't want to be on the photo, you can go straight to lunch.

Thank you.

VINCE: Next speaker is Theofanis talking about aggregations.

THEOFANIS DESPOUDIS: Look it is the guy with the windows!

So, thanks for having me today. My name is Theo, I am a Greek man, I am living in Cyprus for now. Today we are going to talk about like one thing that, for you, for the developers is a set of tools for doing aggregation, and annotations on your data. So, a little bit of thing about me, I used to be the PHP developer, a pretty horrible pages developer I think a lot of people so you started.

I recently, I mean about six months ago starting programming with Python. I have this you know, sound that the snakes do ssss and I work at social airways which is a platform for managing your flights, you can set flights and you can have friends on the flight and you can also do a lot of things with your account and stuff.

So, we can, it is going to be a small talk, not a big one, we are going the talk about aggregations. Then we are going to continue with annotations and then they complement the aggregations notion and then we are going the maybe say a tips and tricks about this and then we are going the answer some questions.

So what are they? What are the aggregations? Basically, database functions, they are functions of the database. Django has a nice layer on top of it. You only have to deal with the effect you know. You just call the method aggregate or annotate. If your parameters are okay, it will return the result for you. So it is useful for producing summaries and collections of objects. You will have a look at collections of objects. It will produce something you say that you need to produce. So, we are going to see some examples later.

It is similar to the reduce function, you have a collection and then it applies the function and then it returns the results for you.

As a single volume. These are some of the functions, that is, those are all supported in Django. They are probably more in the database level but that is, the database limitations specific, some database support, some database don't support. But the average marks means, in this, on the right hand side is the return value. Return the float or the same field or any, or you know.

So we are going to just for this purpose of this demonstration, I have devised like a small database which is we have like this company so this company has like a mini games, on-line games. It has this you know, database schema, then we have gamers, the gamer may like have a gender or a name or whatever he has. We have a game, let's say the game is like on-line risk or on-line strategy game or whatever which is like strategy or an action, whatever we need to categorise it.

We have a game server, where the gamers register themselves. Might be the chance that the gamer registers in multiple servers that is why we have the field here.

The game station, records the games played. We have the game, when it is started, ended, linker to the server and then this mainly we save the result of the game. Then we make sure that the session and the users is unique, so we have this session, this user has won or lost and these, he has accumulated like some points. They might be a chance that we have like a game with like ten users, ten gamers? Maybe some of them have lost, some of them have won and they will just record the result.

Just for a simple minimal database for we need to work on.

Then say let's say okay, how do you use this? So let's say we need to count how many unique this is probably, how does it work? We generate the query that is we apply the function and then we return the result. It is 55 mate.

That is probably the yes. So let's say we need to filter the results so we can apply filtering on the query sets and any kind of filtering, all the tricks and the things you know. You can apply the filters about the user or find the user one or

whatever. Then we need to find the number of wins. So, we remove the you know, the losses and we need to count only the wins.

This is the actually, query in the database. So you want to do it by hand you probably do something like this. But the nice thing is, that Django for this functionality for you, you don't have to do this, the other parts. So left to use the API, the interface and then it will, it will help you build the query and look at the results. That is a bonus for Django.

Also you can do like, you also have a standard deviation and variance, they have a lot of functions so you can while you are there, just compute the average of the scores and the session, then you can also add multiple you know, multiple results so you can have the average standard variation, with the sample and the variance.

All comes one the single results for its function you asked for. So, let's continue with annotations. So annotations helps you also, it is another way to have a summary. This is a method that maps the differences, the aggregations aggregate, but these maps a summary value for each object. So, any query set equates it.

So let's say, want to find the sum of all scores by each user, so we need to go for each user and find all the collective summary of all of these scores. So how do we do this? We just annotate the sum and nothing would do here, we just the ... we find the score. So now, every scores, every gamer this score is a gamer object of course will have in addition a field game session result, because I didn't name it, it will give it to this name and then we will have this score, maybe the second will have another score. This really great tool to just helps you know, collect all of those results in your, in the same place.

So let's say, min, max, sum. Find the score of the session with the most scores accumulated. So what we can do here. We can annotate of course over the game session objects now. We calculate all the sum of each game session object across the relationship then we can aggregate and we can continue with aggregation and we can use the sum of course and apply the Max function on it. While we are there, you can also add all this game, access the free things because when we are over there, we can ask another thing.

The result is of course, the game with the maximum sum of scores, that is, it show it is result and the oldest game is like a daytime object.

Then what we can do is also slice and order our you know, annotations, so let's say you have, you want to ask what is the top five players by awarded points? So annotate the gamer object. You want to find his score, the sum of his score of all his games. We want to filter some because they will appear in our as a result. Then we need to apply ordering. Descending, so the first one will be the highest score, the second the next highest score etc. We also slice the first five.

Then we can see that you know, the first gamer is like this guy and his points are like in these extra field. Because we specified on the annotate you know, function that method.

So what else we can do? We can group that is a really cool thing because grouping is a really nice thing to do. You can use the values method of the related manager and if you do that it will just transform itself and it will group according to the fields specified. What I mean by that, see in the example the results can be used later on for annotated or aggregated queries and then we will be merged into the common groups. What I mean by that? So let's say we have like three servers. So we have, we can get the values and then we can annotate the count of the number of gamers. So the servers in the US, have users registered, the Asia and the Europe. If you want to group some of those results you could, you could make the reads the same for two values, that is probably the best way to do it but again for the demonstration, you could group the values for the Asia and the US and now if you go to the same, you can call the same method again it will merge the results so we have like six users from US and four from Asia, total ten from Asia, then the other one. So it is really, if you are into the sequel, you know what grouping means, groups the values into the common group parameter, so in practice you should always check the generator query, you can do this with, like from the Django tool bar or you can use the you know, the go I, you can easily see what is being generated over the request or you can simply print the query set query class, which holds the generated query.

So this links to another presentation, if you can do it in this sequel, try to do it in Python. If you have the option to do it in sequel, just do it in sequel. Apply the function where the data is you can do it as Python but, I mean, it doesn't look good sometimes.

It can be aggregated.

So what I, this also linkers to another yesterday you know, and to another presentation, how do we use a proxy? Maybe you can use it like this? Maybe collect all your queries related to this what you want to do, what you want to achieve, it may be applied different ordering, maybe you want to display another page with another statistics, so you can put the proxy class and apply different ordering. It maybe you can also create your own manager and then you can do all sorts of things as long as you stay with the boundaries of the proxy you know, proxy class.

So this is the last did for me, try to keep it simple. You can get easily into you need to think of what is your problem basically because you can easily get, you know, you cannot easily get away with a lot of things. You need to find what the problem is, what you are trying to achieve, how Django can help you solve that problem. For us, for developers really easy, because we have a vast library of tools to use and so we have to think about what we are doing every time. So, so all that the results, you need to think about if the result is actually worth it. I mean, can we count it? Cannot it be counted, I mean, what is the point? You have to think about what you are doing so any questions?

(APPLAUSE).

VINCE: First questions, come to the mic.

FROM THE FLOOR: Can you hear me? Thanks for the talk, so, you mentioned that we shouldn't do something in Python, if it can be done in SQL, I agree with that., we have a situation where we need to fetch the objects anyway? We need to list them somewhere. So is it better to do another SQL query, where we have to aggregate, or to use the same queries we fetch to do the?

THEOFANIS DESPOUDIS: If you have the results in hand you can use it. It is better if you reuse those capabilities so if you have like a function with, you have this aggregated function okay, if you use it one way, which will only one way, use it only once that is okay. But if you want to use it many times maybe later on in your code, you might be putting it in the function, maybe in the aggregated query, to reuse it, so you don't have to deal with the situation of how do I filter? How do I aggregate the result again? Om have to call the mb, only have to call the manager and it will take care of the results for you.

FROM THE FLOOR: Okay just another question, that is probably a question that is all over the internet, I didn't find the answer.

Can you filter the objects that you are annotating on?

THEOFANIS DESPOUDIS: I think so, you can, because it is a query set it is, you can filter the query set.

FROM THE FLOOR: I think it is an F object, it is not a query search. I mean if you want to annotate for example, only.

THEOFANIS DESPOUDIS: Because if you see the SQL generated it is a new field that gets applied so it is just like alliance, so you get this field back, so if it is like an F field you can, I think you can filter it.

FROM THE FLOOR: For example, your example if I want to see how many won games for each user was, not all the games, only the ones that you won?

THEOFANIS DESPOUDIS: If you want to find all the won games for each user, you have to count them first. Then put them in an annotated you know, field. So you have to annotate the count and find the, for each user, what is his number of wins. Or maybe if you think about what the problem S you need to find the count of the wins for each user.

FROM THE FLOOR: Okay thank you.

NEW SPEAKER: Hello thank you for your talk. I am wondering if we can use annotation with some simple mathematical operations like if you want to subscribe to 2 fields?

THEOFANIS DESPOUDIS: Well, that where it gets really complicated because you can do things but cannot over exaggerate over things because there are probably more ways to do mathematics in Python than SQL.

NEW SPEAKER: For instance if you want to calculate the distance of the object -

THEOFANIS DESPOUDIS: You can do it but it's really tricky and it's gets into the rabbit hole experience, you're trying to find something that it might be easier to solve with another method. So that's how you have to poor prioritise

and say OK what is the data, what do I have in my disposal, what tools do I have and what's the end result? So, you can try a lot of things but you may be - it may be not worth it sometimes so you may have to just use Python.

NEW SPEAKER: Of course thanks.

NEW SPEAKER: One very quick last question.

NEW SPEAKER: Direct response on that. I'm not completely sure it's 1.8 or 1.9 and there will be some statistic functions in Django {inaudible} that have been launched a couple of weeks ago, at least there is something in that area going to happen.

VINCE: Thank you very much. {Applause}.

1.4.7 Yulia Zozulya: Using Python to load-test web apps

DANIELE PROCIDA: While we're getting connected to let you know at lunch time lunch will be served both on that side and in the other dining room at that end so you can go to either one. Peter Finch's city walks are both now fully booked up for this evening and tomorrow evening, so if you've got a ticket you don't want please cancel it to give somebody else a chance. If you are going to the clink tonight or tomorrow night it's not really compatible with going on the city walk so cancel the walk for that night. OK.

VINCE: Before we start remind everybody about the photo that will be in this room, we'll do that just before lunch. Which brings me to introduce Yulia who will talk about load tests {applause}.

YULIA ZOZULYA: Hello everybody my name is Yulia and work for {inaudible} we develop tools for developers to make their lives better and the most recognisable in this community would be of course {inaudible} Python but we do not only make desktop tools. We also provide a bunch of services such as tracking system, bug tracking system or CI servers or code review tools and when we started our latest web servers which was review tool named outsource we decided we want to somehow re-think the way how we do the performance testing. And this thought brought us to a lot of questions for example the first one would always be what tool to use to do the performance testing?

There are a lot of tools out there from {inaudible} ones to encryption languages which provide a lot of frameworks for you and they can be configured via XML any files so vast majority of options are really hard to cover in one slide.

So if you actually select scription language to do performance testing you might want to understand why you do that and choosing Python over other scription language might not be a question in this community because you already use that language to the web server development.

And why to use scripting language at all? It's quite easy to load servers which require some complicated logic for example for the game in our services me even use sockets for the task and it's not easy with IO oriented tools or if you need complicated data over complicated requests over performance testing it's also easy just to write code tan to fill all those strange forms with a lot of fields and stuff.

Still, in Python there are a lot of tools which provide you such ability and I will cover only few of them and I would like to start with funk load here because it is the simplest one. As you can see it looks a lot like simple unit tests here with test set up and tear down method so you get code in easy recognisable matter and has special set up and cycle methods which I'll explain later. It configured with a in you like configuration file. It may scary because of a lot of parameters but it helps you to tune to your needs. And it can be easily reduced actually. So it is not that bad at all.

So what actually happens when you run crunch runner over your written tests? it actually spawns a number of configurate number of threads one by one with a little day so you don't get all your tests run in a single moment which almost never happens in real life and in each thread test is run in a loop. During configured of amount of time and this time starts only when the last thread had been spawned so you don't get into situations where you don't have all the concurrency that you configured.

When this has ended all the tests attempted to be created are killed so this should not be a problem.

Top cycle and down cycle methods I mentioned before, those are meant to be for configuration parameters that all the users have in common for example URL, web server or some other configuration you might want to pick up from your files.

Those cycles - sorry ... those cycles form a branch in which you can configure different amounts of concurrent threads and this allows you just to compare the performance of your web server over different amount of load.

So why do you actually use funk load?

It allows you to write your tests in easy manner just like your IT or unit tests and there is a runner provided that allows you to run this in single user mode so you can use them as smoke tests for example.

And other advantage would be that it has benchmark more, benchmark - it has a distributed mode and you can easily adapt for virtual glance to your load environment just by 2 lines in the configuration file.

Other thing about funk load is that it is extremely well documented. It has a lot of examples it has a lot of tips and tricks about performance and I really advise its recommendation.

And the last thing would be that it is written in a way that it can be easily extensible. In our company we are actually over written some methods of default bench runner so it would communicate with our CI server and provide the data to our default graphics and it was really easy actually. But why not to use it? The main reason would be it uses default Python threading module only and that's quite a problem because Python threads can use only one cord, Python internal architecture, and it might be OK with IO tasks but if there is at least something you do which is CPO which loads your CPU, you will have ... hmm... sorry, just a moment ...

There might be a problem with that and you won't get to use all the resources of your motive {inaudible} glands(?) and ..., and the other thing about funk load is default reports of this framework is not really good because there is only one type of graphics here and it is not configurable at all, you just get the timers that library think you should get and you are not able to influence it in any way.

And a lot of problems of - some of the problems of funk load library can be solved with multi-mechanise framework and the core here of this framework is transaction class basically an object which should have a run method defined. And it has configuration file in your like - just like the funk load library. It has a lot less parameters here so it looks more light weight than funk load. So it would be easier to start with it actually.

So, what happens when you run multi-mechanise runner over your written class. It loads your transaction into configured number of threads which is spawned one by one with a little delay and it seems a lot like funk load up to this point but the main difference that those threads can be formed into user groups and those user groups are spawned not in different processes using different processing model of Python. Its user group is started simultaneously and if you want - you may turn up the load with several user groups and configure amount of threads so you get load you need.

So why to use multi-mechanise, it uses multi-processing along with threading so you use multi-core with your clients easily and default report graphics is more configurable in multi-mechanise, just for example you can configure your own timers and it show you the results and graphics that you really want to see and actually it has more types of graphics - more than one - it is quite easy - and it looks better for my taste because it uses different graphics back end so it's generally better.

But why not to use it? For me API is not as obvious as simple unit testing because you are not get the idea from the start which helps object would be peak topped by the runner which are not and it's quite frustrating not to understand what's happening and the other draw back would be that distributors work flow of them is not supported out of the box and you will have to come up with the solution to that yourself, for example with some other Python module which you prefer.

And the last but not least would be rather fresh and actually developed framework called locust IO. The runner actually forms randomised locust which will fled your server and the rules of how they will be formed actually described in the related tasks set class which is basically set of tasks. The frequency of each task for each locust are defined with weight attributes and weight time between those tasks are also is not fixed but in some interval so it's getting really randomised here.

And what actually happens when you run locust scripts. The key difference here is that this runner doesn't use threads at all. It uses {inaudible} and greenlets and those are proven to be more efficient in IO tasks than threads and ... that would be it I guess. So while locust IO? Greenlets are much more faster with IO tasks than default p.threads and thing I forgot to mention locust runs light weight web, web server in the background saw you can share the results of testing

between your colleagues or even you can monitor what's happening in the real-time or you can change parameters of the load and it is really convenient.

The other thing would be that distributed work flow is supported is not that easy as in funk load but you just need to run slave for agent on your clients so it will, you will get distributed load here.

The other thing would be you'll get total heterogenic tests it's not like in other frameworks where you get all your tests run in a loop one by one. It is completely randomised and that's a good thing because you get something like real life experience here.

And last but not least it doesn't have any configuration files, all the configuration is done in Python code and there is not much to consider actually.

The only thing why not to use locust is it has rather tricky terminology and all this weird weighted randomised relations can be really hard to get into but once you get into you will see the bite brightness of it.

And all of the examples why not to use Python for performance testing and by Python I mean Python 2 here because all of the frameworks I've covered today can be run only under Python 2 even the fresh locust guys picked up Python 2 and if you already migrated to Python 3 I am sorry for you guys and the reason would be is level interpret to log which doesn't allow you to fully use all the resources of your load glands and you will have to end up with some solutions with multi-processing or adding clients to your load cluster.

In je(?) we ended up with using multi-mechanise along with jeneta(?) because we need a big load here and that is the reason why we abandoned Python because for the performance testing is not as perform itself.

So if you have any complaints about how the things are going in performance testing, please share them with me, I know that feeling, I've been there. Thank you for listening. Do you have any questions? {Applause}.

VINCE: Thank you very much. I see 4 people have gone up keep the questions short in interests of time.

NEW SPEAKER: Do you know of any of the tools you mention have some kind of functionality that allow you to see the synchronised impact on the server like the CP usage, memory usage and the server -

YULIA ZOZULYA: In Python no they don't but jeneta I mention that use Java it has this.

NEW SPEAKER: Thanks for your talk, I always struggle with understanding the output of my load test, I can run them compare them with previous runs and get a sense if it's faster or slower than previous ones but do you have any tips on how to get more information to that? More knowledge?

YULIA ZOZULYA: Um ... actually what knowledge do you mean?

NEW SPEAKER: For example I would like to know which URLs should I consider for improvement to make my service health better in general based on globe testing I guess that should be possible.

YULIA ZOZULYA: Actually if you see on the graphics that all the tools are or providing that some of the requests are not replying in the way you need them to, you can compare them between each other and actually a lot about impact on each URL as in locust 2 you have weight attribute for example which defines how much users will use this URL for example and if those weighted URLs are not performing really good probably you should look into them at first and only after that compare them with another for example. I guess that would be my answer.

NEW SPEAKER: Thank you very much.

VINCE: One last question I'm afraid in the interests of time.

NEW SPEAKER: .

YULIA ZOZULYA: I am here for the rest of the conference.

NEW SPEAKER: It is not really a question. Actually I'm more keen that modular cloud services team and we have the same problem each time we want to release a service and we built a tool that is called load and can spawn the test on many machines at the same time and it chooses statistics D to get some real-time statistics as well as CPU and memory and you write it is almost like funk load, you write a send are you how you want to test, what a user will do with your application and then it runs it a lot of time and it break everything. And we built second version using doc

S and now you can write your loader in any languages then spawn 2000 machine running your documentation at the same time and break everything again so if you are interested contact me.

VINCE: Thank you very much. Thank Yulia one more time. {Applause} various people pointing at each other. We're going to do the photo now.

DANIELE PROCIDA: So you'll direct us for the photo. You are the photographer. You give instructions. If you want to be in the photo stay here for a couple more minutes. It only takes 150th of a second to take a photo. The Django for social will be on Wednesday not today.

(Lunch)

1.4.8 Kat Stevens: The Full Stack Octopus

Welcome back from lunch everyone, just a few quick reminders. There will be a kind of gathering at about 2:00 o'clock during the lunch break tomorrow on one of the tables tomorrow, anybody interested in the Django formation for social good forum or platform or group or whatever we want to call it.

Speaking of social good, do you want to be responsible for making someone else go hungry? Or do you want to be responsible for the unnecessary slaughter of an animal?

So, I think not.

So, would you visit a web page if you could help avoid that? Okay.

Well you can.

All right. Because and you need to. Because this is our catering situation for the Thursday and Friday. Those numbers still do not make sense. Unless many people don't want to go to the party on Thursday evening? Which you should not miss because there will be a lovely barbecue with plenty of food. So, if you are coming to lunch on Thursday, staying for the sprints you must tell us, if you are coming to the party that night, tell us and the same for lunch on Friday.

This is how you do it, go to your e-mail and you find the message that is – hang on, – find the message that says your Djangocon ticket. Edit details, it will take you the your ticket and then you fill in the details for us, please do it right now.

Thanks to AOTV busy filming these days, put the videos on the internet for us, once they finish the task of editing for us, thank you for your team and your work (APPLAUSE).

If you are interested in taking up the opportunity to visit a counsellor from the wellbeing support team, the appointment notes are there on the board at the back, pick some up. There are some available for today and more available for Wednesday and Thursday.

Don't hesitate. Even if you want to sit down and say, I am not really sure why I am here? You might find that starts a conversation that leads somewhere. So a lot of people have gone in for counselling sessions not really sure why or not even sure what is bothering them but the fact they start talking is what helps.

I would like to welcome our next speaker, Kat Stevens so if you want to come up Kat and plug yourself in.

Kat is going to be talking about Django development in a small business, under the title the full stack octopus.

KAT STEVENS: Afternoon, I am Kat Stevens, the h web manager for, we are a fine wine and spirits merchant based in London,.

FROM THE FLOOR: Yes!

KAT STEVENS: Before, the question everyone asks me, is do I get to drink the wine, the answer is not as much as, the answer is not as much as I would like. I am here to tell you about being a full stack developer and some of the challenges and the advantages.

Here is my octopus, and, there are plenty of legs going on here. We have got all the Django side of things, on that side. We have got content we have got a design and user experience.

Admin and all the management and project planning and don't worry, our octopus hasn't come across a nasty accident, he is fine, we will come on to that leg at the end.

Okay, so how am I going the fit all of this into a twenty minute talk? Well, there is not as much code in this presentation as I would like but, that is the story of my life. [screen].

So, where do we start? When I was first joined the company, the company had did have an existing website and it was built in ".net" several years old, wasn't responsive this was 2010 didn't interact well with the companies CRM, so lots that needed improving for an expanding business, so I basically had a free slate to do whatever I wanted. My manager was very trusting and said like, if you can prove to me that you know, Django work then great. You know, you can go and ahead and do it. I decided to go with Django, it is something completely different, different to ".net," but different to what I had done before, never programmed in Python before, I had only recently come across Django because some of my friends were using it. I thought I would give it a go with a couple of small prototype projects my boss is keen on small events websites and things that would you know, branches of that business that were going the be temporary, this seemed perfect to try out Django, one of the prototypes I did was for Bordeaux enprimeur, every year around April, May, the wine industry goes crazy, everybody ships over to France for a couple of weeks and samples new wines out of the barrel. Now this is basically wine futures, you get people investing in wine that doesn't exist yet. Not going the be in a bottle for another two years, my challenge was to make a prototype site with a shopping cart and stock import for a product that didn't exist.

That meant I didn't have to worry about hooking up to any e-commerce, so it didn't need you know, a vast amount of security or a, it wasn't, we weren't dealing with purchases of fine wine that costs a thousand pounds, there was no money changing hands on here. Payment providers get nervous if you try and sell wine, products that don't exist. I think on-line selling regulations means you have to be able to return it within about six weeks and if it hasn't arrived yet then that is difficult.

It all worked pretty well. I managed to reuse some of the modules as part of the main site which is great.

Yes, I proved to the company that Django was something that we could be using and he was happy, I was happy. There was no one else to argue with, just me. So, really straightforward.

Okay. So on to the Django. This is the fun bit. My favourite part of the stack, this is why I am here. I mean I love all this stuff and you know, creating like a nice database model that actually reflects all the complexities of the wine industry, it can, it is very complex trust me.

Even getting down into the nub of the Python coding and fixing bugs, I love all this stuff. As you can see, I love writing stupid snippets of code. I love the open-source nature of it. So if something doesn't quite fit for my use, I can actually just, or even if I don't understand something, I can go in and look at the source code, that is something I could not do with.net, with Django, open up the file and say, great, here is a function, I can go in and see what it is doing.

Also custom template tags as well. This is something I have picked out, out of the huge amount of the things that I like about Django. Throughout the e-commerce site was having to use currency exchange rates because we sell wine all over the world and having to translate things into Hong Kong dollars, being able to use custom template tags and apply it to the exchange rates, it was so useful, adjust it to my needs.

I can lose track of things sometimes, I, it is a large complex site now and I really, it is really difficult. It is hard trying to keep track of things when it is just on your own, I, I do run into trouble and thankfully, the debug tool bar come to my rescue. One of the things about working on your own, you don't know that tools like this are out there half the time. There is no one to tell you, oh great, if you are having trouble with knowing what template you are using, why not use the debug tool bar, you can just put it into your settings file and that is fine. I didn't know it was there, until I was Googling trying to find out how do I know what template I am on? The debug tool bar, when I first started to use it, it was difficult because I was new to Python and Django and it kept crashing. There was some incompatibilities with some other plug ins I was using. I almost put it to one side and then forgot about it almost. Then again I came across a bit of code, a few months later. I am stuck. I can't understand why all these my queries are mounting up and up and I need to see what is going on under the hood. I remembered the debug tool bar, there was a update and it stopped crashing, now I use it all the time. Couldn't be without it.

Python code, I really get enthusiastic about writing a clean efficient views and queries. I also like cleaning all the things in my forms.

Being a sole developer, only have to merge code with myself, if I am coming across code I wrote six months ago, it almost feels like I am a different person, so I write jokes to myself, so I can read them six months later.

Libraries I want, I don't have to worry, I can try it if I don't like it, I can get rid of it. One thing I really did like was the tumbler, I know that the Django tutorial has a Blog and I did not use a Blog app in my website, even though we have a Blog. I used the tumbler API because it was easier for our wines, our sales people, out in France tasting from the barrel, yes, this one tastes good, I will take a picture and put it on the Blog, easier for them to do that, rather than log into the Django site that I had created and mess around with the admin. With tumbler, they could log in and up load the photo. It worked really well for us.

So, as well as being enthusiastic, I can get carried away. There is steam coming off the keyboard there. That means sometimes I can reinvent the wheel, as a sole developer then I don't know sometimes that these libraries are out there. Same with the debug tool bar, I just, there could be, I could be spending hours and hours on writing a piece of code. Someone has already done it. I never know. Yes, I don't even realise the wheel exists, I have to say, I didn't even know that this conference existed until one direction had to make a move! (LAUGHTER).

One of my friends tweeted saying, friends tweeted saying, check out what one direction have done now, they are up to their own tricks. Django! I work with Django! maybe I should go? Say thank you to Daniele and all the organising team who encouraged people like me first time speakers to come and speak here, it has been great. I really enjoyed myself.

So, finally with the on the Python side of things, I don't get any technical feedback, I have no idea whether I am programming well. My code could be amazing, could be awesome but I have got no idea, I know one of the talks on Sunday was about imposter syndrome, overcoming the feeling that you don't know what you are doing, that everyone else knows what they are doing a lot better than you do. I really don't know, I mean, not only do I work on my own at the moment. I have always developed on my own, never worked in a team of developers. It has been really a real challenge to actually try and be confident in my code and make sure that the company is confident in me and I am happy to deploy this code and make sure that our website is going to run okay. Yes, I think we have done all right so it has been a learning process.

So, I am going the quickly run through my advantages and disadvantages with testing and bug fixing. If I don't like something, I can ditch it or better yet, convince my boss we didn't need the feature in the first place.

There is no need for a complex ticketing system, people can shout across the office to me and say it is not working or better send me an e-mail. Everyone knows to report the bug to me. Things don't usually get messed because I can keep them in one place, I can decide whether something is important and how to prioritise it.

I can actually deploy small fixes quickly, if I know it is a typo in a template, I know it is not dependent on other things, I can get in there and do it in half an hour, I don't have to worry about the complex methods you are doing, when you have to make sure that everyone on the team is okay.

But, testing. I don't do enough of it. I am sure no one does enough of it but, we work in a high pressure company. Things need to be done yesterday and when I have to turn around a fix in twenty minutes that means that testing is going to suffer and it always does. I need it more than ever though, if I am trying to deploy something quickly, then it is going to you know, I need a robust testing system to make sure that you know, I can be confident in that deploy.

I keep, definitely don't have any of these in my code at all. Comments saying to do, fix this later.

I am sure none of you do either? But yes, moving swiftly on.

Thinking out loud as well. When I am trying to solve a bug, then I try to think out loud at people and try and put the problem into a sentence and see if that makes anymore sense to me, my poor nontechnical colleagues have been on the receiving end of this and nodding and smiling. Yes, I lost you at Django, sorry, but it helps me to even say it out loud.

But, sometimes I just get really stuck. So stuck overflow it is.

Obviously it is very frustrating when I am sure again, most of you have this problem where you type in the exact question and only it is there but no one has answered it. Or you find it is actually you that answer that question, that ask the question six months ago and still no one answered it.

So I will move on to another leg now. That was my Django bit mostly done.

In terms of design and user experience that is where my background really is. I have been designed websites since the late 90's, lots of marquee text, terrible under construction gif's, you get the idea, having the front end experience helped me in terms of saving time. I know what works and doesn't work in terms of html and J query and, so I didn't have to learn new things, obviously I did learn new things but didn't have to start from scratch there.

And one aspect of working for a small company, we really have a good idea of who our client base is. We know that well, they are fine wine lovers, usually male, over 40 and a large proportion in China and Hong Kong, so we know they are quite tech savvy, we don't have to support IE seven I was pleased when I learned that. I do have to support mobile responsiveness. All the time I was saving not having to worry about IE seven I put into making sure that it works on tablet and mobile.

So all those, trying to save time, on the design side of things. It is completely scuppered if you are having to use photo shop. If you have got your development server, men cached, four different browsers Skype, Spotify and photo shop running on one poor little lap top things start grinding to a halt. I have wasted more time rebooting to try and get photo shop to work, than I have fixing bugs I think. So the solution I found to this is to make friends with your Mac using print designer and be able to be able to send files home and hopefully they can sort it out. I am sure some of you are thinking, why use photo shop? I don't know, never used gimp, been using it for the last ten years and I know I can use it quickly and well, hopefully quickly, I know where all the commands are, I know what I can and can't do. Whereas gimp I would have to start again and learn those things, I don't have time to do it. It will be great to use all these nifty bits of software, but this is what I have got and this is the tool I am using, and everyone has to deal with it.

Okay so my other leg is content. There is no point in having an empty website. So, one of the main challenges I had to deal with, importing our stock data, all of our wines from our CRM built in MS windows sequel server, I was running my sequel and I had to support ETFA characters, wine names have circumflexes and accents in them. Translated into Chinese as well for our Asian market.

How did I fix it? I can't remember.

I don't know, I can't remember, something to do with OCDV drivers, I don't know, it was two years, I know where the file is I can't keep all the things in my head, in doing the full stack, I don't know. I mean I spend most of my time concentrating on Django, that is the bit I love most.

I do not enjoy dealing with ODBC drivers so I just write myself a note, I know where the note is. If I need to change it I have to look it up so this will remain a mystery to you for ever I'm afraid.

As well as the actual typing of data BTFA and things it's the actual wine data itself causes massive problems.

OK burgundy, any burgundy fans? if you are a burgundy fan you will be able to see that 3 of these are actually referred to the same wine but one of them costs half the price of the others. So who thinks it is A? Hands up if you think A is the different one the cheap one? OK how about B? Yes, some hands of course. C? Yes and more people there and D? OK most people going for D.

It's actually B because it doesn't have the Chaumes vineyard there, my knowledge was nowhere near enough good enough to deal with this information and I basically had to get some help.

So these dogs do not have burgundy in their barrels but basically I didn't have time to learn a vast amount of wine knowledge so I got some help, I got an assistant who had some wine knowledge. While they weren't very technical, they could use the Django CMS. So I used, customised the admin as much as I could to try and help them out.

Also with the Chinese translations, my Chinese is worse than my Welsh so I could outsource those and I made use of Django's i18n ...

NEW SPEAKER: Internationalisation.

KAT STEVENS: OK. As well as the fun stuff I have to do boring things, reporting, analytics, invoices, where are the customers coming from, is the site making a profit, is it paying for itself. I have to do all this custom management as well as super tedium stuff renewing domain names, liaising with ad agencies, terms and conditions, being on hold to Barclays trying to explain to them no we're not going to sell wine futures, you don't have to Murray about these 6 week on-line selling regulations, I had to sign a bit of paper promising I would not sell {inaudible} premier on free web-sites and SYS admin my least favourite aspect terrifying for me I would have outsourced it if I could have done.

It is something that fills me with horror because I just don't know enough to really be confident in what I'm doing here so I just have to do my best. In terms of security, well there is just me really so I don't have to worry about complex user permissions. And I could do a quick response when the heart bleed bug came up. I saw it come up on twitter and raced into fix it to try and figure out what was going wrong and then 24 hours later my hosting company sent me a nice email saying you should really fix heart bleed yeah. And a week later my boss comes racing through the door saying have you done anything about heart bleed? ! Yeah don't worry it's fine ...

Some problems are just beyond me and we are going to get some help try and outsource this and make sure everything is up-to-date and secure.

So our octopus sushi leg. Basically working in a small company, you know there is so much unpredictability that goes on, for example I've had to proof a 70 page cook book with matching luxury wines as a gift for our client base. I had a 6 month secondment where one of my colleagues left the company and I had to do email marketing for 6 month's that was fun. I am at the mercy of third parties especially when Barclays are hanging around for 6 months putting a block on our development time line. I also made an HTML 5 roulette wheel for my boss with wine names and things on for an event he wanted to do and unfortunately I don't think this is going to rotate very well - hang on is it going to go? No, that's it, it keeps landing on drunk so there we are {laughter}.

So to summarise, I have learned so much doing this project over the last 4 years or so especially how to research stuff how to look up things on stack overflow and make sure using the right name for everything and I discovered I really love Django and it's something I really want to do more of in the future. Definitely put jokes to yourself in your comment. I recommend that rather than song lyrics because it's what is this? I didn't remember at all. And if there is a problem I'm working on my own so I can reprioritise very quickly and easily.

However, things do get missed especially testing, it's definitely something I need to work on but it's always going to be difficult.

Holiday I haven't had a proper holiday in about 2 years. Generally being not on call but if the web-site does go down any time of day then I'm going to get a text message from somebody. And I've got no time to really get an in depth knowledge. I've got a broad overview but apart from Django I've been really trying hard to learn more about it, I don't consider myself an expert in anything.

Also I've got poor attention span, having to flit between all these aspects of my job so it's amazing I'm still here rather than wondered off out the door somewhere. Also the Barclays hold music is one of the worst things I've ever encountered! {Laughter}.

So I would like to give a big thanks to my friend Pippa who drew the octopus for me and she's at Pippa Alice Art. Anyone have any questions? {Applause}.

DANIELE PROCIDA: Thank you, that was absolutely brilliant and if I'm allowed first go at the questions I just want to say even down to the dates your career path seems to have mirrored mine yet you got away more lightly because I had to support 6 up until a few months ago at Cardiff university, that's not a question but thank you and I hope we see more of you in the community because we need people like you - any more questions?

NEW SPEAKER: That all sounded very familiar. Just one question, how do you make your boss less worried about you being run over by the proverbial bus.

KAT STEVENS: I have very good disaster recovery plan in place which involves hiring my identical twin. Yep ... {laughter}. No, it is a problem, and basically this is a problem I encounter right now because I'm going to leave the company and start a new job in a couple of weeks and they've been desperately trying to hire someone to replace me and instead they've actually decided to outsource the entire thing to an agency who's got a couple of members of the team who can take on the different roles.

NEW SPEAKER: 2 questions. One, you said you were working on a template tag that does dynamic pricing based on world markets and stuff and I'm wondering if you ever open source these kinds of fringe bits you write and could you put it on get up so the rest of us could check it out?

KAT STEVENS: I'd love to I'd have to run it by my legal team at the company to make sure it's not covered by -

NEW SPEAKER: Don't tell them just put it out!

KAT STEVENS: It's definitely something I'd look at because it was a lot of fun to do and would love to contribute something like that.

NEW SPEAKER: There was a couple of times in your talk where you explained something you had a tough time and I was like oh I know how she could work on that but you don't have your fitter up there so I can't tweet you.

KAT STEVENS: It's on the first slide, Kat Stevens.

NEW SPEAKER: Hi, thanks a lot for your talk a lot of that sounded familiar. I didn't have so much a question as a sort of a feedback on some of the things you touched on. Personally I'm in a very similar situation where it's very much a one person show and I'd really encourage you to invest interesting because that is sort of what's kept me sane for the past 2 years knowing I can roll-out something in production and it's not all going to explode in my face.

KAT STEVENS: Absolutely, I mean testing is something I've definitely done more and more as the years have gone, as the project has progressed just because the project has become more and more unruly and like I said I have a poor memory and I can't remember what affects everything else so having like a strong testing framework is essential.

NEW SPEAKER: Hi, that was a great talk. My question is your job sounds incredibly stressful and you said yourself that you are scared of the whole SYS admin thing. And I just wondered what was your background and what made you think when you applied for the job, oh, this is something I can actually do?

KAT STEVENS: Well I looked at the job spec and it said we're looking for a web designer for a wine company so I thought internet and booze those are my 2 favourite things {laughter} and I didn't really spec out what it was and I wasn't working at the time so I thought I'll go for it and see what it is and the company was really exciting and the people were really great so I thought well, it seems like a really good opportunity for me to explore this and be able to see, do the full stack and see what I enjoyed doing, find out what my strengths and weaknesses were. In terms of background I have a computer science degree but I stopped programming pretty much as soon after I finished it because I got disillusioned with the whole theoretical side of it and wanted to do something practical and doing web-sites was much more appealing to me than writing furrier transforms and things like that so as you are doing web design and have a chance to have a free rein to do whatever I want was a great opportunity. I want to say thanks to my boss for being so supportive for the last 4 years.

DANIELE PROCIDA: Thank you very much Kat. {Applause}.

1.4.9 Thomas Turner: Using Django in a desktop application

NEW SPEAKER: Hello everybody, like to introduce Tom from Joinerysoft; Joinerysoft is the UK's biggest supplier of joinery software and talking about using it in a Django application so hello. {applause}.

THOMAS TURNER: I did it the wrong way round ... other way round, start again, that's better. Right so it's a bit unique this talk. It's using Django in a desktop application. About me, I'm Thomas Turner, I own 2 - am part of 2 family companies. One is somcom which does technical web-sites, done stuff to do with people like NICEIC, which does all the electrical stuff and has done technical searching web sites. The other company is joinerysoft which does joinery software for joinery manufacturing or carpentry. I've been using Django since - for about 7 years since 0.96 I think, there is my twitter account and git hub account.

Using Django as a desktop application. Is it a good idea or a bad idea? Let's find out.

So, as I said, the programme that I'm talking about is joinery programme which is called JMS so if I call it JMS, I refer to the programme. So JMS, it is a desktop application but is running Django so let's find a bit more information.

What technologies do we use? Python and Django obviously otherwise I wouldn't be here. It's an MFC application so it's Microsoft foundation classes. We use a library called DHTMLx for all our grids and all the sort of tables and all our sort of framework. We use reportlab for all our printing. And we use fire Bird SQL database and we use CEF as our web browser which is basically Google chrome. And we use openGL and sikuli for testing.

Why did we choose Django for a desktop application? It makes it future proof. So, in the end we want to make the programme go on to the internet to actually go on to as a cloud service. It's faster writing - well can be faster

writing in Python than C++. HTML can look better than normal Microsoft dialogue boxes and becomes more network compatible and we did a similar thing before.

What problems did we have? Lots! One was securing Python and Django code. One was securing the HTML code. And one was finding a Django web server that could be embedded into C++. We also found there were too many programming languages. It's quite a nightmare that is. Finding a database engine that is simple to install. Our customers don't even know what a mouse is so they wouldn't know how to install postgres. It gets picked up by virus checkers because it's using port and it's harder to distribute tan the normal bog standard application and it's a nonstandard set up. And testing, we had a lot of problems with testing.

So, securing the Django and Python code. We put our own encryption into python. We took Python and found where the - so we took Python and put our own encryption into it. We had to build Python and so people could stop reverse engineering because we only distribute the PYC files not the PY files and of course you can still reverse engineer PYC files to get back to a PY file so we stopped that by putting our own encryption into Python.

We also had to modify Django to allow for the PYC because there is one bit that says asterix dot PY. It would be great if it said asterix dot PYC.

We also had problems securing the HTML code. The HTML code we didn't want our customers to be able to change it and do something wacky on the page so we actually put our own encryption I won't go through all the how we encrypted down the side. So we wrote our own template loader to do it first of all I didn't even know template loaders existed so I overrode all the template class but then we worked out that you could override the template loader.

So, finding a web server that could be embedded into C++. We tried Django development server - that's a really bad idea, at the time we were using Django 1.1 and that was a bad idea because it wasn't multi-threaded. It says not to do that.

We tried somebody's project, somebody's own C++ web server and that was too slow and too unreliable.

So, we settled on cherry py. Cherry py if you have used it, it is all written in Python, it is a very good web server, so we embed cherry py into our C++ app.

So finding a database that is easy to install. As I said earlier, our customers don't even know what a mouse is or some of them don't. So they wouldn't have known how to install postgres and postgres actually conflicts - because this is a windows application, it actually conflicts with other software. So, we found it's easier, so we couldn't really use postgres. And SQ light is not very good for networks. Our current version which is out there in the field, we use both of those products and we've had problems with both SQLite and postgres.

So the one we chose was firebird. Has anyone used firebird? Anyone know about it, put your hands up? A few of you but not many. So, it's easy to install. It's open source. It is not owned by oracle. And it works with Django using a third party plugin.

It actually came from a D base so there is the logo.

So, finding a - obviously we needed to put a web browser into the programme so we had to embed a whole web browser into our C++ application. We tried to embed Microsoft internet explorer and me we ran like a mile. It's a nightmare, we didn't want to go there. So we went for CEF. CEF is basically Google Chrome.

Why CEF? It's open source, it's cross platform, it's used in many big applications. I expect a lot of you have got these programmes installed or some of them installed: spotify, steamclient, github and Dreamweaver.

So what CEF is, it's a web browser, it's basically Google Chrome so yes it can be embedded thanks to somebody taking Google code and putting it into - making it so you can embed it into a C++ application.

So, another thing we had problems with was testing we couldn't get selenium to work because there was no web driver, we went for Sikuli, it works by image recognition, it is a bit of a nightmare, the problem is, if you do a star sheet change, you have to redo all the tests again. So that is basically why we had to use Sikuli.

Reports. We needed a way of generating our own pdf, we, a Django application will be running on a different thread to what the C plus plus is using, so we can't directly talk to a printer, so we had to go via a print preview. So we had

to generate a pdf but settled on reportLab it wasn't up to the job, so we had to modify it. It is sort of written in Python, not well coded it is actually written like a C plus plus act, one letter variables, so, that was another problem we had.

So, yes. It was obviously hard to understand the code. So I am going the show you a little demo of the programme now.

So this is right. I can't see that side. Let me go over here, forgotten not duplicating. Hopefully. I can work out which side it is. Right. So, that is the database there obviously so it is a single file database and I am just about to call the programme up here. Yes, at the top is all the files, so it doesn't really look like a Django application.

So back to call it up. So to a user, they wouldn't even know it is written in, they wouldn't even know it is a web application. So back to log in. Obviously that is Django.

Obviously this is all written in, this is all the html web pages and obviously these are all in Django as well. About to get something more interesting, so don't fall asleep yet.

So about to call up a window. So. All of this is written in, apart from this bit here, this bit over on the right hand side is actually written in C plus plus, so we had some difficulties getting information from Python to the C plus plus engine.

So back to come out. Again that is all written in Python, the costing. Back to print a report.

Obviously this is reportLab so we had to modify this so it worked for our requirements, this is the sort of thing that our customer send out to their customers, so that is it anyway for the thing.

Go back to that. Slide show from current slide. Not from the beginning.

Be there in a second.

Right. So was it a good idea? You tell me.

We are currently recruiting, do a small sale pitch, just in Bristol, if you want to get more involved, join our team, any questions?

(APPLAUSE).

DANIELE PROCIDA: Thank you very much Tom that was absolutely amazing. I don't think anyone here has seen anybody try to do something like that. Was really Django the best way to do this? Was there, surely (LAUGHTER), I mean it is.

THOMAS TURNER: I wouldn't do it again. No! I proved it could be done, I proved its been done.

DANIELE PROCIDA: I think that shows another round of I (APPLAUSE). (APPLAUSE).

I am slightly lost for words because that is honestly one of the most radically, different ways of using Django that I have ever seen, congratulations.

THOMAS TURNER: I was expecting them ...

FROM THE FLOOR: Excuse me, I am here mostly to disprove what Daniele just said about 2009 I think I did something similar. We wrote an outlook plug in actually which was presenting html generated by Django for very similar concerns of making the thing, network transparent with intention to later make it available from servers.

We also, we had a lot of the same experiences and just wanted to say that I share a lot of your ...

THOMAS TURNER: Thank you very much.

FROM THE FLOOR: Thanks for the great talk, different application how to use Django, I thought about using Django or in particular for application for desk top application, basically as a data source for (NAME - INAUDIBLE) would you recommend me to do it or not do it?

THOMAS TURNER: Just don't do this! just stay away seriously, just don't do desk top, it is too complex! I just would stick with a desk top, stick with something like C sharp if you are going to do desk top work.

FROM THE FLOOR: Hello, thanks for the talk. NFC and Django, that is kind of a weird combination, is there a reason why you didn't try to use something like GDK or something like that?

THOMAS TURNER: The reason we did it we wanted to go on to the internet. My other company did web applications, I prefer doing web applications than desk top applications, so I probably pushed more for doing this than we should have done it.

FROM THE FLOOR: Yes but for the desk top part why not use GDK or another Python solution?

THOMAS TURNER: Our current version is written in MSC, so it is an extension, it evolved to this. We didn't get there by taking a browser, we only had half the window as a browser at the start. That is why, we got there in iterative steps rather than one big sweep.

FROM THE FLOOR: Okay.

FROM THE FLOOR: Hi there, thanks for your talk, it was interesting. I have got two questions, one you have embedded Django in a C plus plus application, did it ever cross your mind to try and, I don't know how it is easy to do, embed the C plus plus into the Django web page?

THOMAS TURNER: We thought of doing that, our customers wouldn't want to call it up in a web browser, they would want to call it up as an application, we would have still needed an exe for them to call up. So that is why we have done it this way. We use a lot of open GL which is also rendering to the page. So, some of the views we use a, one of the main view uses a canvas which actually does all the open GL part.

FROM THE FLOOR: Yes and my second question was, you mentioned at one point your users don't really know what a mouse is. At the same time, ... to hide it away from them, worried about the users fiddling? Is this copyright...

THOMAS TURNER: The programme is protected with a dongle okay, so one of these dongles so, people would hack it but it is, it ranges from the type of users. Some of them don't even know what a mouse is, but some are large organisations which would have technical people yourselves in there which could reverse engineer.

FROM THE FLOOR: Thanks very much.

THOMAS TURNER: No problem.

FROM THE FLOOR: The question I wanted to ask was just ask before but I wanted to congratulated you for making windows, inside windows, inside windows (APPLAUSE).

I wanted to say we did a similar thing and we experienced many of the same problems you have some others with some other stuff we have a lot of hardware communication going on, I may do a lightening talk.

THOMAS TURNER: Those were a brief amount of the problems, I could carry on and on, seven years' worth of problems. So.

DANIELE PROCIDA: Thanks again very much Tom.

(APPLAUSE).

Go and visit Tom at his stand in the sponsor's hall if you would like a chat with him.

1.4.10 Shai Berger: Lispisms

DANIELE PROCIDA: ... I am very pleased to introduce Shai, Shai is one of my colleagues in the Django team and he will be talking about lispisms.

Thank you Shai.

SHAI BERGER: Okay. Hi there. I am going the talk about things that Python in general in Django in particular could take from lisp and how we could do this and also introducing here a library at a time I have been developing.

The first question that you could be asking is, why lisp? What is so special about lisp? What is interesting about it?

It is not a very new language. It is, it doesn't look very nice. It is hard to read if you are not used to weird things. Yet, it gets people very excited about it. You probably don't know Erik Raymond, like fifteen, twenty years ago, he was an important voice and an earlier adopter of Python.

What really is interesting. So, lisp actually pioneered many things that we take today for granted. Recursion, dynamic typing was first seen in lisp, garbage collection was first seen in lisp. So the question is, basically, are we done pulling things out of lisp or are there still things we could find interesting there?

Or, rather, did these things come first out of lisp just because it happened to be there first, or are there things in lisp that help foster innovation and allow us to and then give reason to their being more things that we should pick up.

People who know lisp usually when you say, let's take a feature from lisp to another language usually the thing that comes to mind is macros, lisp macros are powerful. People start adding similar features to Python, perhaps ... does that and it was rejected. There is libraries that does that, similar, sorry, does limited macros in Python.

But, actually that is not what I want to talk about here. I will have more about it if there is time later.

I want to talk about dynamically scoped variables.

So, what is the problem that dynamically scoped variables solve?

People everywhere for years have wanted access global access to request. Also to other of things it was just last weekend I think someone asked on the developers list for the current app to be available globally. Sorry. Whenever people make those requests on the developers list, it usually get the same answer: No. Globals are evil.

Why are globals evil?

Globals are evil because when you use globals in your code your piece of code is tied down to a specific system and you can't pull it out and reuse it. Because, when you are using globals and you call out to other code that other code can change it under your feet and it is hard to reason about the behaviour of your code.

All that is only, if you use thread locals, you can use those and things like thread safety to worry about.

Yet, evil is more than about being bad. You don't hear six hundred line functions being described as evil. Don't hear spaghetti code being defined as evil. Why are we saying globals are evil?

Because they are mighty convenient!

We like it that we can make ORM queries without passing the connections explicitly, we like to have access to settings whenever we like. In normal Python not related to Django, we like to have the STD out available by default for print and available not by default if we needed to press it.

So, globals are evil because it is hard to control them.

But what if you could have globals that do behave?

Globals were the values can only change down the call chain, so they can change from under your feet. Like a "with" statement where everybody change to the global would be like a "with" statement. Then each piece of code makes sense in isolation, you don't need to know what all your calls do to realise what your globals do.

You can set up local environments, so you can reuse codes that reuse the globals.

So, back to lisp. Lisp local change is the norm. Use let for assignments. What let does is not what assignment do in Python, but introduces a new binding, when ... disappears and then all bindings come back. After the inner let is finished, that is returned is the a from the external let and its values.

Equivalent in Python, you into deuce that shed all the old bindings, this is the equivalent Python to the previous lisp.

Let's keep this because it is not really important and we are short on time.

So, for Python, you still have the "with" statement in mind right? The thing is for Python it is not enough to introduce those bindings with "with," because Python likes objects and mutables with lisp with the examples we have seen before, all the values for atomic. With Python you can change parts of the variable without changing the variable. So, that would not be enough.

Then the obvious problem with the code, you give it a global, it changes its inner parts, you are back to where we were before. So, we need to make the objects immutable if we wanted to do this.

Slightly less obvious problem. Sometimes we do want it to change even in a restorable way we do want to change just part of the object and then changing bindings will again not be a good solution. So, we'll use something special for attribute change.

What we do and now I'm introducing the solution. Library called lispism which explores a special object called D and its attributes are the dynamically scoped variables and this way we have them marked specifically so you don't mistake them for normal variables. And for atomic values this looks exactly like the equivalent lisp. You do with D let and then you have the values for the variables. And then this prints first 1 when the value is 1, then 3 in the inner with and then goes back to 1. . And once you are out of the external with, you get a name error.

For containers, we do special frozen containers so after this, after the first let, you see that A has turned from a list to a topple which is immutable but the thing is they're deeply immutable, they're containers that also freeze their contents so if you access - so B is itself a topple but if you access its members in the originals they were lists.

And if I have a user defined type, with its own attributes, and its own - sorry some of them are just normal, some of them are containers themself, there are 2, one can only read and the other changes state of the object.

So, as long as I have tried to just read the object I get that I can access the attributes and that's OK.

If I try to modify an attribute I get a type error because it's a frozen object.

And attributes are controlled inside methods. I can call the reader and get the frozen attributes. If I call the rider I get a type error.

And I use ORM inspired syntax for changing specific attributes of objects.

Limitations of what I've done here. First of all, the whole freezing thing is a consenting adults API. That means that there are ways to go around it if you really want to. The intention behind it is to prevent values from being changed by mistake, not to force them to be frozen. That's one way you could just change attributes on a dynamically scoped variable without using let.

This a good thing because if anyone actually went to the trouble of writing something like this I suppose they know what they're doing.

And this is not as I said, not designed with security in mind, just with engineering.

I can't control extension models. If anything is written in seasonally it's not really controlled by the pattern library that I wrote.

I can and do try to make sure that values returned from extension methods even though they - sorry, if - dynamically scoped variable holds an object with extension methods and they return objects I make sure those are frozen but I can't enforce that they don't change the variable.

There is this annoying quality of generators that if you write a generator using a dynamically coped variable you change with let and under that you do yield and the variable is changed so that doesn't work yet. I have some things to do about it in mind but not done yet.

Dict access is currently a problem but again this will be handled.

And the future directions to go with this. The library is currently public, you can't pip install it yet. The repository is public. And the next places I'm going to go is, 1, handling threading, if you have a set of dynamically scoped variables and I open a thread they should be inherited but currently they just use thread local storage so they are not inherited by new threads.

Macro py the library I alluded to earlier, I could use that to make better sub object control. And it intend to do that. I could do things like here, to do things like, for example, for I can access not just attribute access.

And another thing that is interesting and dynamically scoped variables enable is something called restarts which is an error mechanism which is in lisp, has not been adapted by any other language I'm aware of and I want to talk about restarts a little.

The idea behind restarts is that when you try to handle errors you're basically in a conflict. The code protecting the error may be in the best position to do something about it, but it is not in the best position to make decisions. Code

higher up knows the big picture, knows what needs to be done, but has no access to details to actually do it. So, when you try to catch an exception higher up most of the thing that caused the exceptions to be strong, to be raised are already gone because the stack was unwound.

So, the idea here is basically that the inner code instead of actually defining what to do defines options, it defines the different restart and each restart - the example is reading a set of a log file or a set of rows trying to pass each row, and recover when things go wrong. And there are 3 options for recovery. You could return a special record indicating error, you could return a default record saying this is OK, and you could drop in a debugger. But the decision of what is the right choice is not made my read rows by the function at the top, it's made by the function at the bottom which is the one calling things, and test parsers defines how to handle each of the errors that could come about, it maps exceptions to selection of restarts.

So, with the setting as it is here, if a record shows and the parser decides it's out of date then it returns an error record, but if it calls a parsing error it drops into the bug because it's still in development what we wanted at the moment.

And another important point about it is that the function in the middle, read file, knows nothing of this. It just goes over rows and return - and read each row and collects - sorry just the function just holds opening the file, it doesn't need to know anything about the specifics of - neither of policy nor of details of error correction. And that is only enabled by dynamically scope variables.

This is like research for the future project. This is like - of course not valid python code just pseudo code.

There is a great paper explaining this in detail how it works in lisp not in Python.

So, that's pretty much what I have to say today. Library will be released on PY PI soon but for now it's just a public repository. I think greenspun's 10th rule of programming does not apply as strongly to Python programmes but it does in some degree.

That's all I have. {Applause}.

DANIELE PROCIDA: Thank you very much. We've got time for 2 quick questions.

NEW SPEAKER: Thanks it looks awesome and clever and dangerous I don't know. I was wondering, is there any library that provides immutable that has structures for Python?

SHAI BERGER: Not that I'm aware of.

NEW SPEAKER: And have you heard about HY?

SHAI BERGER: Implementation in Python yes.

NEW SPEAKER: Thanks.

NEW SPEAKER: Thanks that was great so the question is yes have you used it for anything apart from like hey this will be cool? Is it in use for like a real production system for doing that and what is that and why was that a good idea?

SHAI BERGER: Well, this is not yet in production for anything. This is just something I thought would be cool at this point. But, thought it would be call cool is the applications I have in mind for it are in Django. I think doing this sort of making things globally available because most of the things that people want globally available they want usually for read only access -

NEW SPEAKER: Request object -

SHAI BERGER: Like the request object. People want mostly read only access so we could give it to them with acceptable guarantees that they're not going to make too much trouble. And I think that the thing with restarts would be very valuable for error ending an URL(?) for example.

1.4.11 Markus Holtermann: Forms are static - no, they aren't

DANIELE PROCIDA: Thank you very much. {Applause} I am really pleased to introduce Markus Holtermann who is another colleague in the Django core team. There are tens of people in this room with whom I've shared meals and drinks and quite a few with whom I've shared a hotel room or someone's spare room, but there is only one person with

whom I've actually shared a bed. {Laughter}. And so that's what we do {laughter} - I was going to finish my sentence and say that's what we in the Django core team do for you when we go to sprints! Thank you Markus. {Applause}.

MARKUS HOLTERMANN: Hi, thank you for being here, thank you for having me here. Today I want to talk to you about how you probably use forms and how you could actually use them in a quite different way so might be different, might be similar way. And the way forms can, what forms can actually do beyond what most of you use them for.

But first let me introduce you to - let me introduce myself. I'm Markus Holtermann a master of science student at technical university of Berlin and I'm also a member of the Django core team since the beginning of this year and some of you might have seen me running around at euro Python 2014 fixing internet and Wi-Fi and being pretty busy the first days but conference was cool so I was relaxed the next days.

So, referring back to the title of my talk, forms are static, no they're not. I'm actually talking about the stuff in Django dot forms just to be clear. Before I go into any details I'm going to talk to you about what forms are and what basically you use them in general for and the most important use of forms is basically input validation then you have a user that goes to your web-site, enters some data and you want the data the user enters to be somehow validated against some rules and Django does a pretty fine job at that.

Some of you might not use it for end user direct user interaction but also to validate data that comes from API similar to what serialisers do in Django framework but yeah they are basically input validation.

The other thing is forms are static. But when I talk about static, what does it even mean?

Well, first and foremost forms do not change when you run your web application. More precisely, the {inaudible} layout does not really change. So what does change when I talk about forms? Well, actually the content user types into forms but, well, that's just per request and that doesn't really make any difference to the forms, it's just some data is validated.

So, the question arises why are forms static at all?

And static I define as you have to actually change code, you have to test and modify code, you have to probably run some testing staging environments and test your bundle release and deploy it and, well, this is pretty much the way how forms are used and how Django is a good idea how to use them, but sometimes you want forms to be more dynamic in terms of I have for example a user who whose locked in and I don't want him to have a catch up field he has to fill out when requesting data but on the other hand you want the page or form for anonymous to post data as well and there you have to have a mechanism for be a to prevent {inaudible} from spanning your site so there you have some kind of dynamic form, form feed allocation, the allocation you meant remove forms from fears from the form or you add them depending on what you actually do.

But what also changes in those kinds of situations is what Ola talked about in her keynote about - in the {inaudible} talk - you can have permissions on objects and you want to do filtering on objects you have in model choice spheres and depending on what's in your data basis you have different data for {inaudible} data in the fields which is pretty much obvious.

So I honestly have to disappoint you at this point because those forms are still static because you still have to modify code if you want to make them behave slightly different. You still need to test them on the {inaudible} and deploy them.

So, you might argue now that this is actually the right way to do and yes, I pretty much agree with you. Yes, I think almost all cases but, they are situations where it does not apply.

what are nonstatic forms then, they are forms that can change the lay out, the field that is are there. The fields that are not there, the order of the fields as well and practically, this, the dynamic forms are forms that don't need to be modified on a code level. Therefore, are without necessity to deploy them again.

Well, now you might ask, why do you need them? This is actually a brief story I have had a very, a company I have worked for, for a couple of years and we had a rather small client not that, not that much, not a large, they didn't have that large budget and the only thing they pretty much did for organising events throughout the year and the event organisation or event registrations was more or less the same each time but they all of those found forms had different additional fields they had to provide. Be it a birthday, e-mail address or whatever. So we would have ended up with

coding every week, another form for the entire year which is not really code base you want to maintain after some time.

So, I came up with okay, let's do something else. Let's do, let's be smarter at that. What I later realised is that the application I came up with allows you to do even way more complex things. Before I actually talk about the application itself I want to talk about you, what I am doing there at all. In the end, I create classes dynamically. So, what of you used or are uses model forms?

Yes. Pretty much everybody. Which is what I actually expected. But, model forms are not that quite static. This is actually a pretty simplified version of what Django does when creating a model form. I hope you can read it. Can you see my laser pointer in the end? In the back? You Django first takes all the model fields from the model, creates the class matter, you design on the form, and then sets the model on this matter class and creates the actual class, we have dynamic class construction, which something that Python lets you do.

Then Django lets you create the model form putting the class on there, and you have the model form class you can use that is in somehow from the form and pretty much lets you save in model instance.

But, wait. They are not really static, they are not real, they are static, they are not really dynamic, why? Because the form that is constructed is constructed from the models from the models fields and for the sake of your health I hope nobody of you actually modifies models during the run time of the application. It is, might work but you probably don't want to do that. So, this forms this model forms are still static because if you want change, you actually have to deploy the code of the modification you do to the models and then release and deploy.

Which brings me basically back to the idea of the, to the function that is Django users in this code type. Which is a Python bid in that let's you create classes dynamically, so takes the name of the class, created the base classes it should inherit from the attributes so Django is a framework and pretty much what you do with web frameworks and Django, is always interact with the database, this is pretty much what I did. I put all the information I need for form into the database. So, I have a form instance that is an entry in a database table that has some attributes and I have couple of fields that belong to a specific form and they are at rows or entry in a different table but belong to the field and when you open a view, or when you open the actual view, this application Django takes the database and creates the form dynamically, you don't have any culture changed there.

This is the basic lay out of the application, which I explained. You have the form model which gives you the instance of the form or the form class, it has a name, it has actions and I am going to, to that in a second.

It has urls where the form should be able on and where it should redirect you.

The other thing our form fields which basically wrap the Django form fields into usable interface for this application. They are instances you can bind them to a specific form and then the, the app constructs the forms with those fields.

Talking about the actions, this is the layout of the function base idea of the, that is going the be in the next release, you are pretty much get the form model which is the instance of the form in the, from the database. You get the form that is the form, the instance of the form and you get to the current request. With that, you can actually go and create model, you can actually do and create basically any kind of interaction you want to do with the data the users submitted.

You can interact with the form instance, take the data, the users submitted and do whatever you want with it. In this case, I am sending the data to the administrators of your page, this admin settings.

There are two actions which let you send e-mails to the specifiable e-mail address, or list of e-mail addresses, the presentation to store the information in the database. Right now it is just text field that wraps, that is contains some Json and uses that. Then the next release, the Json field couple of days ago this is going to be Json field for 1.9 so you can later query on the data inside the Json field which is pretty cool. Well the action in the next release gets also the request, which allows you to access the current user in the form, so you can actually create a profile form with dynamic form fields and when and let the user modify their profile with this application. If you want to have a new field you just add it in the admin or the, yourself this field will show up in the user profile.

The other form are the form fields, these are the wrapper, you decorate them, to the application again, they have to inherit from this base dynamic form field and of course needs to read, imported at some point which basically do that in your ep config, it defines which form field it should bundle or relate to. The name how it should show up in select fields and some attributes you want to define. So the type, if ... the field that is used in the admin after all.

Well, and this is basically how it looks like. You have a form that has the name, it has two urls, the urls where you submit it to where it is shown on and submitted to. The success url define in the generic class based views. You have the actions which appear, you have which I added a few versions ago.

Different templates for different forms you will see in a second, one default template for a form, html without any (INAUDIBLE).

There is an option to allow the user to visit a previously entered data item and with that, they can well, review what they entered before. There is an option to for recipients, so if you want to send the form by mail it goes there.

These are pretty much the fields you can enter and just the admin inline, not that much magic there. The attributes you find here show up here and this is basically what it looks like, there is a form that is rendered when you access the url, there is the success page that is, that you get shown when the user, when the user gets shown when you successfully submitted something and this is actually the detail of you when you visit this enter again.

There. This is the application I have developed as part of those, this client project. The original code was I created during a train ride throughout Germany which was, which is, which was pretty much work running in production for a couple of months until I decided okay, this is something I want to release as a third party app. Might be helpful for others, yes, that is pretty much it thank you. (APPLAUSE).

NEW SPEAKER: Thank you, got time for maybe two questions or so, if anyone wants to come up? One question. One of the things if you have, when you are on dynamic forms much of the data series is common throughout all the different forms, the dynamic part is where you would like to have extra fields to add in. Would there be any way to keep your, the same tables for the static data but the common data and then use the dynamic storage to store it into extra tables somewhere. Otherwise if you store all your data in Json it is difficult to query. Particularly if it is common data.

MARKUS HOLTERMANN: I am not sure I understood. The forms inherited from a normal form class, could override the, class based view that is shown up here in the first screen shot? So you could hook up in there and define your custom form it should inherit it from.

The other thing of starting a database, since you have the instance of a user for example, in this particular case, you could actually update user item based on some investigation so you verify that the users, verify the idea and then update the, update the, in respect of the user profile item for example.

FROM THE FLOOR: Hi, how do you manage internationalisation of ... name.

MARKUS HOLTERMANN: I don't. This is pretty much mono content translation which is something I briefly mentioned in the and hash Django thing, you could probably come up with something like, with an idea of Django parlour or something?

It was never used for the cases where I used the application.

FROM THE FLOOR: Okay thank you.

RUSSELL KEITH-MAGEE: Okay, so that all hinges on some Meta programming and sort of self-reflective code, code that is reflective on itself to build the forms. Given that that is a feature that one of the reasons why we formalise the Meta interface in 1.8, how difficult that is to try and let everyone else know, how easy or difficult it is to build something that is reflective or self-reflective?

MARKUS HOLTERMANN: I don't use the Meta ...

RUSSELL KEITH-MAGEE: I know you are not using it in that case, but the same sort of tricks would be involved? So the Meta programming idea you were using?

MARKUS HOLTERMANN: So the actually the most creative stuff and I am not really proud of it honestly is the generation of those option fields here. It involves some really hacky Django admin things, if anyone else comes up with a better code, I am happy to ... the other one is straightforward I think.

NEW SPEAKER: Thank you very much.

(APPLAUSE).

1.4.12 Aaron Bassett: Effortless real time apps in Django

DANIELE PROCIDA: We are going for a coffee break in a moment, so I will keep you briefly. We are running about 15 minutes behind our schedule, if you can come back from coffee 10 minutes earlier, not half an hour, but in 20 minutes that will help rescue some time for our lightening talks. Briefly the numbers are improving, keep them going. As soon as Thursday evening becomes equal or greater than the numbers for lunch on Friday then I will stop bothering you. If you are coming out to dinner tonight, if you have a ticket for the VFS that is at 8:00 o'clock, if you are going on the walk, you can do that if you are going the VFS, tickets for the VFS are still available from the website, need to be at the Clink by 7:30 if you are doing there, there are no more tickets available for that thanks.

(BREAK).

NEW SPEAKER: Can I introduce Aaron.

AARON BASSETT: This place feels a lot fuller when you are standing up here! I don't remember there being this many people in the hall when I was down that side. Yes I am Aaron, a freelance developer based in Glasgow been doing some sort of web development the last 15 years started in Django version 1.0 and been doing that more or less exclusively for the last 5 or 6 years, I run a small studio small is in me based out of Glasgow and as you might be able to tell I'm not actually Scottish myself I am Irish I have lived in Glasgow long enough however that my accent is somewhat muddled. I really don't envy the transcribers today sorry about this but I will try and make sure my slides give some context to what I'm trying to say so if you can't understand me you should be able to understand those. Throughout my talk I will be posting code, links, URLs, don't worry if you miss them I will tweet them after wards.

Before we get into real-time apps today I want to do a little bit of the history in real-time and how we started off and how we've progressed to what me now know is real-time applications.

Have to go back to 1996. 1996 was the introduction of internet explorer 3, it was the first web browser to have I frames, and I frames was really the first time that we could up-date a part of our HTML document without refreshing the entire page. In 1997 another Microsoft team this time the outlet web access team introduced what would go on to become HTML request object and that allowed us then to start doing real-time in a better way than what we were doing with I frames, it wasn't until 2004 with Gmail that we started to see these one page applications. But they were still using things like short polling. So short polling is where you issue an Ajax request to your server and issues lots of them so in this example we're issuing a request once every second so it's a lot of requests from our client to our server and it's a lot of wasted requests as well because we're issuing requests regardless whether the server has information or not, we've no way of knowing if there is anything on the server that's been updated since our last request, we just issued rue guest and see what's there. Then we brought in long polling and long polling was slightly different instead of having lots of very short requests we'd issue one long request with a very long time out. This example I think uses about a minute. And that request would stay open to the server until the server had some information for the browser, it would then send that information via that open connection, we'd close the connection and start a band new one over again and keep that connection open in the and that was slightly better because it meant then we weren't issuing lots and lots of requests whenever the server didn't have any new information to give us but both of these messages they're dirty they're hacks. We're trying to shoe horn in real-time functionality on top of a protocol that wasn't designed for it. In 2010 web sockets came along and web sockets it's what allows us to do effortless real-time apps. It's been specifically designed to solve this kind of problem and the major issue it overcomes if we use short polling as an example, every time we issue that request to the server we have to send our request headers and request headers are according to one of {inaudible} white papers are roughly 700 to 800 bytes. They're not huge but it's one standing on every request half second or whatever the frequency we set our short polling app that doesn't need to be sent, for the majority of these requests we don't really need to be sending these headers along with it web sockets it's slightly different, instead of full request header we have only 2 byte header with one byte that marks the beginning of the data and one byte that marks the end and that's it so it's quite a substantial reduction in the amount of header size.

I know 700 bytes doesn't seem like a lot any way but if we blow it up to web scale so say we have 100,000 users and these 100,000 users are using short polling so they're heading our server every second. And along with that request they send a header that's 871 bytes. So 871 bytes times 100,000 users per second 869 million 800 bytes per second a little over 665 megabits a second - I get bits and bytes mixed up - roughly 83 megabytes that is quite a lot of data we're sending really we have no use for, it's not containing any useful information, it's not new data pushed out to the client, it's just the same boring plate going back and for all the time. If we did the same with web sockets so had

100,000 users let's say the client has the information once a second we still have the same frequency of updates, still sending information every second, headers 2 bytes that's 1600(?) bytes but {inaudible} 0.2 meg. Immediately saving ourselves over 80 mega-bytes per second for 100,000 users so that's bandwidth not only receiving ourselves but in this day and age where we have mobile internets massively over taking {inaudible} internets and people have restrictions on their secular networks and it's costing a lot more for that band width we're saving our users and ourselves money. So to take nothing away from this talk polling is bad web sockets is good.

Imagine going to move on now to how we're going to use web sockets in Django and I'll take a couple of different approaches in this talk.

The first will be a way for you to host your own kind of real-time application of the structure and the next will be how you can use some of the same tools but hand off the actual hosting of it to a cloud based solution so you don't need to worry about the scaling and infrastructure behind it. In both instances we're going to use a thing called swamp dragon. Swamp dragon is a fairly new Django package I think the first commit was back in march last year so slightly over a year old. Don't let that put you off using it it's really everything you wanted from an open source package. It has great test suite's it has good documentation, it has easy to follow examples and the core developer and maintainer is quite ready to answer questions and feature requests to get help, yes it's slightly immature at this point but it really is a pretty complete package already.

So, what is swamp dragon? Swamp dragon is 3 things. It's Django. It's tornado. And it's Reddis. For those who know, Reddis is a key value cash or key value store. It is very similar to memcache in that it holds the data in memory so it's very, very fast but it also writes {inaudible} disk so can be used as a persistent store as well so we get the benefits of memcache but should we need to restart server or turn it off won't lose all our data and comes with pub sub publication subscription built in which is what we'll use for real-time stuff. Second part is tornado, Python based, a web server, a non-blocking - uses non-blocking U curls(?) so can use up to {inaudible} connections fairly simple and lastly Django, well if you don't know what Django is you'll be very confused the last couple of days. {Laughter}.

So how does this pull together what is our stack?

Well we've got our regular Django application and it still saves to a regular database in this case postgres. What we also have is in swamp dragon. Swamp dragon sits on top of our Django application and whenever we make modification to our data it sends the data to re dis. Whenever you develop this and you'll see examples shortly you are only working with Django, all the magic that does the real-time communication for you kind of bio directional communication via web sockets that's done via swamp dragon you don't need to worry about that, you write your Django applications more or less as you would any way and it takes care of the rest, so in front of Reddis we have our tornado server which is going to be creating our web sockets and they're going - we're going to create our subscriptions to these web sockets and browsers. We'll be talking about web browsers solely today. It doesn't mean it only works with web browsers, it can work with anything that understands web sockets, there is native libraries for IOS and androids, web sockets is just another protocol like http so really you can build apps that can subscribe to these services across a multitude of different platforms.

The easiest way to demonstrate how you build an application is to build an application so we're going to build another to do app because the world doesn't have enough of those.

To get started to do that we're going to install swamp dragon. Swamp dragon is a Python package a simple pip install. It will install most of your dependencies as well, you'll get tornado and things like that. What it won't install unfortunately is re dis. You need to install Reddis you self. You can do in apt gets brew install it's straight forward. The Reddis folks recommend you down load the {inaudible} off the Reddis site and do a stop to make sure you get an up-to-date version. The only place you'll find problems is on windows. Reddis doesn't support windows at all. If you are a windows developer then {laughter} I feel for you ... {laughter} if you are a windows developer you can run it in a virtual machine. There is also services such as Reddis to go, cloud based Reddis you can connect to instead but won't be able to install it natively on OIS.

Once we have these installed swamp dragon comes with its own dragon admin. Swamp dragon admin is very similar to Django admin and has a 38 project command. I personally don't like this approach. The 38 project will create a new project with D fault Django directory structure and I'm sure as many others here I don't use D fault Django directory structure I have a modified directory structure I have a different way of setting up my settings . I have multiple settings depending upon environments and lots of other things you'll probably - I've nicked from 2 scoops book so I don't

actually use the dragon admin but looking at what dragon admin does is, well, it creates a new Django project, we can do that ourselves that's straight forward, adds a couple of settings in your settings file, well documented, and it creates this server dot py. Server dot py is a new file swamp dragon drops into the route of your project that controls the run server command for tornado, it's very similar to the managed up higher run server that comes with Django except instead of starting your Django server it will start tornado server so you need to run both. It's great for development same as the Django server. You're not going to want to put it in production. You probably want to manage your tornado server much the same may as you manage your Django application and want to put that under supervisor or use g unicorn(?) yes you can drop it in don't need to use swamp dragon to create it copy into side drop yourself but when you go into production you probably won't use it anyway.

So, we have our application all installed, we have updated our settings file. Then create models. I'm not going to put the full models up here. It's a to do app it's pretty basic. We have a to do list model which has a title and description and then we have a to do item model which has a text {inaudible} to hold is, {inaudible} and a foreign key to our list so we can group all our to do items together.

You notice it looks very much like a regular Django model, there is nothing really strange or exciting about it. We do have this swamp dragon code that we've imported here. Mixing called self-publish model. What that mixing is going to do is it's going to override the save method on our model. So whenever we create a new instance of a model, whenever we instance of our model it's going to call this new save method and what that save method will do is take the data we're interested from that model and it's going to send it to Reddis so then we can push that out to anyone who subscribes to browsers via tornado.

What we also have on our model is a serialiser. For anybody who has not come across a serialiser before they're a way to translate Python code into something that your client can understand. So, in this instance we are translating our Python objects into Json because we're sending it to a web browser, web browser can't understand Python objects so we convert it to Java script which it can understand. The serialiser is completely new it's not a modification of a Django model or anything, going to create a new py file for it saw in the see serialiser we have the model serialiser and they're kind of like our model forms in that you don't need to use a model serialiser, you tell it your function, it's there to make things easier because a lot of the time you're going to be dealing directly with models but you don't need to only serialise model - if you wanted to build an app let's say was moderate in your server so was looking at your CPE load or at the hard drive space you had left, could serialise that data and send it. So like the models form and forms class you use a model form if you just want to do the basics model or get into forms class - same as serialisers in swamp dragon you use model serialiser if you want to serialise a model or you can dive down into the kind of bare serialisers themselves and write one for any bespoke data you may have with the model serialiser we're going to tell it here is the model we want to serialise, tell them what fields we're interested in publishing so our done to mark whenever we've completed an item and text if there is in an item and we're going to find this up-date field. The up-date fields will see you later front end and that allows bio directional communication so we're telling it that in the client we want to be in up-date done status, want to be the tick to do item in our browser and have that saved back into Django as well.

For anybody who has used Django rest framework in the past they'll probably be very familiar with serialisers. DRF uses something very, very similar. It's a bit annoying how similar they are because they can't be used for the same purpose so you do end up with a bit of code duplication. There is however a branch currently on swamp dragon that actually adds support for DRF style serialisers so hopefully in the very near future you'll write one set of serialisers and that will {inaudible} rest PR I via rest framework and also {inaudible} via notifications via swamp dragon.

As well as serialisers we need to create roots. Roots are very similar to views in Django. So the fact we have our object which is going to return an instance single instance of an object and we have this get query set which allows us to override the get query set method and return like our own filtered query set. In this instance whenever I am looking at a list I obviously only want to do items that are part of that list so I am overriding query set and go OK here is the list ID only give me items that correspond to that. There is a bunch of built in verbs we have our get list, also have get list, delete, create, subscribe, unsubscribe, these are similar to the allowed http methods in our class issues. In much the same way you can define your own methods as well, you're not stuck using these verbs you can define your own and subscribe to them in the front end.

When we do subscriptions where you give it a new name so in this instance it's called to do I am. You'll hear me referring to roots also referring to channels depending on the software you are using almost interchangeable. Some

will refer to them as roots some refer to them as channels. I'll use both in this talk so apologies if it gets confusing but in this instance they call them roots.

On the front ends a very basic kind of to do app front end. Use this thing called twitter boot strap, anyone heard of it? I don't know I never see it anywhere. On our template here we have our list title and list subscription then have some to do items 4 of which are marked in green as done one of which is in red still to do.

Now to make this actually function, we need to link it up to our Django code. I'm going to use angular. I know angular is no longer the hot new thing. I should have done it in react is that right is that the one that is cool this week? {Laughter} doesn't really matter which one you use to be honest or which one you prefer. Swamp dragon is not tied to any particular framework. It does come with an angular service that's why I use it because it's easier for me. If you wanted to use react or backbone you are free to do, also comes with Java script service if you are retro like that so don't worry if you don't write angular or you prefer something else or your front end team prefers something else. I'm using this for ease at the moment. It's not a deal breaker. You can use what you like with swamp dragon I'm not going to go {inaudible} that's really boring just the bit that matters and that's this small snippet that is going to control that template we saw.

So we have our to do list name, description then have this loop that's going to run through our list of to do items and it's going to put them on to the page as well.

We are wrapping it in these verbatim tags, the reason for that is angular, also uses the double braces syntax for variable names that Django uses, without them Django would try and replace our angular variables then it would disappear, we don't want that to happen, so we have wrapped it up.

Okay, I know it is way too small for people to see, we will go through it. Subscribe to your channel, subscribing to the channel is how we specify that we want to receive any updates on this route. So in this case, the to do. We are only really interested in to do items that belong to the first list, I have hard coded the list of one in here, if you are going the do this real, you wouldn't hard code the ID you would have some way for users to select which list you are interested in. Also worth noting that the query syntax for swamp dragon is similar to Djangos, pre-set filters syntax, the reason the double one to specify a property on a foreign key.

We have our get single, these are going to run when the page first loads, this is to make sure when you first arrive at the page it is not empty and blank. We get our initial to do list and items. Only interested in the first list, get single an ID of one., then get list, all the items in the database list and pop on the page for the first time. The list idea overflow clone one, but wouldn't do that in real life. Etc. etc.

So quick one back. So quick run back, subscribe

Get a list of our to do items, really thought better of naming these lists, pre-set lists and Python lists, when you call lists, the to do item lists, that will populate our templates.

Okay so. This is a bit of real-time where it comes in. We have the on channel message it is going the get called any time the server has new information for our clients, anymore new information for the browser, call this on channel message funs.

So we can check if it is the channel we are interest. ... The to do items, we apply the data mapper, look to see is this a new item, there is an existing item already with the ID, if there is, let's add that one, we don't want duplicate to do items.

Then I am nowhere brave enough to do a real-time demo on conference Wi-Fi, we will have a video, this is a real application, I do have it on my lap top. I can prove it does work, I am more than happy to do that away from the pressures that is the stage. Come and see me.

Here we have a chrome window, a fire fox window and a safari, then beneath the Django admin. As I go through, I am updating stuff in the Django admin, changing the status to done, immediately changes in all three browsers, normally it will change in the browsers before my Django admin page even finishes refreshing, obviously going the be running quick because it is running on my local machine, but when you are working on it remotely, because the connection is always open and ready to have information pushed down, almost as quick when you are working with remote servers as well. Notice I am able to update stuff in the actual browser and that is then sending that information back to Django, so it is bidirectional, not that we are pushing out from Django, but the clients can also send information back. We

won't see the updates automatically in Django, in the same way, but it is saving it back, so if I refreshed that admin screen you will have seen all the changes I made were reflected in the admin screen.

All the code for this is on, have a look, play around, put it to react, backbone or whatever you want., if you do mess with your urls's, don't worry, I will tweak them.

So in the first example we were introducing additional infrastructure, what is your normal Django stack, I don't know about anybody else, real-time, I didn't have tornado running, I had install just for swamp dragon, so it is adding additional complexity and more things to scale. I have ... I don't want to have to manage anything else.

I have much better things to be doing.

So we look at the platforms of services. There is a couple of different ones out there, fire base, we have pub nub, we have pusher, some of the sponsors to have day. There is a few database as a service out there, there is not a huge difference between people doing pub nub and database as a service, most concerned with data sink, making sure you have lots of read reply cars and the ability to sync across platforms, the other ones pub nub and pusher and ones you mentioned earlier are more interested in the pub sub part of it.

There is an interesting aside, when researching the different services out there. One of the ones is called data Fly, I don't know how many work in corporates or enterprise – good luck putting a purchase order through for data Fly.

Phil one of the pusher guys you may have seen earlier, done a really good Blog post, fairly non-biased considering he works for pusher on getting a. ... , don't take my word for which platform to use, read that, that should help you make your decision.

We will be using pusher today, it is pt1 I am most familiar with, it is kind of good for me up here because they do have this debug console, makes it better.

So how does pusher fit into our application stack that we had earlier?

So we are going the get rid of Reddis, going to keep pusher, we are going to keep swamp dragon but reduce the amount we use. The idea is we don't want to have to rewrite the application that we started with earlier, we wanted to keep as much of the swamp dragon stuff we can, swap out the data store, don't want to have to start off in swamp dragon and then hit scale and put in one of the other services and rewrite the application. Nobody wants to do that. Try to look at how we can keep much of the same but swap out what is it is actually doing the pub sub.

So now we are in a, ... how do we publish our information? Pusher has got a Python library, takes a pusher.trigger, send to channel. This is what I was talking of earlier, now we are switching to using channel to confuse everybody. We have a channel name and event, that is very much like the verbs we had earlier, so things like our updated created, deleted and then we have a payload, serialised model. Yes, that is my API key and secret key, yes, I did revoke them before I stepped on stage, better luck next time (LAUGHTER).

Okay, so how can we end up inviting pusher and swamp dragon, it sends our data to Reddis is our self-published model. That mix in at the start to be included in the models. We are going the rewrite the self-published model. So, we are going to keep our serialisers the same, they will still be the swamp dragon, we don't need to modify them in any way, but instead of sending the information to Reddis and then on to tornado, send it to pusher. Here you can see I have just used the same pusher code we saw a couple of slides back, sending it there rather than to Reddis, this is the way, I wrote both the applications, the swamp dragon first and then challenged to do the pusher application using few edits than I can.

This is the get dif. Most of it was deleting stuff. I didn't need the server.py. I no locker needed the, no longer needed the routers, that is by pusher, I added some code to do the self-published model that we saw a sect ago. Again using angular, I don't want to change, I am trying not to change as much code as I can., pusher has an angular library, it worked out well.

So this is our new angular code, I have had to make modifications here, because the pusher angular service doesn't map one to one with the swamp dragon one, we are now creating a pusher client, but we are still subscribing in much the same way. Subscribe to do item channel, then written a bit of code, updated to that same list we saw.

If I was to take this further, which is probably little bit out of scope of this, it would be java script heavy – we would look at the angular service and ensures it has a one to one map with the, the swamp dragon and our pusher angular

service and one to one map. The data we are saving through and the methods are the same. Then we can code the exact same.

Okay, another demo, so this time it actually isn't always going the be running on my local machine, sending information to the server and then going the be seeing that come back to the browser window, so I have the two browser windows on the left, then the machine, safari, I think fire fox, then the browser in the right is the debug console I mentioned briefly. The one on the right is running on pusher servers, the two on the left are my local machine, the admin is on my local machine, that is where I create the information and send it up.

As I create in the Django admin, you will see that on the pusher debug, you also roughly the same time see it in the two left hand browsers so you see it is being pushed back down the local machine.

That will work for all the same things as we had earlier, I will be able to create new to do items and edit, I will be able to delete to do items.

Code for that is also up on get hub, have a look, have a look at the dif, make sure I wasn't lying to you and give me any feedback you have got.

So the final bit I want to talk about, so far we have been talking about how you serialise model data, for a lot of the modification stuff, you might not want to send the whole model. This is a logo I am working with the council up in Glasgow, a data portal, a way for the council to open up the data they currently have and make it available to everybody, it is a pretty good product. Data from everything from bicycle rack locations through to school truancy rates and things like congestion levels and pollution and all the other stuff that should be able to us as, as like, members of that society or of that council area.

When I am working on the application, there is a lot of the data is automatically harvested from coup sill systems, a lot of the data can't be, the council systems are not exactly up to date. Not a lot of API's, available., it has to be manually entered a lot of the time by a team of data entry people. We found sometimes if you have got more than one person working on the same file or data resource, we can run into issues, we have our user one, two opens the source, user one saves the changes, user one changes vanish, that was the end goal to have some kind of collaborative editing, like Google docks style, but this is government work we don't always have the biggest budgets. Also it is not a problem that really occurred that often because we don't want to spend the developer time or the budget on it. A service called help Skype has a unique way of dealing with this. An icon that goes blue, red, it is telling you that somebody else accessed the thread and somebody is reading it and replying to it. That is all we needed. Needed to know when somebody else was working on the data set. Three desks over tell them, please don't do that. The easiest way to drop in the pusher code we saw earlier into our view, have it send an event r vent and the user name so we knew who was using it. Not really pythonic or dry, I have put a mix in, the response, this is published in a, about a week ago, some really good feedback on it. One of the things we are discussing, where is the best place to put the mix. If it is updating something, should it override? There is a big discussion I would love peoples feedback.

We are only really interested in updates about a particular instance of a model. We don't want to know when somebody is viewing any model. We want to know when they are viewing that instance. To achieve that per instances. So each channel gets a unique name to subscribe to in the front end, that is the model name plus the primary came. The data is the person's name, who else is editing this? Doesn't necessarily need to be something basic as that, we could use serialisers we saw previously, there is no reason not to. This could be any kind of data, using the same publishing methods that we did before.

On the front end, I haven't used angular, because it is small. Created a pusher client instance with an API key, subscribing the channel, the objects, this will only work on pages such as detailed views or update views that we have the object in the current context.

Yes, you can see I am kind of specified that there. I don't have a fancy video in action, it is still in development at work. I have wrapped this up in a Python package that is installable now from py.py. It is a few mix ins from your, creates update, delete, also the java script code we saw there, is all wrapped together in a nice template tags so really simple to use and also up on get hub. So this is a pull request going on, fortunate one of the kind of core developers of braces has been giving us some great feedback, I don't know anybody better to give feedback on mixings than anyone involved in Django braces, I am pleased to see that.

It is on get hub. I would love people to get involved in the conversation around that and the arguments where we should override the different types of events links there.

So, that is it really to be honest. Any questions?

(APPLAUSE).

FROM THE FLOOR: How do you handle log in authentication and access control to the data especially when you are using a service like pusher?

AARON BASSETT: Probably not the best, those guys, the swamp dragon stuff does have things for authentication as well, you can authenticate users before they subscribe to it. With the pusher I know they do have private channels which can also be encrypted as well, I don't know the ins and outs of that I am afraid.

FROM THE FLOOR: So when you push back the update from the client so when you click down on the to do item, it goes back query {inaudible} direction into the Reddis {inaudible} and from there reaches Django how?

AARON BASSETT: So although it's coming in via tornado then into Reddis we still have that swamp dragon layer and that's getting from tornado and that's back saving into your Django model.

NEW SPEAKER: So swamp dragon basically runs a listener thread where it listens for incoming connections from Reddis?

AARON BASSETT: Swamp dragon is integrated into tornado, that's why you see the run server for tornado in swamp dragon so it actually handles that side of it for you as well, as long as you are using the swamp dragon way of getting tornado up and running then it should handle that as well.

NEW SPEAKER: OK thanks.

NEW SPEAKER: That's great, can we thank Aaron again. {Applause}.

1.4.13 Loek Van Gent: True beauty is on the inside, but users are shallow

I think we're back on time now so here is Loek Van Gent from the one per cent club.

LOEK VAN GENT: We'll be talking about some of the things you heard earlier today just in Aaron's talk and David's talk but hopefully I can be of service to the ones that were thinking that's really cool stuff, but that's not for me. Hopefully I can convince you that part of things they were talking about can apply to any of you. I don't know, I reckon most of you would be back end developers. Oh, a first. Are you still awake? Who is asleep? Hi, quite some, who is awake, 2 hands yeah that's better!

The last talk but we should get pumped up for the lightening talks after right?

So, I'm going to be talking about and the title is true beauty is on the inside but your users are shallow. That should be a familiar - should sound familiar.

I want to tell you something about writing about front end code. So first about me this is me away from key board, I have done PHP for many, many years before as have others of you I think so too. 3 years I been riding ponies. Proper multi-Dutch Django association. If you are not a member yet and are from the Netherlands they just lowered the annual membership fee by 10 euros and it's free now so you should sign up. {Laughter}.

And I work for 1 per cent club. 1 per cent club has been remembered several times yesterday. We're not the richest people in America. We do run a platform for small projects that have a social impact. So far we're 800 - close to 900 successful projects, very small, couple of a thousand euros just to start for instance to buy solar lights for a village in Nigeria so kids can study when the sun goes down and very small, smart changes that have huge impact.

So, we're doing quite well. We are changing lots of lives, and doing projects with social impact that's something that also big corporates are interested in doing now and mainly as a corporate social responsibility programme, so now we're selling our web-site to them too and then we can earn some money because from do good crowd filling that we do you can't take money from that.

And so one day our product manager came and say, listen up guys, we've got a really cool idea, let's make a multi-tenant and a white label web-site out of the web-site we have. At that point, we were running our own web-site and not a web-site for a big company. And they said 2 web-sites should merge into one and make a multi-tenant. Interesting stuff of course. I won't tell too much about it but it was a big refactor we had to do and I only want to show the changes we did - oh no - it's too many changes to talk about it right now. But I can show you - you can imagine we went from multiple databases to one database with different schemas in it and it worked all very well. This is what overview with the project looked like. Then we changed it to look like this. Minor changes but the good thing is everything worked, people could still do their nations and do wall posts on the project site and even the project manager was really impressed and in big swirling letters he said wow this amazing guys we really love that new design! Yeah,, {laughter} and that was only 2 per cent of the time we spent on that. So true beauty is on the inside but your product manager is shallow {laughter} sounds familiar?

So what's all this fuzz about front end?

We have to go a little back in time. Second reference to that movie I think. OK.

Web-sites something like this. This is the first web-site. You all should recognise that right?

Anyone could put a year on this?

NEW SPEAKER: 1996, 7 ... {applause}.

LOEK VAN GENT: That's almost good. This is when the screen shot was taken. Before that they didn't take a screen shot it's from 1989 but good enough.

So this is what web-site used to look like and this is how mobile phones what mobile phones looked like.

And of course, mobile phones have taken a rise and we're not texting like this anymore we're swiping away and playing games and using the apps and in fact phone cells the interface to mobile apps are so - come so naturally, that they are more easy to use than most web-sites, they're intuitive, they know what the user wants.

And web-sites are starting to lag behind, the traditional web-sites. Of course you have things like Gmail. Anyone has this mail in their in box by Baptiste that you really should start filling in your dates for the menus that you want to have? ... Of course you can't call this a web-site any more. It's an application. So, that is clear.

Web-sites, the front end for a web-site is the user facing part of your web-site is very important, but how can we write better front end codes?

Now it's not a live demo but it's a console screen.

Let's create a Django project. We can use Django admin for that. Probably everyone did at once when they did the manual or the tutorial and after that narrow down but let's try it. We're going to create a project named true beauty. We're going to add an app called heart and a app called spirit.

Let's look at the project layout.

You'll have the main project and your 2 apps down; that's all good. The templates and the front end doesn't have those bits yet but normally you would place those inside the heart and in the spirit.

Because they're portable and you can take them out and copy and paste them somewhere else, that's traditional may how we Django go noughts think but let's ask Danny and Audrey what they think about it and they say you have to really group your front end code, at least I have to look it up in their newest 2 groups book, 2 scoops has been mentioned before and it has been said you should own that book so that's all clear. So they say group your front end code, group it in a typical project Django project layout. You should put your templates and your static files, your Java scripts inside your main project folder.

This is where you are. This bit is for the front end guys. That's it right. But I don't know, how many do full stack or - full stack? Most of you right? So, if you have a bigger company then you would hand that over to the front end guys and when we first did this in our company to start a grouping, our front end guys were really happy because they turned crazy every time we did changes in our template and had to {inaudible} all their app folders.

Let's look at how we can further improve that little bit that's really the main front end.

I'm going to just shout out some tools. Hopefully most of you are already using them or similar one we can discuss about it later if these are the best ones or the other.

So, take Bower, it's very useful to do your requirements instead of having just a base HML where you specify all the Java script you want to include like J query and J query UI, you have dependencies like you have in your Django folder with your requirements file.

If you've ever tried to do CSS coming from the back end you would, yeah, it would still be spaghetti code, you switch to Django from PHP and really happy, but with something like sas or las you can really structure your code that would in the end generate CSS and you would use something like Grunt to run those tasks to create your sas file or minify some templates and you can use NPM tool to install all that so if you do deploy NPM will ensure everyone is in power and run Bower and suddenly everything is magically done.

When I first started with projects, running Django I would just include my J query in my repository which is a bad thing, I hope you all agree. Yes so you have all these tools to get your front end in shape but it's starting to look like an entire app in your Django project right?

Now you're getting it!

So you have this back end and this front end. The back end is probably has to deal with some file storage and the database and there is probably a user talking to your front end hopefully. You need some back and forth communication.

Django server can talk, can perfectly have that data, talk to your front end using Django Rest Framework in Json, that has been - it was in error sock as well of course. In front end you'll use full tack framework like Ember or Angular or React.

The good thing about splitting those is if you have an API in your server, if your server is just talking API, then you can easily test that and have an integration test. Of course you can have your unit test on a low level, but you can start writing your back end and you don't have to wait for all those front end as they are tweaking their pixels and can just run off if the functional design is clear, you know what the API should be able to spew out and accept back then you can easily write your test and your back end is done.

And the other way round is the same. You can start creating your front end, you just use a mock API, and you don't have to wait for all those back enders over complicating things. You can easily run a demo if they suggest new designs, it's very easy to just change the styles, and you can even deploy the front end separately from the back end unless you have API schema changes of course.

So, what we use it's not our hit thing Aaron was trying to think of. We use Ember CLI, for our front end, it's just an example how you can use it. And I'll leave enough time so we can argue if Ember or Angular is better.

Back to your console. All you do is ember - no we want to create a new application. It's called good looks because it's on the outside. Let's go into good looks. We generate the route that will - Ember starts with the route and then your MPC(?) pattern backing that up and create a smile, that's the 2 apps to talk in Django terms and it would look something like this. You have your app JS that will spin up your ember server and then all the components that will give you the eyes and the smile and will make your user happy.

If you want to really go wild, you can also use ember serve that will spin up a web server and you can proxy your own API after that and then you - then the API calls will be proxy to your Django server. And then you have a fully independent front end just like a web app, just like a mobile app, but backed up by your API.

So, summing it up in a few easy steps to hopefully give you a hand to changing your full stack Django web-site into 2 things back in the front end, first thing is love your front end. It's the only thing you use they get to see. You don't see the marvellous thing you've done on the back end and the marvellous ways you manage your database and stuff like that. Group your front end, if you are the only container, Kat where are you? Then still it is good to have it in clustered in one place.

Use tools to help structure your front end code, if you are not using barrow or NPN or something like that, try it, the same goes for, for instance Ember start using it, try a demo organisers use Angular or React, but forget about Backbone, it used to be popular among the Django developers but that is not the way to go to build it all yourselves.

So that is all from me.

If there is any questions or suggestions please, be welcome.

(APPLAUSE).

FROM THE FLOOR: Thank you for your talk, that was a really good talk. I did look into Bower and Grunt and all those lovely looking tools, all my colleagues, I was on a didn't have the internet and the project I am on, the client didn't have the internet on the network we are going to deliver on. Any advice for trying to use the tools without access to the repositories they will pull from?

LOEK VAN GENT: So you are talking about an intranet.

FROM THE FLOOR: Essentially.

LOEK VAN GENT: I would reckon you could do a stage deploy, so first, make sure step one, build it as you would in a normal life in the world website and then deploy that. But yes, I am not sure.

FROM THE FLOOR: Okay thanks.

FROM THE FLOOR: Hi there, you mentioned about multi-tenanting bit, what package did you use and what problems did you have with it?

LOEK VAN GENT: Were you we were, we were using Django tenant schema.

FROM THE FLOOR: I patched that,.

LOEK VAN GENT: We did the same thing, extended it with useful stuff. I did send one pull request that I didn't get on.

FROM THE FLOOR: Maybe I will have a chat with you, thanks.

FROM THE FLOOR: Thank you very much. I would like to know if you did try to implement some life with load, so every time you changed your SASS files or java script files automatically the ...

LOEK VAN GENT: Ember does ... I don't know if you mean that.

FROM THE FLOOR: You don't use a grant for that?

LOEK VAN GENT: No, so if you use embassy live, use that as the web server for the front end, that fires up live reloads if you are coding it will reload like the Django server run server can do right? Is that what you were looking for?

FROM THE FLOOR: Yes, the way to implement it, Django server with this life will load methodology.

LOEK VAN GENT: Ember will do the same and I think Angular has started shipping a CLI as well I think.

FROM THE FLOOR: Thanks.

FROM THE FLOOR: Hello. What would you recommend if I would, were to, if I would like to recycle one of the apps in another project? With front end and all?

LOEK VAN GENT: Sorry?

FROM THE FLOOR: If I have Django app and the java and I would like to use it in another project, how would you do that if you can't group it all into one project?

LOEK VAN GENT: Yes I think it shouldn't be in your projects from the start. It should be separated out but I think the question you want to ask is, you have this lovely app that is we all use, all these libraries and they do also have templates in them right? And if you split it yes, then you have this, that Django template stuff going on and you have to rewrite that in your Ember or Angular application right? Yes. I don't know.

I understand the problem. But, yes, the main thing is, how, how can we continue using Django in, now I think we all sense this is happening that we, that web servers will be mainly to pride data and that representation will happen in the browsers. But, yes, I am, I don't know. So you can really use your app up and until the API so the API framework. I think.

FROM THE FLOOR: Thank you.

FROM THE FLOOR: It is going to be really quick, I just want to answer the question that has just been asked.

If Ember is your plus Django is your regular stack, ember is component if you want to do an app in Django, you can usually map it to your component in Ember, when you want to have your template with your reusable app, you can pack it as a component and put it in there.

LOEK VAN GENT: True.

DANIELE PROCIDA: Thank you very much Loek.

(APPLAUSE).

1.4.14 Matthew Somerville: Using Django's StreamingHTTPResponse for JSON & HTML

MATTHEW SOMERVILLE: Hi, I work for mySociety, a charity that builds things that empower citizens, in the UK and around the world, such as FixMyStreet and Alaveteli (FOI).

MapIt is a Django application for mapping points or postcodes to administrative areas, either standalone or within a Django project. Our UK installation powers many of our own and others' projects such as TheyWorkForYou; Global MapIt is an installation that uses all the administrative boundaries from OpenStreetMap.

A few months ago, one of our servers fell over, due to running entirely out of memory. Looking into what had caused this, it was a request for information on every "level 8" boundary in Global MapIt. This turned out to be just under 200,000 rows from one table, along with associated data in other tables. Most uses of Global MapIt are for point lookups, returning only the few areas covering a particular latitude and longitude.

Using python's resource module, I manually ran through the steps of this particular view:

50Mb initially. Plus 500Mb fetching/creating a lookup of associated identifiers for each area, plus 675Mb attaching those identifiers to their areas, which runs the query to fetch the areas, plus 400Mb creating a dictionary of the areas for output, plus 250Mb outputting the dictionary as JSON. A total of 1875Mb!

The associated identifiers were added in Python code because doing the join in the database (with select_related) was far too slow, but I clearly needed a way to make this request using less memory. There's no reason why this request should not be able to work, but it shouldn't be loading everything into memory, only to then output it all to the client asking for it. We want to stream the data from the database to the client as JSON as it arrives; we want in some way to use Django's StreamingHTTPResponse.

The first straightforward step was to sort the areas list in the database, not in code, as doing it in code meant all the results needed to be loaded into memory first. I then tweaked our JSONP middleware so that it could cope when given a StreamingHTTPResponse as well as an HTTPResponse. The next step was to use the json module's iterencode function to have it output a generator of the JSON data, rather than one giant dump of the encoded data. We're still supporting Django 1.4 until it end-of-lifes, so I included workarounds in this for the possibility of StreamingHTTPResponse not being available.

But having a StreamingHTTPResponse is not enough if something in the process consumes the generator, and as we want to output a dictionary, creating a dictionary for json's iterencode will suck everything into memory, only then iterating for the output – not much use! I need a way to have it be able to iterate over a dictionary…

The solution was to invent the iterdict, which is a subclass of dict that isn't actually a dict, but only puts an iterable (of key/value tuples) on items and iteritems. This tricks python's JSON module into being able to iterate over such a "dictionary", producing dictionary output but not requiring the dict to be created in memory; just what we want. I then made sure that the whole request workflow was lazy and evaluated nothing until it would reach the end of the chain and be streamed to the client. I also stored the associated identifiers on the area directly in another iterator, not via an intermediary of (in the end) unneeded objects that just take up more memory.

I could now look at the new memory usage. Starting at 50Mb again, it added 140Mb attaching the associated codes to the areas, and actually streaming the output took about 25Mb. Whilst it took a while to start returning data, it also let the data stream to the client when the database was ready, rather than wait for all the data to be returned to Django first.

But I was not done. Doing the above then revealed a couple of bugs in Django itself when using StreamingHTTPResponse with GZip middleware:

- It would die with any Unicode data;
- Gzipped responses were larger than non-gzipped.

Both of these I patched and were included in Django 1.8.

Lastly, in all the above, I've ignored the HTML version of our JSON output. This contains just as many rows, is just as big an output, and could just as easily cripple our server. But sadly, Django templates do not act as generators, they read in all the data for output. So what MapIt does here is a bit of a hack – it has in its main template a placeholder, and creates an iterator out of the template before/after that placeholder, and one compiled template for each row of the results.

Now Django 1.8 is out, the alternate Jinja2 templating system supports a generate() function to render a template iteratively, which would be a cleaner way of dealing with the issue (though the templates would need to be translated to Jinja2, of course, and it would be more awkward to support less than 1.8). Alternatively, creating a generator version of Django's Template.render() is Django ticket #13910, and it might be interesting to work on that at the Django sprint later this week. Thank you!

CHAPTER 2

Indices and tables

- genindex
- modindex
- search