Djangocon 2015.

Date: 3rd June 2015.

DANIELE PROCIDA: Good morning everyone, welcome today 4 of Djangocon. Welcome to Wednesday. Excuse my jacket I've been splashed by another member of the Django core team to start the day so ... {laughter} ... I'd just like to remind you that if you are interested in taking up the opportunity for a counselling session with one of the university's counsellors all you have to do is pick up a post it note from the sheet at the back. They're on post it notes; just pick one up if you've got yours. Several people have spoken to me and said they have found it helpful, it's not going to solve anything but it might be the start of something.

I think that the room numbers on those might be wrong so we're going to have to change those in a moment. Sorry that was the counselling point. . .

In front of me here I've got a kind of skate board which I won't demonstrate to you - and these are prizes which have been donated by sponsors, cape board from {inaudible}, will be giving out prizes in a raffle, we'll do a collection for funds to help members of the Django community travel to events who wouldn't otherwise be able to afford it.

We're amassing a collection of lost and found items. If they're not claimed perhaps they will be added to the raffle {laughter} so there is a rather nice jacket with a purple lining found in the Django Girls workshop room on Sunday.

NEW SPEAKER: It was there when we arrived - I shouldn't have brought it over -

DANIELE PROCIDA: Oh you don't think it is ours? We can try it on for size. Somebody has lost a pair of glasses so we've got those. ... Sunday night? The glasses were lost at the VSF on Sunday night.

Meeting today during the lunch hour and a half at 2 o'clock, just back there behind the partitions if you are interested in talking with Django noughts interested in that kind of thing. Using Django for adding social value to the work.

The artists from the Cardiff print workshop will be here today. They're the artists who produce the art work that we've got on screen on our teeshirts and the artists who produce the 14 different prints that you've got in your bags so they'll have some stuff for sale and you'll be able to meet them, have a chat and they'll be pleased if you say that even if you don't buy anything you enjoyed their work.

Actually I want to say a couple of things about this particular Djangocon. And the first thing of course is me are all amateurs running this and an amateur is strictly speaking someone who does things because they love it because that's what the word means and this brings us some disadvantages but advantages also and we've tried to make as much use as we can of the advantages that being amateurs has given us so we've been able to put on an event that conforms to our vision of an event even if it's slightly unusual in some ways. And another thing we're able to rely on is the very different relationship with our attendees. You've bought tickets or someone has bought a ticket for you, but you are not our customers, we're not in a customer provider relationship. So, the first thing that happened on Saturday when people arrived in Cardiff is that they came to the university to find out if they could help doing things like bag packing. You wouldn't get that if you were the customer. The customer doesn't normally go to the other side of the service desk. So, thank you everyone for not being our customers but being participants in a really genuine sense in an event like this.

I want to say a few things about some of the things we did specifically at this Djangocon. The first one was our open day on Sunday which was a day of workshops and talks, it was our attempt to do some local engagement beyond our own Django community to get people in who wouldn't otherwise come and also had the function of giving some people the opportunity to attend workshops for example, talks, that would prepare them for the following day. So, I think that worked quite well for us.

You will have noticed that we had a very large number of first time speakers at this event. Faces that you haven't seen before. And this was a deliberate choice. We supported it with a speaker mentor programme.

Now I know many people here put their names forward to be speaker mentors to just offer support to speakers who ask for it. So can you put your hand up if you signed up for that? Yes I know you did, Russell, there were way more than that, there were about 10 people, so thank you very much, I know that was really appreciated by the speakers and I'm sure it also helped them produce - {applause} - it helped them produce better talks.

Everybody pays to come to Djangocon. Even the organisers. Except the people who receive financial assistance. We didn't provide any financial assistance ourselves. The financial assistance has been provided through the Django software foundation which operates a grants programme and we're extremely grateful for that because there are quite a few people here who benefited from that. At the same time, what we were able to do is give out a number of free tickets to people who wouldn't otherwise have been able to afford it.

So, if you bought a ticket or your company bought a ticket or you're a sponsor, thank you very much because you have made that possible.

We've had the first Djangocon crèche with at least 2 children in it and that has made it possible for a couple of people to attend talks who wouldn't otherwise be able to attend.

We've made big efforts in accessibility, most obvious of which is the marvellous speech-to-text transcription, by Hilary and Sheryll, so they've promised to do a little lightning talk at the end of the day to explain what they do for us - {applause}.

In terms of our speaker line up I think this is the most diverse Djangocon ever. We're still extremely male and extremely white and I personally am probably going to be both of those things for as long as I live, that can't be helped {laughter} but on our open day, one in 4 of our speakers was a woman, and in the talks today I think it is 1 in 3. So, each year, you manage to do better with that, and I'm sure that next year portions will be even higher. It's a ratchet effect. It's a lot of effort, but bit by bit we are making the speakers and then in turn of course the audiences less of a monoculture and more of a healthy diverse one.

We've also been thinking about what happens beyond our own little lovely Djangocon bubble.

Now, we had a very strong social responsibility policy right through this conference. You might not realise this but me are the 0.1 per cent club. That is the population of Cardiff is about 340,000 people and there are well over 350 people at this conference. So, we represent 1 in a thousand people in Cardiff. That actually gives us a lot of power. If we decided to riot, for example, {laughter} there probably aren't enough police in the town to hold us back.

NEW SPEAKER: Ooh! {Applause}.

DANIELE PROCIDA: So, don't let's get carried away! ...

But we've been able to enforce things - for example, we've got a policy on waste. Most obvious thing you've seen is the bottles you have in your hands. Our food policy. Our food policy the way we've been able to bug you and bug you about telling us which means you come to so we don't throw away food is possible because you are not our customers but are part of our event, you are participants in it.

We have a food policy at the event. We've asked all our suppliers about where the food comes from about animal welfare standards so we're not serving battery farmed pork or chicken. You can do this when you are saying to a provider we've got 370 people coming for lunch. What can you give us? So, we can use the weight of our community to change other parts of the world.

We have been able to talk to our providers and actually choose our venues based on their pay policies. So Cardiff university for example and Cardiff City Hall which is owned by Cardiff Council are both signatories to the living wage foundation which guarantees its lowest paid employees and the employees of their contractors a rate of pay that is significantly higher than the legal minimum wage.

The teeshirts that you are wearing and the bags you've got were not made in sweat shops. They were sourced from companies that sign up to fair trade standards and have a responsible sourcing supply chain.

So, we can make a difference not just to the wellbeing of ourselves and each other, but also through our sheer numbers to the wider society that we're in so, thank you for listening to that and thank you for helping us do that. {Applause}.

VINCE: The first speaker is James who is a lover of PHP and ancient Greek. {applause}.

JAMES BENNETT: Good morning. So, one of those things was true ..., I did in fact take ancient Greek in school, I did do PHP for a living before coming to Django.

To get started just a quick bio. My names is James Bennett, I've been working with Django about 9 years, Django came out summer of 2000 and 5, I started playing with it, ended up getting a job at the Laurence journal world in early 2006 to work with Django, stayed there about 5 years, got my commit bit in 2007 and here is the thing that's relevant for this talk: I've been involved in Django's release and security processes for about 8 years now.

So, this is a security talk. It is not the security talk. We had several of those.

What we're going to talk about here is not so much how to write a secure Django application or not so much from a user perspective but more from our perspective in the Django core team having been there for almost a decade watching release and security processes. Talk a little bit about the history of Django and how we've approached security. Some of the things that Django now does to try to protect you and how we've added those over the years and the thing that you are actually here for: how we've screwed up over the years and what you can learn from that.

So the overriding message here: security is hard, really, really hard.

I am going to talk for a few minutes about things we've done, processes we have, things we've put into Django to try to protect you, and it is going to sound like I am saying "oh we're really good at this me know what we're doing, we're good at securing" - we're not, in fact that's why the last time part of the talk is entirely ways we have failed at that.

Security is hard. There is no such thing as a secure framework, there is no such thing as a secure application. There are always going to be moments when some things are not quite right, there are always going to be things you could be doing better. If you take away anything from this talk hopefully it is that we've spent 10 years with this philosophy of we're not always going to be perfect, but we're going to try to get better.

Let's look at how things were once upon a time, let's dial back the clocks all the way to August 2006 and January Django's very first published security vulnerability.

Now there is some good and some bad going on here. The good thing is this was found without anyone exploiting it, in fact it was found by a member of the core team, it was fixed, it was acknowledged, before anyone else you know started hurting people. We provided description of the issue and we published a fix for it.

There are some bad things here though. One of those is if you go back and read this announcement and you can find it in the Django project web log if you go back to August 06, even if it's not a particularly easy thing to exploit arbitrary command execution is kind of a big deal that's kind of big security hole and this blog post kind of played it down a little bit, well, you know, you had to have these circumstances to exploit it, and we don't know of anybody ... that's not necessarily the best way to talk about a security vulnerability.

The other thing is - I want you to notice ... look at what we recommended as a fix. Oh just down load a new copy of this file and over right the old one.

That is really not a great security process. Now, that was 2006. It has been 9 years since then. We have had 48 more reported security issues in Django, in fact we have an archive and I'll give you a link in a moment. Let's look at Django's latest vulnerability which in fact was less than a month ago, May 20's this year.

We are not telling anybody to down load a file and replace, we are doings a full release.

That is bare minimum you do a release, on project process.com, the description here doesn't pull punches or try to down play what is going on this can result in people getting screwed up session.

We have credit to the reporter, thankfully now, we have a community of fairly responsibly disclosing, responsibly reporting people who find security issues in jab Django, they let us know about it so we can fix them this way, there is a proper call to action which is upgrade immediately! Although I don't think that made it on to the slide.

So, if you ever want to go through this entire list, these are all in log posts that is how we publish them. If you want to see every single security vulnerability we have ever had in Django, they are all archived right there, we add one every single time.

So, a little bit of the history, how did we get from there to here? We started no 2007 this is before Django 1.0. We started, if you want to look up...

Was the first time we pre-notified Linux distributions we are shipping a package; we are going to have a security issue coming up. That was the first step.

Next step Django 1.0 template autoescaping, if you have characters html sensitive, the template escapes these for you. This was controversial when it was added there were a lot of people who did not think Django's template system should autoescape, there were some fairly long debates on the mailing list, eventually it went in. It went in because the philosophy that had been decided at that point already in 2008 which is a while ago was this is too important, it is too important to leave to chance, Django needs to protect you from cross side scripting vulnerabilities out of the box, we can't just put a note in the documentation saying, by the way, escape your html that leads to bad, bad things.

Next step we picked up in 2010 Django 1.2, the modern CSRF framework, they were two middle wares and they were optional. Several problems with that.

The modern CRSF framework is one middle ware, in core, on by default, you are protected from cross side request boardry out of the box with Django, sometimes it is annoying, you have to remember to put the tag every time you have a form.

But again that philosophy was developing, it is better to be safe out of the box, better to have someone to come to the mailing list and complain, why are my angle brackets escaping, a forbidden response on this. Better to have that than someone say, my site got hacked, cross site requests, my users got hijacked it is better to be safe out of the box.

Next big step was Django 1.4, there is a big list, it was a really big deal for security in fact it is still a supported release because it was a long term support release.

Django 1.4 was sort of the first version of Django that I would say took security really, really seriously, we added much better password hashing, crypt traffic tools, the signed cookies, the sensitive error protection. Who has run into Django's sensitive error ... this is a feature that is maybe not as well documented as it could be.

You know, when you are running debug, you get the nice big trace backs or when you are not running debug, in production you don't with debug on, you get the e-mails sent to you with the trace back of errors on the site. There is a problem, they contain all of the local stack variables and all of the request parameters, some of that might be sensitive information that you don't necessarily want to show in a trace back on your debug page or to your site admins in an e-mail. So Django does by default suppress a few things, mostly passwords and has decorators to add to the views, to say to depress other variables as well. One is sensitive variables and one called sensitive post params, depending whether you want to request variables or things coming in, in the request body.

If you want to read our security policy they are also available at https. ... That will link you to the latest version, updates on occasion as we go in and edit and make minor fixes. We have a published security policy saying how we will deal with security vulnerabilities and how to approach them.

Django 1.6, host header hardening, we will talk about that in a bit. That was one of the failures, we got increased hash iterations, when you hash a password, the naive thing, when Django was released we did MD five without salt., now the default is ... which is more complex and we do a lot of iterations of it, originally we did 10,000, now we do 24,000 iterations when we are hashing a password, we have been bumping that up. Hash truncation, that was solved by the ... hasher.

Password truncation is a fun thing, there are some algorithms that will truncate the characters before they hash. The original crypt truncates ... so as long as yaw know with crypt the first eight characters you can throw any junk on the end of it, it works because it crops it., B crypt, if you know someone who has a 72 ... we have a version that hashes ... before going the B crypt to avoid the truncation problem.

We also gained more recently the system check framework and in one point the security middle ware and the deployment ... to the system check framework, who has used the system check framework? Who has seen it in a way, other than complaining in a way and fixing it.

Fairly new, it will check a lot of things in your set up, your configuration, there is also a deploy option that you can point at your planned deployment settings and it will tell you if there are problems with that, it will look for insecure settings, it will look for insecure configurations, tell you about them, warn you about them. Tell them how to fix them. The security middle ware is new in 1.8. This does some fun things, adds settings that lets you control script transfer ... turns on an in browser cross sight scripting filter if available. That is kind of a fun feature, browsers now can look for things that look like java script in a submission, look to see if the same text got echoed back in the html source and refuse to display the page. You have to opt into this, the security middle ware will opt you in. It is another layer of cross side scripting production.

So that what we have added in the framework, gained a formal security process which was laid out in Django 1.4, if you want the easy vexer, e-mail security at ... if you think you have found a vulnerability, you can read the full policy.

This basically is the fun times, we first tried to verify the issue hopefully, either we have or can come up with a proof of concept, talk to the reporter, once ... next we patch it, patches again under the private tracker, we review them, if we need to, we port them o multiple versions of Django, at the moment, security support to versions of Django., fairly even these need to be back boarded, the recent one only affected 1.8, we tell you which versions are affected.

Then there is a notification process, request a CBE identifier, this is the common vulnerabilities database, this is the standard way to refer to security issues, we request one every time we have a problem, one week prior to the release we send notification, full description and patches to our private security notification list. This is something that we occasionally get requests to join. There is documentation in the security policy about this, our security notification list is very tightly controlled. It is not just a list of people who use Django. It is mostly the big critical distributors of Django, this is most of the people on the list are Linux distributors, who ship, they need to do advance warning that we are going the do a security release.

Sometimes we abbreviate ... (INAUDIBLE) I remember not so fondly doing once a release at one in the morning in a hotel from Denver, which is not where I live, there was a denial of service issue that was reported publicly and was in the wild.

Then finally we do the release, we merge the patches from the private tracker, issue the releases, go up on Django.com and ... Blog post and identifies then spam all the social media site, we put it on hacker news, IRC, send it out via e-mail, we do all of that.

The goal here is responsible disclosure of security issues we rely heavily on the community to help us with that to come to us with the issues they have found, help us keep that information confidential for the time it takes to find a fix and get the release out.

Hi Rus!

So, what does Django do in terms of security for you?

If you have never heard of the open web application security project, wasp, they publish a lot of, they publish a top 10 list fairly regularly, you can see that top 10 list right there at that url.

Their current top 10 list is injection attacks that sequel injection, e-mail header injection, that sort of thing, Django is hardened to that by default.

Authentication and session management, these are really, really hard as we have learned several times over the years. Pretty confident nowadays in Django's session frameworks they try to follow the best practices.

Cross side scripting, got you covered on that.

Direct object references is kind of a weird sounding one, this is where you see a url and you can tell it has a database ID in it, hey I wonder what happened if I change the ID to some other number? Django by default doesn't push you into either one of these but our url configuration system was designed to produce human readable rather than database ones. You strongly recommend you do.

We have the system check framework, can check both your develop it and your production settings to look for potential misconfigurations.

Sensitive data exposure. We can't force you to use https. Baptiste is currently working on a patch if you deploy under non-ssl replaces everything with emoji pile of poo, so we can't force you to do it, we do provide a lot of support for using https on your site. We provide http only cookies, strict checking, (INAUDIBLE) gets better, a whole bunch of features for that.

Function level access control is making sure that making sure that not at the database level, ... see and access things, Django contains permissions and. Cross site request forgery, we have ...

Components with known vulnerabilities, we don't have a lot of control over, we do our best to try and notify and deprecate, we require recent versions of the database and the drivers and when there is a major issue we advise on it. We did the I think two years ago, there was discovery that Python's xml libraries had a major vulnerability with n them, we did an advisory on them.

Finally, unvalidated redirects and forwards. This is number 10. This is sending a user some place. Django itself has gone through a lot of evolution to try and nail this one down.

We also do a few more things above and beyond. We have a click jacking ... integration like I said with the in browsing, cross side scripting, we have contribute graphic signing ...

Constant type ... who has heard of timing attacks?

You are all are awesome. Basic idea here is, you submit a password, we hash it and compare it to the hash in the database, there is a problem, if we go character by character and bail out, you can tell how much you matched by how long it takes. We use a constant time algorithm, takes the time to look at the strengths.

Releases, every single release of Django, comes with a GBG sign ... git sign tabs, ... can verify that you got the Django we intended for you to get.

So all of these things have been added over the years. Going back to those template autoescaping discussions, Django has a philosophy of trying hard to be secure by default. Trying very hard to help you, our users, our community be safe by defaulted and give you the tools you need to do that.

However that is not what some of you are here for. Some of you are here for this. Because it turns out we have had some security problems in Django over the years.

It is worth asking why?

Batman begins I believe, Alfred asks Bruce Wayne, why do we fall? We want to do this properly for the internet, the question is why do we fail?

Most of the problem is good intentions and overlooking things, so for example, I will show you some quotes. These are all reflections of things we have done at various times. Parsing the accept language header is hard to do every time, let's do it once for every unique value and cache the result. This is a great performance optimisation. Use a base 361 time token to do password resets there is good. Form sets should be able to dynamically grow the number of forms they use, this is important feature you need to be able to add to a form set.

Restrictions on password link are dumb.

Long passwords are better.

Anybody want to guess what those quotes had in common?

Well, oh that was supposed to be a performance improvement.

Oh, that is bad.

Oh dear.

Oh no! no!

Huh!

So let's talk a little bit about those.

The accept language header. We were keeping a dictionary of the value language, the problem is someone sends you an accept language header that is this long then they do it a couple of 100 times and you are storing all of them in memory. Turns out the denial of service factor. We actually ended up changing it to shorten and normalise the header values before using them as keys in a dictionary.

The pass word reset, they based 36 token, you could send us some pretty long values for that and we would spend a lot of processor cycles trying to verify it even though it had no chance.

Yet another denial of service factor.

Now the pass word reset token we'll just automatically reject anything more than 13 digits long. If that sounds short remember in base 36 that handles any 64 bit anyger. If you want one that is larger than a 64 you are on your own.

The form set problem. There was this value called maxnumb that you could or could not set and if you didn't set it we trusted a hidden field on the form set to tell us how many forms to generate which meant end-users could just plug in a number like, say, a very large number, like one larger than sys dot max INT and cause an overflow error or chew up all memory making it substantiate 10,000 forms.

Pass link was the fun one. Who thinks long pass words are bad? Long pass words are good - until the denial of service factor. Turned out where we were doing the iterative hashing with PHFK 2 got really badly slow as length of pass word increased. At first we did introduce a cap, later we went back and sped up the algorithm, we figured out we were doing it wrong by rehashing the H max key on every iteration, we didn't need to do that.

There was a theme here. Something we're probably used to from working in Python instead of a language like C or C++ because Python is really great because it doesn't have buffer overflows so we feel really, really safe about writing Python. Turns out you can still very easily do a denial of service on yourself if you're not careful. As you've seen we've learned that the hard way multiple times. Please take our lesson. You can't see in the back that comic it says "and the Jane sell said to him, stop hitting yourself, but he could not stop for the angel was hitting him with his own hands". I'm not sure which book of the Bible that comes from but it sounds about right.

So, some things to remember about this. All of these basically came down to input validation problems. Sanity checking things before we started to process them like that base 36 token. If you are sending us a 500 digit based 36 token there is no way that's valid, we should just reject that before we start trying to process it. Yes you should even when appropriate you know sanitise your pass word inputs, configure your web server, this is something Django can't do for you but every good web server mill let you limit the size of http request bodies or headers so make sure you are doing that if you're worried.

More quotes.

"URL field should check whether URL exists before accepting the value. URL field should accept anything that matches value or URL, email field should accept anything that matches format of valid email address.

Checking for corrupt image files is easy. PIL has routines for that.

Most image formats store their meta data in a header, let's find it by only reading a few bytes of the file at a time. "

What do these have in common? Who knows now what's coming oops,, oops,, oh no ... that's bad ... yeah ... once again ...

So, URL field used to have this option verify exist. That had to be removed ultimately. We tried a couple of times to make it secure. We were giving you the ability to give you Django application URL and make it send a request and you could very easily to make that request take a very long time and tie up the server process while you did it. URL email fields tried very hard to match anything that was legal, turns out it's very easy to make those regular expressions take for ever to parse. Now we use much simpler regular expressions that don't actually match everything, everyone's in a file someone will post an article on hacker news about why are you and you allowing everything that's allowed in an email address. Talk more about that in a moment.

Denial of service via compressed image files. Turns out some of those routines will unpack a compressed image. What's the problem with compressed images? They can get really big when you decompress them. Ever seen the zip bomb where the zip file containing a zip file containing a zip file - it's the same basic idea, there is an attack like this against XML.

And denial of service via large image files, switch to a few bytes at a time, switch to the what we should have done doubling the read at a time because it turns out some formats you have to read the whole like to get meta data like image size by {inaudible} each read you drastically cut down the number of reads, this gets into something you'll hear referred to as big O notation. I shamelessly got that image off wiki. Big O in mathematics and computer sciences expressed an upper bound on complexity and run time of your algorithm, it's also an expression of an MNA(?) but I didn't want to see that. More important is the upper bound bit. Sometimes talk thought about as worst case, technically not a correct description, go with upper bound, but leads to a very important question which is: what's the worst that could happen? This is a very important question to ask yourself when writing code.

Figure out how much work your code should be doing. We didn't. Then figure out whether you can make it do even more work than that. We didn't do that multiple times. Then if you are us after issuing a security advisory figure out how to make it do less work. A lot of these issues they're not on top 10 lists where you can still find documentation about them, things like compressed formats, things like incremental file reading, things like pathological regular expression performance. These have been around a long time, it's embarrassing we fell victim to them because they've been known so long but take a lesson from us don't repeat our mistakes. And if somebody suggests to you that - I don't know if anyone can see this - that you should support everything that's legal in an email address. That is a regular expression that supports everything that is legal in an email address. It's from a pearl module and the author says and this is a quote "the grammar described in RFC 822 is surprisingly complex implementing validation with regular expressions somewhat pushes the limits of what it is sensible to do with regular expressions" {laughter}.

One final group. Here is a quote: "values of cookies we've set can be trusted. Admin users can be trusted with a bit of the look up API." used to be able to do this in the admin you could add to the query string look up arguments and it would apply those to the change list.

"We can trust the browser same origin sand box"

We can rust admin users with the history log"

"Once we've validated a value and stored it we can trust it"

Everybody knows what's coming here, right?

Where the crossed side scripting vulnerability because we trusted our own cross side token coming back as a cookie displayed unescaped in the page now we escape it we had information leak in the admin because of those lower RM query words you could put in the query string. This one was fun. It turns out if you send http 307 redirect who knows what status code 307 does and was not a member of the core team when this vulnerability happened?

307 is a temporary redirect preserving the request method combination of that and some problems in flash allowed you to do arbitrary headers on a cross domain request. Turns out for a while there was not in fact a browser same origin sand box in multiple browsers.

We had information leak from the admin history log. Turned out we had insufficient permission checks, we were trusting if you had access to admin it would be OK to see the history of what happened in the admin, turned out it wasn't. We had a cross site scripting from URL field, turned out it wasn't escape ping values when it showed value of the field in the admin so if you could get a URL field to accept a value that had Java script in it that might execute you could cause Java scripting attack against admin.

There is one more in the group, it gets its own standalone slide. We can trust the http host header now.

Before I go on. Hands raised who thinks this caused one vulnerability? Who thinks it caused 2? Who thinks it caused 3? Who thinks it caused 4? Who thinks it caused 5? Do I hear 6? {Laughter} so, the answer is: 1, 2, 3, 4 and 5 times. We have had to deal with the http host header.

The basic problem here is Django tries at certain points to construct a fully qualified absolute URL for certain types of redirect and links it needs to generate which means it needs to know the domain name your site is running on which normally would come from the host header except you should never ever, ever, ever trust the host header {laughter}.

This one, first one was the X forwarded host header was our attempt to solve it recommended it how to configure. Turns out you could get around that. Second attempt involved get host which was our function for par sing the host header did not correctly parse out user names and pass words which meant you could hide the domain. Third attempt we restricted the character ranges allowed in the host header and then finally we had 2 more at the end going to what Django now knows which is requires an explicit white list of allowed host header values and if you get a request that doesn't match it Django header is out. Please do not trust the host header. Please do not trust anything. All of these issues, all of these vulnerabilities that huge pile of stuff came from us trusting things we should not have trusted.

I started the talk with a game of thrones reference and this is obviously also one.

Personally though I prefer may be because I grew up in the eighties and nineties I prefer slightly different TV series approach to this - this should be your philosophy. Trust no one. Trust nothing. Don't trust your users. Also don't trust your web server, it will lie to you. Don't trust your database especially if you are using my SQL. Never just trust anything. Always verify things before you do anything with them. We could have saved ourselves so much headache, so much trouble, so many vulnerabilities if we had taken this to heart earlier.

Now we do a bit better. We validate some things before we check them. We use white lists rather than black list's there is a great article out there, I believe it is called "default allow is the dumbst idea in the history of security" if - always use the white list always specify what you will allow what you will trust. Take these lessons from us because we did learn the hard way.

Remember there is no such thing as secure. I can talk about how Django does things for security. I can't talk about how Django is secure I can't make that claim. But remember why this happened. So many of these were well intentioned. So many of these were because our good intentions were combined with our ignorance or in some cases just our complete not paying attention to what was going on. Remember why we fall. Answer is so I can learn to pick ourselves up. If nothing else, learn to pick yourself up. We have that to pick ourselves up off the floor 48 times since 2006. We are going to have to pick ourselves up off the floor more in the future I guarantee it.

This is the only may we can do it. We are never going to be perfect, we are never going to manage to lock it down completely, all we can do is do our best and learn to pick ourselves up.

And if you want to learn more about security there is what was covered in this talk and there is the actual topic. It's far too large a topic. There was a talk yesterday about security with some good information, there are other good talks and resources floating around, I'm obliged to remind you don't implement your own cryptography for a whole lot of reasons, other than that, go forth, code boldly remember to pick yourselves up, it's what everybody ends up doing.

And we have any time for questions. {Applause}.

VINCE: Thank you very much, James. If anyone who has any questions would like to make their way to the microphone we have time for a few so that would be great.

NEW SPEAKER: So, you talked a lot about denial of service which is indeed something that is very easy to trigger in Python just because it's so easy to allocate lots of memory. It seems to me that we are trying to fix the exponential cases whereas {inaudible} requests consume huge amount of memory but we're probably still vulnerable to the linear cases for instance if you send a pass request that contains tons of very small variables I think there is almost no way we can find sensible limit for example load find {inaudible} services open issue in tracker. So my question is in the long run do you think we have to accept that cutting in Python are going to stay vulnerable to data mind attacker or is there hope to have security of service in general?

JAMES BENNETT: I don't even think its Python, I think as long as we're coding we're going to be vulnerable. There are all sorts of denial of service attacks out there, things like slow lorries that are just incredibly difficult to defend against unless you have lots of infrastructure like the big CVNs(?) do to defend against. Python has its own quirks and issues and it's good to be aware of them but I don't think there is ever going to be total security in Python or any other language, we're going to have to learn to live with that.

NEW SPEAKER: Sorry I missed the beginning of your talk, I hope you didn't cover this. How do you go around finding these vulnerabilities, is there a process attached to it?

JAMES BENNETT: How do we find issues in Django itself? We do look at it and we found, we've had several vulnerabilities found by members of the core team, we also encourage people to report and in fact of the 49 vulnerabilities we've had that we've known about, most of them were reported to us by third parties who found them and contacted us and said hey I think I've found a security bug. There are people out there who do that I think now Russ are we participating in a security programme at this point? We're also now participating in security bug programme that pays out to people who find and responsibly report security issues to us.

FROM THE FLOOR: Thank you.

FROM THE FLOOR: Earlier on you say 48 vulnerabilities, now you say 49 is that because there is one in process right now.

JAMES BENNETT: I misread, yes, there are 49.

DANIELE PROCIDA: As someone who had the experience of presenting something and then have people come up to say, we need to point out the security vulnerabilities the thing you just demonstrated, what advice do you have for people who, who don't necessarily have the experience or knowledge to know that they are possibly doing something wrong, who don't know about the unknown unknowns?

JAMES BENNETT: Well, if you know, tell me.

Yes. I don't know there is a good universal answer. Educating about security and educating about even common security problems is at this point not really part of a lot of coding curricular, and I think that is a problem. There are a lot of really wonderful really smart, really dedicated people out there who write about software security. We have one guy actually on the core team, we have Paul who does security for a living and helps us out. I would really think a good project if you want to take one on is compiling a list of this sort of thing, compiling documentation, examples and trying to build it into introductory coding curricular, you don't have to get everything right out of the box, give people the vocabulary and they will feel a lot more confident going forward.

FROM THE FLOOR: Hello, thank you for your talk. You mentioned the, the top 10 and one thing that I have seen going on again and again is that the Django's implementation of CRF protection, does not fit the exact model recommended by wasp organisers if it was recommended a few years ago, and that makes security audits flag it falsely again and again. Request you comment on that,.

JAMES BENNETT: I don't know why it was chosen to be the way it is, I know people who I trust and I know I just said don't trust anyone. People I trust have signed off on it. I don't know whether it is worth rewriting, rewiring it to fit what o wasp says is their preferred way.

We set a cookie with a token in it and we send the same token out in hidden fields and forms and we look to see we get them back and they match.

Yes, that is a topic that maybe explored as to whether it is worth replacing with a more standard implementation, but at this point I don't know if it is worth the effort to rewrite it.

FROM THE FLOOR: Thank you.

VINCE: Thank you, if there are no more questions (APPLAUSE).

JAMES BENNETT: Thank you.

VINCE: Okay if next speaker is David who will be talking about views.

(APPLAUSE).

DAVID WINTERBOTTOM: (APPLAUSE) good morning hello, so I am David. I am going to talk about views.

So, views are one of the fundamental building blocks of Django projects meet them on part three of the tutorial on poles. Simple, at route it is a function or a class that can be turned into a function, takes a request object and maybe a return from a url and turns it into response. From years of code review, Django projects I find the views are where the wild things live. The parts of Django forms and models and that kind of thing, stronger conventions about how you build things and what you put in there. What belongs in the components that is less true of views, so much more freedom there. I found they become like a junk draw of project, people shove all kinds of things that don't belong in the there.

This is to talk through three different things I will start by exploring more about where views fit into the general structure of Django projects and software architecture in general, then explore the thing that is go wrong and maybe some things we can do to try and avoid the temptations and the things that lead us astray in our views modules.

Okay.

So let's talk about software architecture and design patterns and that kind of thing. The first kind of principle that is relevant here is the, something that was coined by (NAME - INAUDIBLE) in the 70's, the separation of concerns. This is a design principle that says, in a code base you know, you want to separate into sections into kind of layers if you like, only concerned with one thing, you can do that by encapsulating functionality, by hiding information from the other layers and defining a clean interface for how the layers talk to each other.

This is a broad concept can be applied to large scale, to pollings, in the small scale, function or a class, it is related acronym, the SRP, the single ... principle, taking this into the small.

Anyway, so, if you have a code base which is, which obeys, separation of concerns principle, said to be a modular you know, you have got the separate modules that encapsulate information and talk to each other through a clean interface.

So a design pattern that is relevant to web frameworks that is a derivative effect of SOC is the well-known MVC pattern, another thing that originated in the 70's, from around graphical user interface programming, since then adopted by the various web frameworks each of which interpreted differently and it is evolving even now, so MVC and the web framework, has been on the server side.

Now with the web apps, seeing MVC, parts of MVC functionality moving into the browser.

To recap how this works, the three layers we have identified, the model layer, that is where data is managed where business logic is enforced, rules and variants and that kind of thing.

I was confused by the word business logic, the word confused. I ran a project about games, I am not doing business. I was wrong, really just the rules and of your particular, of your particular project. I prefer the way domain logic, as oppose to business logic.

That is the kind of model. The view layer is what is presented to the end user, when this is originated that is the graphical interface you will see on the screen. In the world of web frameworks, the html for the user.

The controller layer, this is where there is always confusion, people talk about it being a traffic cop. The role to interpret requests from the user and then take a call from the model. To give that information to the views, so it can be rendered to the organisers shown to the user.

So this is a diagram in Wikipedia, MVC, walk through it, user at the bottom. Here we see the user is talking to the controller in some way, the controller is talking ... manipulating the model, asking it to do something. Then the model is giving to the view, so the view can update itself.

I find it easier to think of this, sequence diagram. In the top left, the person is making the request, the controller handles the request, routes it to a function, then the function will make a call into the model layer, get information and then it will pass it off to a view to turn that information, to give back to delight the user at the end of this flow.

Okay, Django has its own interpretation of MVC, called MTV, this is certainly a cause of confusion to people new to Django, uses view in a different context, there is a page on the Django's FAQ, explaining it.

Now the template is what used to be called that view in the MVC layer, when you had data ... in Django, that is templates and statics now in this case the view is part of the controller layer. If I can expand that further here, if we break down the MVC components in Django's terminology, start with the view layer that is simple. The templates and statics you normally see. Then the model layer, things like models and forms and other things as I get on to in a minute.

The controller layer is more interesting, broken down into several components, the basic http handling, then you have got url, whether that is mapped.

Then your request has to go through a stack and rerouted and this kind of thing.

Finally the views, the focus of this talk. So, from this, I would say that a single call is really an adapter, translates requests in the language of the transport, the http in to the language of the domain and then on the way back out, translate the language of the domain, back to the language of the transport, http and it is simple. It does this by taking data and giving it to presentation layer that knows how to turn the data into something that the user wants to see.

Okay. So, why does it all go so wrong then?

Well. One thing that often strikes me when I look at others people’s views or my own, they are always too big. There is an import, below the fold of the first page, all kinds imported to there, write to the file system, make network calls then there is all the blurring of the layers you know, this separation of concerns has been violated in lots of different ways.

That really happens in, something you can think of pollution see the layers reaching into the view layer, they get attached from several directions. You can see, a kind of business logic inside your views, where you might do validation, applying the rules of your particular problem in there and it is not what is supposed to be there. Persistence logic, database transactions or effectively writing SQL queries inside of views, you can see pollution from the presentation layer as well, less pernicious than the one from the model layer, but you can see logic and functionality, it is about manipulating data how it is displayed to the user, something can be done through the template tag system. Cluttering up your view, making it hard to understand and you are not encapsulating the things you could be.

So, everyone here is a perfectionist with a dead line as we know, I think sometimes what Django does to make you so productive kind of blurs the boundaries between some of the layers between the software engineering, where we see this? The ORM, the active record design pattern where you have a model, a wrap around a database drone., the layers they mix business logic with persistence, sailing out to some storage back end.

Then things like model forms and even kind of class based generic views, a kind of mixing, very directly accessing your model which is the proxy on to the database, into a layer which is further down.

This is fine most of time, enormous productivity saver, this is getting close to blurring the responsibilities and you can kind of lead you down the wrong path. Especially as the Django application grows over time. As I will come to, important to appreciate where the boundaries are, and know when you have gone too far down a path and it is time to split them out.

One thing that is fortunate is that you know, Django using a file system convention, it has a module, has a similar name to model and ... leads people down to the path, the model is where you put your business logic and all your rules live there. This is also kind of people are led towards this from this idea that you move away from fat views or fat controllers and thin models to thin controllers and fat models but really you are taking it from one layer and bringing it to another one.

You see this with large models which have nothing, do all kinds of things unrelated to their own data and fields. You can see models sending e-mails within their methods and, this is just a general confusion sometimes about what should live in this particular layer.

If I designed the web framework to send people in the right direction, I would have a domain package, this is the whole layer, this is not just your models, your forms are a form of business logic and the validation rules and also advocate using services and just other modules which aren't dictated by Django, to break out your business logic into different structures and not try and force everything into your models.

But services is a bit of an overloaded word, there is SOA and microservices these days, various ways you can think of a service. One where it is totally separate, where you talk to it (INAUDIBLE) all the service within the context of your project, this is something that comes from a book ... once people have grasped that, okay break out of models and put them in services, the next level of confusion sometimes, this is a real live snippet of code from the code base I have worked on. Service used in four different ways, you see this java brain thing, people believe they are too good for functions, everything is put in a class of static methods.

These classes use strange name spaces, you should prefer simplicity in general, functions in a module, refer to name spacing, using class as a name spacing and bundling methods in there.

I am trying to say, make use of all you have in Python and object oriented programming to solve your domain layer, don't force everything into active model reports, this is a place of freedom. This is the fun part I would say of writing a Django application, modelling a domain you are working with and capturing it within Python code.

So let's highlight a few smells, as you mature you get a sense of taste where you kind of, when you recode, you don't have to think hard about it, things scream at you, this is wrong, and this is not quite right. A couple of things worth highlighting, definitely a smell if you are requesting out of the view layer, into the domain, that is blurring the boundary of your transport, your http request, that should be convert intoed the language of your domain, so, e-commerce for instance you get a http post saying I want to add a product to my basket, view layer, here is the basket, here is the product, I want to add to the basket. There is no more reference to sessions.

Similarly with the response as well, sometimes the response passed into the domain layer, so start writing CSV data to it.

Forms, where you should have validation logic ... (INAUDIBLE), there is a definite smell if you see it in the views model.

An interesting one, Tom Christie, on the dab app's web, manipulating your saving models the rule of thumb, never call save or manipulating a models field from a views model. As far as to say, from anywhere within the actual model itself.

Even if you just want to change one field on your model it's better to write a function which set this field then call save rather than do it directly and this is a really nice way of enforcing this clean interface between the layers and it encapsulates the functionality within your model layer. If your view dot py is creating a model then creating save - you should look into moving into functionality in the model layer and as probably anything that is this basic idea of translating http into doming layer if you see sending emails or accessing the network anything like that you should look hard at that and think can I move that into a different part of my application?

So, any talk about views, I then talk about generic class based views a bit more.

So I find - again these can be enormously misused. Great productivity safer. Django builds on the ORM fantastically with admin model forms and generic class based views to give you enormous time safer S but you have to be careful. It's a path you go down as you use generic path based view, it's complicated and you're no longer created one model, several different forms different things listed. You've got to know on this path is going to take you to a dangerous place, you have to know when to stop and retreat to the safety of a plain form view or template view. I've seen horrendous things done in detailed views and create views, to bend it to solve the problem at hand when really it's not helping you after a while. And then people, lots of people have a strange aversion to generic class based views and you can find once it gets complicated the flow of execution is hopping between the subclass and parent class they become hard to follow. So the main message knows when to stop to retreat back to the views, you control the logic and the execution flow isn't constantly jumping between the 2 different layers.

So what if you take one message away from this talk it should be views should be boring, should be dull, shouldn't be doing anything interesting, they're adaptors, they take one language turn it into another and the same on the way back the dough ming layer is where the party is at, that's where you should be having fun and using powers of programming model of domain and capturing it in an elegant way that's where you want to spend your time. That's a rule of thumb.

Before you commit something you can look at your views and say have I got domain logic persistence logic - that should be a smell. But a better way is to think if I wanted expose this same functionality via a rest API what would I have to change and if you've got any logic in your view needed for the functionality you'll see it there. If you can put your mind into if I was to swap this out to something different rest P/A+ I or management command then this mental exercise will rapidly highlight where you've made assumptions and pored functionality down into the view layer that doesn't belong there.

A quick example. So here is an example view. I have tried to keep it non-trivial to illustrate something interesting. This is a basic create view, similar to what I do in day job, I work at {inaudible} wallet where we use Django to write web-sites but don't tend to use it on database talk rest API in a different language to provide functionality.

The detail. At the top importing things in domain layer. I have a preference for importing modules instead of objects, more concise simpler. {Inaudible} test in a second. This is really where I see the link interface between my view layer and doming layer. I want it simple. Don't want to see hundreds of things being imported from that layer.

If we jump down to the actual generic view class subclass of a form view, then override dispatch do basic access control, ACL model my access controls, no logic here other than calling into this layer, of course can be done with mixer or decorator, included it here for simplicity.

This is interesting, very much my personal preference. I don't think you should constantly suffix things with view or form which you can already get from the module it's in its repeating yourself unnecessarily. Some people don't like this fair enough because you often end up with the same class name in different modules but really it's the responsibility of your importing class to rename things if they want to if you have name classes should have port view at the end if you want to. I find this is simpler more readable way of doing things. I've written lots of code bases over the years and none of them have exploded from using this - or not using this suffix. So - one minute I'll be quick ...

So one quick thing on the - oh here we see sorry core straight into doming layer. Not doing anything interesting. Calling the API. Handling exceptions that might come out of it. This is what you see in code handling exceptions that might come from your doming layer. You want to be careful of exceptions jumping boundaries. If you call into network from requests in doming layer you often have different exceptions, time out or SSL errors, they shouldn't percolate back to view layer. Turn it into dome layer perception which might be available - is that 15 minutes or 20 minutes? ... Oh right OK ... I'll jump strait to the end then. So my message is be careful about keeping things separate. Concentrate on having clean views. Use this mentally swapping the layers technique to test your assumptions. We talked about some of the rules of thumb. Think about domain layer - models modules, be aware of fat model syndrome. And I want to leave you with - this is 8 guidelines Evan Williams set out when he was CEO of Twitter. They're fantastic, I like number 5 taking an artful approach. While I can say don't do this or that writing views is a subjective area and there are times when you have to break the rules and times when it's difficult to find the right interface and right separation between the layers. I think you always have to be aware of the guidelines and rules of thumb but you still have to think carefully and not blindly follow rules when you are writing your own software.

OK I think that's enough for now. Thank you. {Applause}.

VINCE: In the interests of time I'm sure there are lots of interesting questions but we'll let you find David during the break for those. Let's thank David one more time {applause} quick swap talk amongst yourselves ... OK ready to go so our next speaker is Rae who will be talking about hypothesis which is a testing framework {applause}.

RAE KNOWLER: Hi. OK good you can hear me all right. I am one of these first time speakers that Daniele mentioned earlier. Firstly I'm really happy to be here and secondly I'm incredibly nervous to be up here.

OK. Well, I'll get started. I'm going to talk to you about hypothesis which is a library not a framework for randomised testing. Also known as fuzz testing and it also does property based testing which is basically a kind of testing where you say: this property has to hold, then you test whether it does or not. I have slides on this. Let's go.

Quick introduction to me. My name is Rae Knowler, I am mostly work professionally with CKA in which is framework written in Python and simfony which is PHP but I use Django for everything I do at home. That is my twitter handle up there just in case you want to tweet how badly I'm doing during the presentation and a quick note I use gender neutral pronouns so they them not she or he. {Applause}.

Thank you.

Finally I work for L//P in Zurich, a really nice web agency and they have paid for me to get here and my accommodation which is awesome.

So, this is David MacIver who I think is at this time the creator and only contributor to hypothesis. He used to be my housemate so I have seen how much work has gone into this library and really it's a lot. That is his twitter handle.

And this is sinister cats who is the mascot of hypothesis because he is very fuzzy and loves catching bugs. He's also on twitter.

So, what is hypothesis? I started talking to you about it earlier. It's property based testing library written in Python obviously it uses randomised testing to really get in all of the corners of what could go wrong with your code.

Who here writes Haskell? A couple of hands. That's great. I don't write any Haskell and you don't need to this project started out as a port of quick check from Haskell and it's now more inspired by.

So basically it is taken all of the really cool ideas, added a bunch more and you don't need to know any scary Haskell. If you want to that's fine but you don't have to.

And that is the URL, obviously all of these slides will be up on-line later for the documentation which is the main web-site. There is lots in there. I encourage you to go read it if I wake your interest.

Now why should I wake your interest in this talk? Why is hypothesis so great? We all test, right so you are already testing you don't need any more help? So the first thing I've already mentioned a couple more times is hypothesis will generate you lots of randomised data. You know in a regular kind of test you often have fixtures or you say give me a customer model and the name is Rae and the age is 30 and the address is Switzerland and then you run some tests and, yeah OK it works great. But you wrote the code. You know may be consciously may be unconsciously, which is obviously worse, where it's going to fall down, and you've taught yourself to avoid all of those spots where things are just going to break.

So what hypothesis does you tell it: we're going to make a customer, me want the name to be a string, the age to be an integer and we want the address to be an email address or the physical address to be a SYS. Then it will come up with the default is 2000 but you can make this bigger or smaller, it will come up with 2000 customers with all kinds of strings for names and integers for ages and so on and if there are inputs that are going to break your code it will find them.

So that is awesome.

Then we get into more of the clever stuff. It simplifies examples if we find out your code breaks with this kind of customer, it will then change it around and hammer on the weak points until we go from you need supercalifragilisticexpialidocious plus a dollar sign to break down to this if you have a dollar sign in your string it breaks. And once it has got that simplified example which will show you more closely where the bugs are it will remember it for you. It makes its own database and saves these minimised examples of bugs so when you run these tests again it can jump through all of these examples it already knows have broken past versions of your code just to check if these things are still OK.

And then this plays into what I'm going to get to with the rest of this talk. Hypothesis is quite clever and can understand and deal with complex data types. So, one example here obviously is the Django models like I just said, we know we have customer with a string for an address and a string for a name and maybe a foreign key, relationship to all the things that they've bought, and so on, and hypothesis can accept and understand this kind of complex data structure any way, but it also has an extra library hypothesis Django which is good for us because it understands about Django models and you can do some other stuff with it too.

Oh yeah and the final thing before I move on is it works with py.test, with unit test, it works with everything because it's just a library, it just returns Python functions. So if you saw Mathieus great talk about switching to py.test and you thought yeah I'm going to do it, you can also do hypothesis. You can have everything.

Hypothesis and Django, why is this relevant to us?

I already said that hypothesis Django will understand your Django models and you can write tests. I'm going to show a demo of writing simple tests in a minute. You can write tests that look very similar to your Django tests. You can also use it to test customer field types. And you can also use it to generate child models. Say I have a customer and this customer has a bunch of products, then I can set up hypothesis in my testing to not just tell it I want a customer, but tell it I want a customer and this customer has a bunch of child models and it will generate those as well. You have to set it up separately but it's quite simple to do.

The last thing is you can also use hypothesis to generate your fixtures. If you are really set on having fixtures for some reason, it says here this is new, it's so new that on Sunday David called me up and said have you done your presentation yet? Do you want to talk about fixtures? I said I'm sure I can fit it in but actually I'm just mentioning it. That's all you get. You can look it up.

OK. I've woken your interest. How do we get started with hypothesis. It's simple. It's on py py, it's on yeah just py py may be other things as well. All you have to do is start up a virtual run and do pip install hypothesis, pip install hypothesis Django, that's it. Now I'm going to do a demo.

I considered doing a live demo for this but considering I've eaten up some of my time with battery trouble I'm going to go through my screen shots. Hope that is OK. Yes, yes OK screen shots we have them! {Laughter}.

OKI hope you can read this if not I'm going to go through it quickly. I made up an incredibly simple test repo for this and I am, am going to post the link to have - repo if you are interested later on, basically this is my model's py. I am importing models and importing date time and this example is like a web-site that can subscribe to various docs and say you are really interested in the stock market, you have lots of stocks, you want to be notified during the day on how your stocks are doing. So I have made a stock subscription model. And this has a symbol and it tells you it has a last notified property and it has a notifications per day property. I want to be told 10 times per day, 12 times per day how my stock is doing.

I should just interject here that it turns out it is really hard to come up deliberately with bug's like really hard. I can do it without thinking when I'm trying to make the code work, but - so these are quite simple examples because it's hard. Any way ...

So, yep, so I have the string of this model, which is just the symbol and how many notifications per day. Then I have his property next notification time which calculates when I will next be notified about this stock. OK that's quite simple. And here are my tests my test dot py.

OK you can see that I am importing test case, not the regular Django test case, but this is from hypothesis, and I am also importing models from hypothesis, and given, which is the really important thing. This is the main thing I want you to pick up on here. And I have a test case. And what I want to do - assert is that the next notification should always be after the last notification. There is no point saying I want to be notified about this stock last Tuesday, next Tuesday is OK. So I write the test quite simple and I have here this given decorator which tells hypothesis what type it wants the input to the test to be so here I'm calling this function on self and subscription and subscription wants to be a stock subscription model and it has property notifications per day which should be an integer and this integer is a strategy from hypothesis. Yes, don't need to go too much into that just that it wants it to be an integer.

OK so everything should be fine, I run my tests; durr ...! Hypothesis has obviously found an error. So you can see it creates a test database and it runs the test, this is going slightly off the slides but I hope you can still see it. It finds a falsifying example. For this function. And we can see that the subscription that it found that is a falsifying example for my test is name is an empty string, don't have to worry about that and the notifications are zero per day.

Obviously this causes me an error because in my calculating the next notification I assume I have zero notifications per day and I'm dividing 24 by that and if that's zero that is division by zero error obviously.

How long have I got? 5 minutes OK.

So, I will race through my second example which is not too complicated either, this was supposed to be a complicated one but like I said it is hard.

OK we fixed that problem. And me now want the portfolio in our web-site. So I have 6 stocks and I want to know through the day how they're all doing. In fact, I don't just want to know how they're doing individually. I want to know what the median value of my stocks is. And I was quite surprised to find that in Python 2.7 which is what I'm still using there isn't a built in median function so I worked up this median function just to calculate that. And my {inaudible} has this function median value which returns it.

Have another test here. And you can see we've learned from before. I'm using another strategy from hypothesis's lists because this portfolio is going to have a list of my stocks. And so here we have - I am testing portfolio and subscriptions and testing it to give it model portfolio and list of stock subscriptions and yep they have value which is minimum 1 and maximum like a thousand I think. Kind of went off the edge but that's not really important.

Oh yeah, up to 100.

And I just want to assert in my test that the portfolio median value is between zero and a hundred. If all of my subscriptions have values between 1 and 100 then obviously the median should be in-between those 2 numbers.

Hypothesis again finds an error here, because it turns out that when you have this so. It turns out, when you have this, it tells me I have an error, has an example. A portfolio that has this list of subscriptions in.

If you, if you were to read a bit more, it gives you a stack trace, I can tell that the problem here isn't necessarily that these subscriptions don't have symbols or that they have large negative notifications per day, although this is an example where it is simplified the answer down. You don't need any, any strings in the symbols to falsify this, you can just give it the lowest possible number which is its way of saying yes, this low number is enough to trigger the problem.

The problem as I can see from the full stack trace is that I have two subscriptions in this list. If I go back to probably some of you have already seen this, there is an error in my calculation of the median. I think one of these pluses here should be a negative. This means that when you have exactly two, exactly two subscriptions in your list, you again I think gets a division by zero error or something similar.

Okay. So, these are quite simple bugs, I probably would have been able to spot them myself but hypothesis made it quite simple. And, it will also catch a lot more subtle bugs that were a bit too complicated for me to put together for this presentation.

But what I want you to get away from this is hypothesis is awesome, that is point number one, it makes writing your own tests way simpler, I hate coming up with fixtures I hate sort of coming up with things to get exactly at the right education, you don't have to do that with hypothesis, tell it, I want one of these and these, it will sniff out all the bugs for you and remember them. Which in my experience makes things a lot simpler.

Then you don't have a trade off with hypothesis you get simple tests to write that insanely effective. So, I think it is great.

I think you should try it out and finally here are some links to the documentation to my little example there. Again, my twitter handle and my website.

So, if everyone is sort of taking pictures of that, these slides will go up on-line. Thank you to all of you for listening and for Djangocon for having me.

(APPLAUSE).

VINCE: Okay we have time for questions, if you want to make your way to the microphone.

FROM THE FLOOR: Thank you, very nice talk and it looks awesome. I know quick check have ported to, the languages, so it is awesome we have it for Python. You said at the beginning it will find interesting fixtures or values that are breaking your test that it will use later. Is there a way to export those or save them in your repository for future contributors?

RAE KNOWLER: Huh that is a really good question, it is one I don't know the answer to. Probably, and if not, I will tell David that he should do that because that would, that would obviously be really useful thank you.

FROM THE FLOOR: Thanks.

FROM THE FLOOR: Thank you for a great talk. Django 1.8 chips with the set up test data, functions on the test case which basically serialise the data and reads them for every single test function, is there any way to use that with hypothesis Django directly, so you don't have to do specific test methods.

RAE KNOWLER: I will say again that is a great question, I don't know the answer to I am sorry. I don't think so at this time. Django support is quite new, hypothesis has only been usable, but out in the stable one point naught since sometime this year, active development is on-going, I think there are more things planned for the Django integration.

FROM THE FLOOR: Hen you said in the beginning that you using passing basically when does this process stop, so if you have written code that is has no errors, will it run indefinitely, what conditions are to break up the loop?

RAE KNOWLER: You can tell it how many examples to generate. And, so I think the default is 2,000 but you can set it to like 10,000 or a hundred thousand if you are really paranoid. Yes. That is it.

FROM THE FLOOR: Thanks.

FROM THE FLOOR: Thank you for your talk, does it use non-Latin characters for the user fields?

RAE KNOWLER: I think it could yes.

Okay? That is it I guess, thank you again.

(APPLAUSE).

VINCE: Okay just.

DANIELE PROCIDA: I have move it had appointments board with the post it notes it is in the foyer, just to make it easier, just on the left as you go out of these doors after the first desk. Okay that is the appointments board for the counselling sessions.

VINCE: Okay a pleasure to introduce Owen, I have met him because I taught his son and then since we have been working on some code together, I secretly believe because he wants to make sure I am teaching his son correctly. Owen will take us through his aging code.

OWEN CAMPBELL: Good morning everybody (APPLAUSE).

I was delighted when I got the slot for this talk to see that it was the one just before everybody's dying for a coffee. I am cob tent to use that as an excuse to go nowhere near a presentation or a single line of code I written on the project.

I am the aging coder of the talk. So, there is the outline of what I would like to take you through, I will tell you a bit of my background, where the aging code came from and why I needed to do something about it. No surprise, the solution was a Django application, I will tell you a little bit about why I came to that conclusion and what I found along the way, when I was attempting to make that work.

So, the aging coder. Me.

I am, I put the title in for this talk because for the first time in my life at the tender age of 46 when I signed up for this conference I qualified for special treatment, early bird registration was available to me, I have never qualified for anything on account of old age! until I signed up to do this, so I sent them the title for my talk.

This probably sums up my coding experience. Back to 1980, I am 11 years of age, I just passed my 11 plus exam and my grandfather brought me a Sinclair ZX 81, the height of technology at the time. A massive 1,024 bites of memory, no way for me to ever store anything I written and the only way of doing anything was it was to programme in it in its Assembly language, then I am, 11 with the almighty machine to play with, if it had no storage, you will recognise these. That, it took some finding by the way, that is actually the tape recorder that I had. I had to explain to my 15 year old, the thing on the right goes in the thing on the left and it is called a cassette tape.

That machine has led me to here. In Assembly language I managed to programme that thing to draw and I was going the get a nice gift to show you what I drew, I think we can do better than that, I managed to draw 1980's icon, not only that, I think to this day, the greatest programming challenge, I managed to make the icon do its dance. Rather than give you an animation, I will do it for you. I managed to make this thing do this:

(APPLAUSE).

Yes, by poking binary values into memory locations I got black squares on a screen and I got a space invader to move from left to right.

Anything that can give me that amount of joy had to be something I do for a living and persuade people to pay me for it.

So I moved on Secondary School, High School, this colossus of a machine, 32 k of memory, nowhere to store anything, I was introduced to a high level language known as Basic.

I was once set some homework by my teacher, what I want you to go away and do, to draw random squares on the screen. Great I thought, I get home and, what exactly does that mean? What do random squares mean?

Yes, I will make it up. So I drew random sizes at random locations, in random colours and at random orientations, I brought this piece of work back the following day. Graded a C minus.

The teacher said to me, I have marked you right down because very few of those are squares, they are nearly all diamonds!

Valuable lesson, if the requirement are a bit vague, don't just run away and start programming, you probably not going the give him what he was expecting. Another valuable lesson that I seem to forget from time to time, it ought to stay with me for the rest of my days.

Off I go the university, I am studying an engineering degree, I am really looking forward to do a bit more computing at this esteemed establishment that I joined. What do they teach me to do? Where in the first year, Assembly language, second year, the motor row la, Assembly language programming.

Fortran77 I was told a high level language. What that meant was, you got get your programme submitted to the university mainframe by 4:00 o'clock in the afternoon or forget it. When you come back in the morning, you will have your executable ready to run. No you won't, you will have your report from the mainframe saying, missed a comma, get it back in by four this afternoon!

So what have I done since? Worked for various organisations and I have worked for myself for the last 15 years and over the years, I have been exposed to this stuff! this is a list of technologies where I have worked on projects that have delivered something using that technology. Most of it have to say, not at my choice. I seem to have had the misfortune of joining projects that were already using some piece of technology and I have had to learn it and get on with it. That is fairly typical for someone who has been around for the length of time I have now.

You will see Python and Django is on that list, probably stretching the truth a built far, that was one project where I had to get to grips with it because that is what they were doing, I told them they needed to cancel the project it was entirely the wrong thing. I thought I would put it on here anyway.

There is no pattern to that list or great career plan, that is just stuff I have ended up working with over the years.

So, what is this aging code I am talking about?

Well, it was an accounting system that came out of a project, a research and development project done in the mid 19 nighties, a series of projects looking at database design, needed a couple of applications to demonstrate whether the designs worked, one of those was an accounting system, it was written in lotus approach. It was a project I joined half way through, it was.

I needed when I set up my own company an accounting system, I thought, oh I know where there is one. Since then moved on to Microsoft access which again I still wasn't keen on. But quickly picked it up and convert it into a SQL server database and a dot.client sitting on top of it. I got up and running, I was going the say I had one that was steal but it was funded by the European Union, it is available to anyone.

A system I could take and get myself up and running quickly, that has been running for something like 15 years quite happily.

But I needed to do something with it. First and foremost, that SQL server licence I really didn't like paying for. I was every time the renewal came up, I would sit and think, I need to do something about it this, I don't have the time, I will pay them the money and away we go.

What I really hated was I was now having to maintain a windows development environment purely to deal with this system. So, even though I never made any changes to it. The fact that I might need to, meant that I was having to maintain a completely separate environment. Also meant I was having to run a windows virtual machine on my lap top just so that I could use this thing.

I had had enough of that. I finished a large project didn't fancy doing real work for a few months and decided to do something about this.

So, the way forward. What was I faced with at the point where I sat down and decided to take this project on? Well first of all, I wanted a web application, that means using a framework, well I thought so anyway.

If you go back to my slide with all my list of things I have worked on, fair to say I have done a lot more work in Ruby and rails than I ever had on Python and Django, that is where I started. I hit a brick wall within the first morning.

I have had a lot of experience with rails, our family enjoyed working it. Realised only ever worked on projects that were starting from scratch with rails. Rails is very opinionated, that is fine on a brand new project, you can decide to go along with the opinions and everything is great. This wasn't a new project, this was a database design, originated from 94 and rails opinion of it was low. Very low.

It didn't like it at all. In one or two cases I probably agreed but I didn't feel like doing very much about it.

I found myself fighting the framework before I could make any progress at all and that felt like a rabbit hole to me!

So I made the decision to climb out of it. I started looking around. I thought well if I am not going to do rails, what am I going the do? Somebody suggested this to me. The pitch was, well you are going to end up using java script in the front end somewhere, you might as well get familiar with it and use it across the application, that is powerful argument except the word java script in it!

I suspect I am not the only one in the room, I hate it! I hate working in it!

This was, it is a purely emotional response, I just couldn't face taking this project on and working purely in java script for months on end. So without getting any further at all, I discounted that fairly quickly.

I had read a lot about the scala language and the play framework, I fancied having a play with it. I down loaded it, fired it up and read the books. The scala language itself was a thing of beauty, I enjoyed working with that, it is a much younger language had the benefits of taking the good things out of Python and Ruby and trying to make them work in scala and they have made a good job of it. The play framework had no particular problems with it, got it up and running with the database quickly, but it is young, really young. As I started to look at what I wanted to do next, I was expecting to find libraries and tools to use, because I am lazy, I found they didn't exist, the answer to the question was, I probably have to write it myself. I am too bone idle. I put this to one side, I was reluctant, I enjoyed working with those two pieces of kit. I wanted to get my project done, I didn't want to write the framework for them.

So I went back to Python and Django, I had worked with a couple of years beforehand and bit by bit managed to build this thing without hitting brick walls. If it had an opinion of my database it kept it to itself which was fine by me (LAUGHTER).

I found I could easily put the admin interface on, find various tools to make life easy for myself and the application came along quickly, so that is where I ended up.

So, how are we doing for time? Okay.

This is the structure of what I have ended up with. In my original application the SQL server database not just handling persistence, the entire domain logic was done as SQL server procedures that needed to change. So what I have managed to replace it, posted it to a database, all the logic come out of there and now handled within Django, some of it is straight Django code, as you will see in a minute. Using the REST framework as well. Seen a couple of talks, starting to suggest to me, I might want to move that somewhere else. Like I said, I am not showing you that code because you are all desperate for a coffee.

Up until yesterday afternoon I was quite nervous to do this: Because I took the decision to move the entire front end out of Django, and do it entirely separately. The main reason for that, I have a customer who uses Angular JS I knew nothing about it., this was an excuse to play with it before I had to walk in the door and look like an expert. That was the main reason for doing the front end separately. So I was a bit nervous about hitting the button here and owning up to the fact that I did that. Many thanks to Luke, in his talk, made me relaxed. He seems to think I should have done. If there is only one person in the room that agrees with me, there is one more that I thought there might have been this time yesterday.

As you can see my loathing of java script is so complete that I am one I think about five people in the world as far adds I can tell, that drives angular JS with coffee script, that comes with its own pain ... it does mean that every time you ask a question on stack overflow all you get a 1500 responses saying, if you solve this, please publish your solution somewhere.

At the moment I am serving that separately, I have the front end and the back end completely utterly separate, for no other reason I never did anything about it. I could try to serve the angular JS app within Django, I see no reason to do so, until I see one, I am bone idle I won't take that on.

If the button works? There we go.

So, just to conclude. What I have ended up with is a nice modular application, I have got a completely separate back end and a completely separate front end I can muck about with to my heart’s content. The whole lot is entirely open source, not paying any licence fees for any part of my application anymore and that makes me very happy indeed.

I have kept my original database design. That is a sort of personal pride rather than a decision I would make if I was starting from scratch. I like the fact that the database has sat there running since 1994 and it will continue to run my company until I find myself a real job and decide to do something else for a living.

It is not a space invader, not got back to being 11 years of age and quite that level of joy, but that made me happier than I was before I started this project.

That is all I have to say, thank you for your time and attention, if you have any questions, I would be delighted to take them. (APPLAUSE).

VINCE: Do we have any questions for Owen? If you do, make your way to the microphone?

FROM THE FLOOR: So, I am also using Django and angular JS in my day job, to you use them to framework to integrate them, or throw Django rest framework into ...

OWEN CAMPBELL: Entirely rest framework and the Json and the angular application is happy, has no knowledge of what is going on there at all.

VINCE: If there are no questions, I am sure people are dying for coffee, thank Owen one more time. (APPLAUSE).

(BREAK).

NEW SPEAKER: Welcome after the break. My name is Dominic and I have pleasure to introduce Anssi who is Django core developer from Finland and Anssi is going to tell us something about new features in Django ORM. {Applause}.

ANSSI KÄÄRIÄINEN: Hi, so my name is Anssi Kääriäinen and I am glad to be here in Cardiff, thanks for having me. Just a bit of talk about me at first. I've been Djangoconty a better for 5 years. Now in 1.5 and 1.6 time frame I fix pum box in the ORM and did a lot of internal refactorings and in 1.7 I wrote look ups and transforms features and in 1.8 we got expressions, it wasn't written by me but written by Joss. So these are new features in 1.7 and 1.8.

The important thing about these new features is that the re-useable component for ORM so you can inject customer SQL to the {inaudible} where is and the user change this structures together.

So I'm going to do an introductory talk about this so I'm going to do some lab coding. I'm going to try to look ups transforms and expression.

First a bit about these features what are these? look ups. These are quite friendly. We have extract and links like that look ups in Django. The example I'm going to write. This is an SQL light look up. So this is a global look up. It's similar to the like look up in SQL but this uses asterisks as {inaudible} instead of the person in time.

Next we have transforms so this is a transform that takes in a string and transforms into lower case format, so the idea is that we can do look ups to case sensitive format so to do a title lower globe Django look up.

And last we have expressions, these are a little bit different, kind of thing, these are like the aggregations we've had in Django for a long time but the expressions don't need to be aggregates, you can do any kind of expressions ORM, they don't have to be aggregates they can be regular SQL and we're going to find a group object expression, it's going to take in the names of the books authors and then concat them together, this again SQL light specific expression. SQLite.

First try not extra. Extra has some problems. First, you can use your extra SQL, hardcore some SQL to {inaudible} and if you want to use it to some other model you can do it with look P ups and transforms you can already use your existing transforms and look ups.

You can change or sorry chain these together. So if you do first an extra expression for lower casing something you can then do a look up against lower case format which as we saw in last slides you can do this easily with transforms.

Then we have multi-DB support for the new features. This doesn't exist for X {inaudible} doesn't work for some other database you are basically out of luck.

So last programme with extra. This is actually from Django's documentation. So, what this does, it A annotates 2 columns and as you can see the syntax isn't actually that nice. So, look ups and transforms have nice syntax and is going to be a lot easier to read the code when you use those.

Finally the extra implementation inside ORM is really, really hacky so look ups and transforms are a lot easier to {inaudible} more than current way extra works.

So on to the look up class. Look up is basically a condition in the query's where clause. Django has {inaudible} extra contains and so on you can look up all of these but problem is before Django 1.7 you couldn't write your own look ups. In Django 1.7 it's going to be really easy to do this and, well, in 1.6 you kind of could wrote your own look ups but the problem was you had to subclass a couple of internal classes of Django and then hope things would work and they kind of work but in many cases they didn't. In 1.7 this all is solved for you by the customer look ups feature.

So, how do you create a customer look up?

There are 4 steps to creating custom. Look up first you subclass models look up. Then you set a look up name. The look up name attribute is something Django uses to identify look up so for example global look up is identified by the globe string then implement as SQL. This is where a produce SQL parameters for the {inaudible} finally you have to register the new look up to a field or subclass of field so you can have different look ups for Java fields and data fields and so on.

Next we are going to do some live coding. OK I have a file here. Do you see this? No OK. That's going to be a problem. Do you see it now? OK. So, I can make the font a bit bigger. It's going to be hard for the line changes but -

NEW SPEAKER: Can't see -

ANSSI KÄÄRIÄINEN: I don't know how to do that live.

NEW SPEAKER: Colon colour default.

ANSSI KÄÄRIÄINEN: Sorry? ... nope ... okay let's see can we do it this way. Actually I think I can do it this way. {Applause}.

OK so we have - have some set up code and then we have model, all we need to know about model is it has a title feed. Here are some instructions for me. It's basically just the steps I showed you a moment ago. And then we have some testing code. Let's test the testing code after I have written to look up. So, we just start by writing that look up better instructions. So first - look up name ... implement as SQL ... it has compiler connection ... implementation is always going to look something like this ... so we are going to process the LHS the LHS is the title here. So, the look up is given the title expression which is the LHS side of the process and we call it the LHS method which does the magic and gives us SQL for the title column. The same thing we do for the right hand side. - Smaller OK? May be still ... yes.

So, what we have here is we are just going to process errors and {inaudible} same as for SQL method and finally we're going to write an actual SQL. ...

So we are going to use the globe look up this SQLite operator. We are going to give it the title on the left hand side and then the value on the right hand side. And then me need to add parameters and finally we have the step to register it to Charfield. This is done by annotating. Now if everything is correct we have a working new look up.

So ... let's see what happens. Of course. So what we see here we have the where, this look up implemented. This is the same SQL we wrote at the look up, if we look at the demo one py - sorry ... so, we had version S and we supply the LHS SQL to it, and the title was the left hand side and the Django string was the right hand side and ... ooh ... and when we look at the SQL we see that it is here in the same format, Django has provided us the left hand side SQL and value the right hand side.

Then we have some code here that tested that it actually does something so we create a couple of books, one is the Python book, one is the Django book then we have the look up, one book, Django book and as we see it we've got the Django book from database.

So, that is how you implement a look up. {Applause}.

OK. Our look up had one problem. It wasn't case insensitive. So we want to fix that. For that we are going to write transforms and transforms are feature that take in a value and output a transformed value. So for example we can take in a string and output it in lower case format. Of course this is happening in the SQL file not in right hand side. And when you create a transform, you pretty much the same steps as for look ups. So you subclass models, transform instance of look up and everything else is pretty much the same.

So, actually ... yes ... we have here another file, demo file, and here we have the global look up we wrote in the last part. Then we have some instructions for me and let's implement the lower transform. Again we subclass model transform, set look up name, so the look up name here lower matches what we have here, so that is the reason for the look up name. Finally we define as SQL, this is again going to be pretty easy. This time we are using a bit different we are not calling process LHS but we are just directly using the compiler to compile the LHS. The compiler - this method is just a class, all you need to know about this that it has compile method and the connection is your connection to the database, it's the currently used connection.

And then we are going to return the SQL from here which is going to be really simple. OK.

I think it's ready we need to register it. Yes we have all 4 steps then done.

OK. Let's hope it works. No ...

NEW SPEAKER: {Inaudible}.

ANSSI KÄÄRIÄINEN: Yes I have written here transforms ... sorry?

NEW SPEAKER: Register transform ...

ANSSI KÄÄRIÄINEN: No actually it's currently still registered look up I think we are going to change that so you can use register transform but for now on this hour register look up if you want to register this. So another try. Ooh still not yet.

NEW SPEAKER: Per cent sign.

ANSSI KÄÄRIÄINEN: Yeah, yeah correct. Lab coding with 400 people is really productive. Fast. ... Still? No? That's good yeah. Yes now it's working {applause}. OK. What we did here is that transform got a title column and applied the lower transform to it. And if we look what we did here in the test, we used title lower glob. So now that glob look up got a transform message LHS the glob look up doesn't need to know what it gets in LHS, it is just going to process it no matter what it is so the idea is the glob look up doesn't need to know what is left hand side but by this we get really cool feature because we can change this up as much as we want. We could apply it again here. Not too great thing to do but as we can see we can just change them up.

Finally, we have one problem here. We want to apply the transform to the right hand side too. This is done easily by marking transform by light row.

This is why you want static typing.

Okay. So, now we see that we also lower case the value on the right hand side.by this you can build look ups and transforms for example, an accent take all the strange characters out of the string and have things like that, this is actually really useful if you are going to do matching for example for names, so I don't know how many of you know how to write Kääriäinen on the keyboard, if you want to do a search on the keyboard, find an accent and put it in. They got it ooo ...

Okay.

So, this was the lower transform demo.

Expressions are a different kind of animal. These are the closest in the select part of the query.

For example, we have the average of the author’s age in Django, since 1.2 but before 1.8 you couldn't right customer expressions or if you tried you have to again sub class some internal classes and got pretty ugly pretty fast.

In 1.8 creating the expressions is really easy. There are two steps to it. Sub class models if you are not going to write an expression or models aggregate if you are going the write a models aggregate.

Then set function attribute of the new class, there are only two steps. I am going the again, try how things go here.

Now.

So, again we have some set up stuff going on and we are just going to implement the group.

This is going to be an aggregate. I am going to double check my typing here. Okay.

We are done.

Let's see if this was easy enough?

Ooo it worked. So we can check what it actually did. So, the test code we are having here is that we are, are just going the annotate group concat over author's name, this is going to have the authors of the books and the separator is going to be a comma.

Okay?

Then we test the look up is working and do an annotation and then we are just ... then print the results.

Okay? As we see, we got two authors for the bottom book and two authors for the Django book. I had some set up that created books with the. We see the results are correct.

I am going the show a bit more about where this transforms and look ups into act with expressions about how you can reduce your companies. I have here what I wrote earlier in first. So, now what I am going to do is I am going to annotate the string to the query. The concat string, then doing a filter over the lower case format of the string, with the glob I wrote also.

Then I am just going to have query. As we can see, let's check still what was, I am going the look for up for the a, m string.

As we can see, Linda Melissa has the string in lower case format here. So if everything works that should match.

Oo it worked. We also have T okay, as we say, we must match it correctly. As we can see the query isn't that simple actually. Django is doing a lot of stuff and, so we are created a couple of joins, we have a group going on and close, the look up is actually added to the queries, having close instead of the, because, as this is an aggregate we have to use the close.

So, this is one of the reasons I really like Django's ORM we don't mention anything about joins, not about left out joins we don't mention anything about group or having things that work.

So, this is all the live coding I am going to do today. (APPLAUSE).

Okay. Then some other stuff about this feature, one new feature in Django 1.8, support ... you can add the Y and do expenses and all aggregates are expressions also and by these two new features we get things like annotate, price per page, with this sum of books price, divided by sum books pages. This is something you really couldn't do in 1.7 earlier.

Another new thing in, new thing in the expressions and look ups and this things like this are that you can set an output field when the type of the value changes as a result of transformation or a recent expression you must tell Django that the value type has changed. You do this by setting the output field. In this case we have a linked function. This is in the string and calculates the length of the string. So now you can annotate the length of the name to the query and then check which objects have greater than 10 length of the name.

The output field is used both in when Django takes in parameters it must do some transformations to them so that they can be used by the, underlying back end. Also when values came out of the database, so if you had something like let's say Json field or something like that, where you need to convert the string to a format, you would do some conversations by the output field parameter. But this fields conversations functions.

By the way, we have new functions for values in 1.8 fields so before, if you had some problems with conversing values that should be a problem in, shouldn't be a problem in Django 1.8 anymore.

Multiple SQL dialects are supported too. What we can do is created a special name for example, for postgres special name is ..., if you have this method then, the ORM is going to first call a special named method and if it doesn't exist then it is called the SQL method. In this case, we have the function length which works for SQ lite but in both function is (INAUDIBLE) just telling the S SQL that we want to use the function for postgres similar you can lose sql lite then as for my sql.

Okay. Then we have some examples of new features in Django 1.8, these are unfortunately only postgres specific features, these are done by (INAUDIBLE) project.

So, H store is a field that is a key value store for postgres so you have keys and strings and values as strings. The example is that we can have dynamic data in models for example, we have dog model and breed and tricks for the first one. Then only the breed for the second one.

Now, we have a very special transform that can extract the key value by using the transform syntax, so we can check which models have collie as the breed. You can see this matches but this doesn't.

We can change this up, so we can check data, breed, contents so these breeds.

So, again, we can chain this transforms and look ups together and we can reuse what we have already created.

The H store is going the be really useful for dynamic data in already in 1.8, I think in 1.9 we are going to have Json field this means that we can actually store any dynamic data we want into the database.

So it can be ... so this is going the be powerful new feature for 1.9.

Okay. Array field is another kind of field new in Django 1.8, it allows us to store kind of like lists in the model that except every element of the list must be of the same type. So here we have tags for both. We can check which both have both Django and tags. As you can see the contents look up already access for fields but we can override it for array fields to do something else than it does for text fields.

We can also check how well it works, so this is that we check which post has at least one of these tags.

Finally, we can take the first element here, Django, out of the tags, then glob it. Then we use lower case formatted and it matches here.

So, this really, so the power of this new features, so you can build really complicated features by creating simple classes that do one thing well.

Then. What is next?

We want to implement more common expressions in Django core and contrib this means we want to have things like lower case and upper case and string length and things like that already in Django, these are things everybody uses and every database implements so why not have them in Django directly?

Then the transforms aren't yet usable in all places, only usable in the field method so we want to make it possible to use them in ... for example if you look at the ... ordering by the breed of the dog would be a really useful. So this is something we want to do and I think we could get this in for 1.9 but no promises.

Expressions and transforms are almost like each other, but unfortunately transforms were first implemented in 1.7 and then expressions in 1.8 and then we saw almost exactly like each other, so we want to unify the expressions and transform implementations we are not quite sure how to do this, but in experience we are going the check how we are able to make this work.

Then, this is a wild idea but, this would be a really nice syntax to use these transform the problems with transforms is, you can take arguments for them. They are just lower case and something like that, but if you want to take a sub string from ... you can do it with current transforms at least not easily, it is not going to look nice if you are trying to do it. If you get syntax working, I think this would be a nice feature, with this, you can do transform and submit arguments. No promises at all this we are going the get this, just an idea.

Okay. This was just an introduction, I showed how to create a look up and transform an expression, there is a lot of more to details so if you want to read more about this features, go check Django's documentation.

Okay. Thank you.

(APPLAUSE).

I want to say a little bit about fund raise for Django project. This is a Django software foundation fund raising and we have actually nowadays one full time worker for the coder for Django and this is actually the most important thing that has happened to Django, since 1.0, so please, support Django.

(APPLAUSE).

NEW SPEAKER: Thank you, we have time for a couple of questions, so if you have any, please line up in the middle of the room.

FROM THE FLOOR: So, I recently had a case where I wanted to annotate over a many to many field but by looking up the, by restricting to just certain types of the field it was connecting a user and another model and I wanted to only annotate based on particular users data. Is that currently possible or is that something I would need to wait 1.9 for?

ANSSI KÄÄRIÄINEN: We have expressions in Django 1.8, yes, it should be possible.

FROM THE FLOOR: Looking at the SQL methods it looks like kind of redundant because you probably will always have compiler process and compiler process, left hand side, right hand side in there, would it make sense to override, get attribute and pre-process that already, put it on the class that you don't have to re-implement it on your own?

ANSSI KÄÄRIÄINEN: I think we could do that. There is a balance of making things easy to use as possible. If you want to change something, do something else it is better to have those methods and so you can, so you can it more easily change your code but, I think yes, simple template based look up and template based transform would be useful that is what you are mostly going to do.

This is similar to the expressions in expressions we have actually the front end aggregate classes this are simple template based classes but there is underlying expression class which has, if you want to go and write something strange you can do that.

FROM THE FLOOR: Hi, so, this is obviously two enormous big feature that is dramatically express the power of the ORM, I was wondering what ideas you have for things we can do after. Obviously not any commitment but other crazy big ideas you have got for future Django?

ANSSI KÄÄRIÄINEN: I think the most important features are composite foreign keys then annotation of let's say custom joins. Django joins for you, but if you need to do something like left join something and field on some value on the left of the join close, you can do that currently. That is something that would be really useful. A lot of ideas but I think those two are really big ones. Some bit that is need to be fixed too, but they are not too serious but they annoy me a lot.

FROM THE FLOOR: Hello, I was wondering if there is a way we can limit the items on array field to certain values by for example, setting like a choices attribute?

ANSSI KÄÄRIÄINEN: I think you can write a validate for that. It should work, I believe, so the validate that checks the values in the field are in the place you want to use. So not a direct attribute on the array field but just write a validate.

FROM THE FLOOR: Thank you very nice talk it is nice to see that ... is getting some log, but Json, that would be a really nice extension, any plans?

ANSSI KÄÄRIÄINEN: I think it is commenting all right, Json field, yes.

FROM THE FLOOR: Oh thank you.

NEW SPEAKER: Thank you, looks like you had some favours with ... (INAUDIBLE).

Next up. We have Django developer from far away Sweden who is Ludvig is going to talk about user testing. Please a big round of applause please.

(APPLAUSE).

DANIELE PROCIDA: While Ludvig is setting up the machine, the artists that produced the art work in the conference, pick something up, if you don't, say thank you for producing this for us, they will be pleased to hear from you.

LUDVIG WADENSTEIN: I am here to talk about user testing, this will be a talk void of any Python or Django or anything like that. I hope you will find it useful anyway.

So I am from Umea in Sweden. Work at a company called Codemill. We're web developers, build stuff for others, use a lot of Django. I work primarily as a developer but do a lot of user experience stuff, assign user testing, stuff like that.

So, what is user testing then?

I find this definition quite good I make someone use your web application or web-site, you see what happens and you try to learn something.

So how many of you have ever tried doing some user testing? So how many of you have tried deploying to production? OK that's quite a lot. I would say that all of you have ever deployed something to production has actually does a user test but perhaps not at the best time. {Laughter}.

So, why should we do user testing then and perhaps earlier in production?

It gives you a fresh perspective of what your application looks like and how it works. If you work with something for a long time you can't really see how it looks for someone who hasn't seen it before what the problems with it might be. Another reason is to see if your users actually get what you are trying to accomplish, do they really understand what you are doing and why it works the way it works? And finally because you will probably find some small things to change that will make a huge difference for users.

So, when should we test? You can start testing even before you have built something. You can try to test the competition. You can test an earlier version of what you are doing. You can test with sketches and prototypes and this is to help you make sure you are solving the right problem, something your users want solving and stuff like that.

Then when you are building it, you should continue to use testing to find usability issues and improve the work flows and make sure it's easy to use. And when you are finished you can do some more testing and at this point it's more or less you only have time to polish it and make sure it is shiny so the earlier the test the better it is and it's often cheaper to change stuff at the beginning of the project so do early testing.

How often should we test? one recommendation is you should do user testing about once a month. It's frequent enough that you learn stuff and infrequent enough that you bother doing it so that's a good rule.

And if you are going to do user testing you'll need some participants to test on. How do we select those?

I would say you can select basically anyone as long as they're not on your team because everyone on your team has already seen your application for so long that they won't be very suitable as participants. Don't focus too much on finding the perfect participant. You can do most tests with almost anyone. If your application requires some specific domain knowledge, you should certainly try to find at least some users with that knowledge but it's always good to have a mix of beginners and professionals because your application will probably be used by beginners as well and you will find out that a lot of experts doesn't actually know everything.

And it's always good to have a diverse group of participants so you get different perspectives.

Always remember to compensate your participants for their time as well because that's just the right thing to do and you won't get very many participants if you don't.

So, that leaves us on to how do we actually test stuff? And there is a million ways to do user testing. This is one way that has worked pretty well for us and it's quite fast and cheap so I like it, takes 50 minute per participant and sometime for planning and discussion.

So we usually use around 3 participants each test session. You need 2 rooms. One where you have the participant and one where you have the rest of your team that observes. The idea is we should test regularly with small groups. So in this test chamber as you call it you have the participant of course, you also have a facilitator, someone probably a people’s person that can kind of view what you're going to do and make them comfortable and make sure that they actually speak what they think. You also need device to test on, can be a lap-top, phone or whatever you are building stuff for. And a written list of tasks for them to perform.

In the observation room on the other hand, you should bring in as many from your team as you can and also other stakeholders in the project, maybe external client, maybe I don't know your boss? And you could try to lure them in with snacks. That seems to work most of the time. You'll also need a big screen to show what's going on in the testing room and probably something to write on.

So you set up a screen sharing between the device and the test chamber and big screen in the observation room. Use Skype or hangout for this works pretty well and just make sure observation room is muted so the participants can't hear them. The reason you have 2 rooms is you want a lot of people observing so you can be all on the same page of what needs to be done but you don't want 15 people behind the participant because then they will be probably as nervous as I am.

In the observation room, you instruct them to take as much notes as you want but they have to come up with a top 3 list of things to improve for each of the participants. In the test chamber you should explain that you are not testing the participant’s performance. You are actually testing your web application and also make sure that they are OK with brought to the other room and perhaps recorded for other use. You should always at all times encourage them to think allowed. And one thing you can do before you start with the actual testing is just show them your landing page and ask them what they see, ask them to describe it and the reason for this is twofold: first you get a lot of first impressions for your landing base so that's useful and also you will realise if they actually know what they are about to encounter and if they don't understand what you're trying to do you can explain what your app does before you begin testing.

So then you present one of the tasks for them and encourage them to think about again and they might get stuck. They might have questions. Try not to answer them too much. Try not to help them too much because you want to find out what happens when they aren't guided so you can say what would you do if I weren't here and you can tell them that you will answer all questions at the end of the session.

Then you repeat until you are out of tasks or out of time. : And one thing that I like to make a point of is make sure they use realistic data. I've seen a lot of bugs that have been hidden between data. It's impossible to tell if information is showing up in the right place if everything looks the same and validation logic and stuff like that is probably not very well tested if you just smash the key board.

So at the end of the test session you should ask them any questions and answer any questions they might have then unmute the observers and see if they have any questions for the participant and repeat for the next participant. So when you are done with all of them you meet up with observers, you are the facilitator here, and they try to discuss what have you found out and what should be done. Try to agree on a top list of priorities to fix. This might not just be usability issues, might be all kind of issues, but as long as you have seen it during the tests bring it up and prioritise it. You can also make a list of easy quick fixes, always good to have like for the end of the day and stuff like that. And then plan the next session sometime after you think your fix is top priorities. So maybe next month or something like that. But remember to plan it so you actually do more than one session.

And the idea here is that you should do repeated tests. Each test session you will probably find the most important biggest issues because those are usually the easiest ones to find and then you fix those and the next session you will find other stuff and eventually you will have fixed most issues, I mean this is a bit like security, you can't believe that you have fixed everything but it will probably be a lot better.

And as I said, there are several ways to do this and I am going to go through few alternatives. You can also do remote user testing. You set up a screen sharing with the participant at their workplace or home instead of having them come to you. This lowers the travel costs and you also have made a much bigger pool of participants to select from but remember the plan for some technical issues, maybe they don't have a microphone, maybe they don't know how to connect it and may be Skype just doesn't work. You can also do on sight user testing. You go to their workplace. This makes it a bit more realistic. They can actually perform the tasks at the intended environment but it also means a lot of travelling for you. . Finally you can try and moderate the remote testing which means you find a web-site where you can pay someone to test your web-site and send a video and you look at that and see what you find out may be it works for you may be it doesn't. It's quite cheap and you can try this.

If you have found this interesting I really encourage you to learn a bit more about usability and user experience and user testing because technology tends to change really fast. Try to think of a Java script library that will be relevant in 2 years, 5 years or 20 years. But people changes quite slowly. This presentation is almost entirely based on at least 15 year old material. I hope it's still relevant. If you want to read some of that material I recommend the books by Steve Krug the first one is more about usability in general and has a nice chapter on user testing, the second is a tutorial on user testing and the first one is published 15 years ago the first version. And "the sign of everyday things" by norman(?) it's a nice book about how we interact with things around us and general good introduction to usability.

So in summary testing is good for you and will help you build better things. It's not very hard, everyone can do this and I encourage you to try it and see what works for you. I think if you try it you will continue doing it because it is extremely rewarding. It is like travelling to new culture. You see stuff that you have never seen before. It's really amazing. Try it out. Thank you. {Applause}.

NEW SPEAKER: Thank you very much Ludvig. We have time for a couple of questions you know where the mic is so please line up.

NEW SPEAKER: Hi. Thank you for that talk. I am really glad you gave this talk. I had a similar topic on the open day so I'm glad to hear it on Wednesday as well. You said that it's important that the users the participants think out loud as they do it would you say this is always the case or is it sometimes better that they finish the task and do the talking after wards for example if they need to focus on the task and - lack the thinking out loud would affect the process itself?

LUDVIG WADENSTEIN: Yes is think you should use common sense here and see what works. I guess you can let them finish and then if they seem to have a problem with something you can ask them after that task what did you think here and so on but the longer you wait the more context they tend to lose so it's always good to try to get to know what they were thinking as they think it because, well, those kind of thoughts are kind of going away so quickly. But yeah you have a good point.

NEW SPEAKER: OK thank you.

NEW SPEAKER: Hi, you suggested a regiment like once a month one hour session to go through user testing. How much can you realistically achieve, how much of your app can you realistically expect to test in that timeframe? If you've got a large application would you expect to be able to see the entire application being tested once a month or do you focus on particular features of each session?

LUDVIG WADENSTEIN: I would say it's more important you test some parts and try to test different parts than to test everything at once and maybe I wasn't clear. It is one hour per participant so will probably take you 4 hours with some discussions, but I think Steve Krug wrote in his book that once a month is usually enough because you usually find more than enough to fix in one month any way. But I mean, if you feel that you didn't find enough you can just do it more often.

NEW SPEAKER: Hello. What would your approach be when you are writing an application for specific company that would a relatively small internal team using that application because after 2 or 3 cycles they would get used to the look and feel of the application so judgment would be clouded as well?

LUDVIG WADENSTEIN: I would suggest that you use that team in that case but that you also try to bring in new people. Presumably they might hire new people eventually so - but I mean, if most of your users are from a very small group it is not really a problem that they learn the application. It is almost a good thing.

NEW SPEAKER: Thank you.

NEW SPEAKER: All right, looks like that is all the questions we had. Thank you very much Ludvig, very good talk. {Applause} next up we have Emma who is also developer from Belgium and she is going to talk about project structuring and bear with us while we set it up.

A round of applause for Emma.

EMMANUELLE DELESCOLLE: Thank you so, while the computer is still booting up, I will get started and introduce myself. I am Emma, I come from Belgium, I have been Django developer since around 2010 my first job was about 10 years ago and I have to say that I have not always felt welcomed by the Python community.so a few years back somebody gave me --

So I set this up.

So I was saying, I haven't always felt welcomed by the Python community. A few years back someone told me the Python community was turning around and trying to be welcoming to people and that I should try it again. So, I took a second look and I discover it had Django community W the Django community.

NEW SPEAKER: Can't see anything.

EMMANUELLE DELESCOLLE: It is coming.

So, with the Django community I discovered some pretty amazing people who are also nice, welcoming, there we go.

Now you should see something.

Here we go. So the Django community is nice welcoming, there has been recent effort to include diversity and that is something which is pretty nice.

So the person I want to specially thanks is Naomi, she is a member of the PSF, I would like to thank the Django community and the whole team who have made Djangocon I found this year in Cardiff, really made a great job.

So, let's get started with this talk, what does Django project look like? Does this look like a Django project to you?

Raise your hand if you, if it looks like a Django project? Okay. Doesn't that bother you?

Well, personally, that bothers me. I think in most of my application, most of my project I think having a core application or base application this personally bothers me.

In Python usually only one way to do things, one way to do things, even if I didn't like it I have been doing it for a while and then I saw something, I saw someone actually, saw someone else with this problem. How many of you saw that talk?

Okay. Not a lot of you, I guess that is a good thing. This is Mark Lavin he feels the same way about it I do. So we don't see things exactly the same way, but, we have some things in common.

This looks more like I want to see when I create a Django project. I don't want to have a base or core app in my all my Django project when I have a Django project I want to see everything in the project directory. Now, I don't know if this is clear for everybody but basically, everything that you would put in your base directly is now in the directory which has the name of the project.

I also moved the urls and the settings outside of the project directory. I moved them up one level.so that is a personal choice you don't have to do it.

Moving the urls.py out of the directory, allows you to have a second in the directory, so you can use the project directory as any other application as your core application would do.

So, here is what the settings.py file looks like. Of course, you can have a complicated settings structure like ... (INAUDIBLE) in two scoops of Django, but for the basics here, in my first installed app is the project, it is the app created along with my project and it is there on the top. So that if I create any templates inside that application, it will be loaded first. Same for static files.

Now there is my templates configuration and for the ones of you who have already looked at the templates are defined by default in the settings. It is usually two loaders, this is, this is, there is the fastest time loader and the ... (NAME - INAUDIBLE) loader, by using the structure, we can forget about the first one since everything will be taken care of by the app directories folder.

So, how can we get such a project structure easily without asking the core team to change the way they do it with Django?

It is really simple actually, if you look at the docs, there is the dash, dash template option. With that you are able to specify a projects template that you want to use for your project.

You can store your templates anywhere you want, here you can find one (INAUDIBLE) on the repository, those are some examples. I would encourage you to create your own templates and use your own templates with everything you want in them. This is just an example. Feel free to download it to use it, I will treat all the links, I will tweet all the linkings from the talks later, if you don't get all the urls.

Here I am planning the repository with every project I am planning to show today. Simply, you start a project and you specify the dash, dash template and it will just create your template.

There are three templates that are available on that, on that url, there is a template for regular project, there is a template for reusable apps when you wanted to create a reusable app while it is not exactly the same structure as when you want to create a regular project, you want a lie accepts file, read me file, a set up file for publishing by.py, if this is something you do quite often you might want to have a template to handle that too.

There is a sort of the VRF app project which is the same as the project app but for Django rest framework application, so it is basically, it is different requirements file and a few different settings.

So, what does a project template look like? If you want to create our own template this is something you might want to know. You might need to know.

So basically, template a project template is just a directory structures with some files in it. This is almost what you get when you create your project when you for instance start a project. The difference is that as you can see here, you get a project name directory. This project name will be replaced by the project name you supply.

Another difference between this and what you actually get when you start your project is that inside the Python's files there are variables, the Python files are templates, the same that you use when you write your html for your application, so you can come and use your project name inside your Python file. When you create your own template you can use that project name a little bit everywhere you want you have a few variables available and I encourage you the read the documentation to know what to do.

That is quite nice. This will allow you to gain a lot of time when creating your project. But, it is not enough.

Why is it not enough? As I said all the Python files are templates but, I would like to use my project name and other variables somewhere else for example, yesterday, Luke talks about SASS I don't know how many of you use SASS or compiler, but for my SASS file I have con fig.rb file, which n which I would like to use variables too, it is not a Python file, it is not going the work.

Something else, whenever I start a new project I create a new virtual environment for it. Well, if something could do that for me it would be really nice too.

So the solution is called cookie cutter. Cookie cutter an application by audreyr and pydanny, so the main features of cookie cutters compared to the basic Django project templates are that everything is templateable, every file is templateable and jinja two instead of Django templating.

It uses project creation a little further.

So, starting a new project with cookie cutter is not harder than starting a project directly with Django, all you have to do is type in a comment with the pass to the, the path to the project template. After that, here are a few changes. Cookie cutter will ask you to fill in questions and variables, like the project name, your name, and basically everything you would like because those variables are something that you defined yourself.

So this is the output of the project created with the cookie cutter. So you see it is basically the same thing as what we saw before. You can see the difference here in the read.me file, that is my name in the read.me file.

My name is in the read.me file because this is what I entered here.

So, what does a cookie cutter look like? A cookie cutter template is like a Django template, like a Django project template it is a directory structure. One of the differences is that here instead of every project name you get cookie cutter.repo name. This is a template. So this is the same format as a template variable, so this is, this is the difference with the basic Django template is that you are going to be using variable names here too. So if I defined another variable like if I wanted to have a directory with my name, with the name, I could use cookie cutter.... I am not limited to the project name.

So, where does variable come from? You might have noticed in the previous slide, I will go back. You might have noticed that this structure is one directory deeper than the Django template. You will find a file, cookie cutter.Json, in this file this is where you will define your variables, here I have only three variables as you can see, use jinja two filters in that file. When I enter my project name it will automatically underscore it and lower case it to have default from my repo name.

So, I also mention that I would like to have a virtual environment that is built for me. This is where hooks come in. Hooks are basically transcripts you can bite Python scripts and shelf scripts, it is advisable to have Python scripts for portability., if you want your own, for windows you can also use bat files.

In my hook here, this is the shelf scribes, it will be interpreted as a script too, I am still able to use my variable I find in my Json file. Here, I can define some shell variable, environment variables with my user name, my e-mail. This particular script is an example of hook which also down loads boot straps and compiles it for you, it initially licences your virtual end and your database.

So if you want to know more about this, there is a workshop on Friday morning. Come, if you have any special requests, we will try to work on them. If you don't have any special requests we can try to work on Django and ... templates, more interested by more what Luke talked about yesterday, (INAUDIBLE).

You can also come for the sprints where they will be a sprint on Django ... another application that is developed by Levit which is my company and it is an application that allows you to easily have multiple websites running on the same code base and on the same database which is much easier since Django 1.7 and 1.8.

(APPLAUSE).

NEW SPEAKER: Thank you, we have time for a couple of questions, microphone is in the middle of the room.

FROM THE FLOOR: Just a quick note more than a question, I love using cookie cutter but something that tripped me up a few times and was I kept forgetting to put cookie cutter dot. Before the name, it is another thing that can trip you up.

EMMANUELLE DELESCOLLE: Thank you.

RUSSELL KEITH-MAGEE: Interesting that the template you suggest is similar to the old project five template which we moved away from.

The question, you had two templates one for rest framework and one more normal projects, is there a difference in the templates other than like the apps that are in settings.py or is there a structural difference between the layer?

EMMANUELLE DELESCOLLE: One structural difference is that I add an API.py file to the Django rest work one. There is a requirements.text file within the each template. So of course they are different. No basically the settings are different.

RUSSELL KEITH-MAGEE: Thanks.

EMMANUELLE DELESCOLLE: Hi thanks for the talk, I have just one question, you took settings out, urls out, why not take wsgi....

EMMANUELLE DELESCOLLE: That is a question I ask myself quite often. I guess you can take wsgi out if you want to. I left it there because it is something that I only use in production environment and I wanted it out of the way. So that is all. In.

NEW SPEAKER: Thank you very much Emma.

(APPLAUSE).

I really enjoyed the talk. This was the last session before lunch but before you go, one small announcement, the gathering for social ..., take place at the 2:00 p.m., at 1400 hours at the back of this room, enjoy your lunch.

(LUNCH)

DANIELE PROCIDA: Welcome back everyone from lunch. If you could make a round of applause to everyone out there to signal we're starting that would help {applause} an announcement. Peter Inglesby is running our job seekers clinic tomorrow where you can seek a mock interview, apply for a fake job, maybe that's not so ... appealing ... that's going to be when we're back at university between 12 and 2.

I have put on the Djangocon twitter account a link to Peter's mock job vacancies for you to have a look at so you might find that useful to have a look at if you're planning to come to the session. Also if you are really interested in helping out in the session as a potential mock recruiter speak to Peter so there is Peter as you are waving. James Pick you fur looking for Peter over there.

I've got some news about a prize raffle so we've got some nice things to give away in fact some lovely things to give away. There is a name missing. Excuse me... fantastic things. We will be passing around a box for you to put your money in you put in what you think might be a fair amount. All the proceeds from this will go to the Djangocon Committee to help the community and individuals in Wales and the UK and internationally go to events, set up meet ups and so on. We did a similar thing in France last year at Djangocon raised enough money to bring someone to this event who couldn't have come otherwise so that is where the money will go.

We have some lovely prize's we have from heroku we have this lovely skate board. We have books from Lincoln loop. We have something from plural sight 500 dollars and from Ola we have the original water colours of the Alice slides Alice watercolours she did for her talk yesterday so those are the prizes.

We'll choose them randomly from our attendee. S if your name is chosen and were asked if you put some money on the pot say no sorry go to the next person just say.

It was suggested to me by someone that we should choose the prizes by flinging the Kate board into the crowd but I thought that a bad idea on health and safety grounds. Our sponsor Tiger has come up with a much better solution. We're going to select are winners in a different way, he's participating in an archery session later today so is a really good shot so you could make yourself in do lucky winner of Djangocon prize.

Now to our next session the state of the CMS in Django with Iacopo Spaletti and Tom Dyson. Please welcome them {applause}.

So you may well know both Tom and Iacopo who have been long time members of the community and they're both leaders of 2 of the most active and vibrant CMS projects in Django. Django CMS and Wagtail.

I think the CMS is really important. It's a content management systems are not regarded as supposedly sexy topics in Django, partly because nearly everybody at some point or another uses a CMS. But by the same token because everybody does use a CMS, for many people it's their way into Django. So, one of the things I'm interested in is the way that word press has been a way into PHP for so many developers and the question that I'm interested in for Django is how might this work for us?

So we're going to have a little discussion comparison of these 2 projects under different lights and we would also like to invite you the audience, if at any time during the conversation if you want to ask something or say something please walk up to the microphone and join in the conversation. So I wonder if you could both take a couple of moments to start to describe the core ideas or philosophies at the heart of your respective projects what you have decided to include in the system or exclude from it, exclude from it, whom it is aimed at and design approaches.

IACOPO SPALLETTI: So the main philosophy of Django CMS is being just another Django application so it's something you can add to your projects and not take over your project and does not require you to write application in some specific way so it's one of the approach of Django CMS is just be another tool in your tool set to use when you double up projects and that's basically the main principle behind Django CMS. And over the time Django CMS has been developed having 4 different users in mind and the first one being the developer obviously who set up the project, who write applications, get all the pieces together and make beautiful projects running.

The second one being front end developer who write the front end and defines the graphics out of the web-site and so on.

Then there are the 2 most important users of the CMS the editors. The power enter who in the Django CMS define structure of content has power to define exactly in which content goes in which place into the - into each page. And the end users, so the low level editors so to speak that actually populate the content and everything in Django CMS has been built so everyone of these actors has its tools and his work flow to work on a Django CMS project.

TOM DYSON: I think that's really an interesting differentiation between the projects because in some ways wagtail as Iacopo just said the most important users for them are the power editors and editors. In some ways wagtail is seeking to remove control from the editors from the site administrators and we do - so I run a web agency called torchbox and wag tail is our project and we do for our sins quite a lot of work in droopel which another powerful CMS and it's one that gives item administrators quite a lot of control so you can create different views and create a structure of the site but as we find as people implementing sites often that power doesn't get used by administrators but it adds a level of friction for developers. So wagtail is focused more squarely on Django developers and that sounds obvious to say but actually you can use word press or droopel without knowing much PHP but wagtail is about removing as much friction as possible as a Django developer so using standard Django models to define pages and using migrations to make changes between pages and using standard templates and so on so that's an interesting differentiation. Another 2 core points for wagtail are the user interface so many of you will have built your own content management systems in Django and typically start with Django admin and often it stays there and some CMS's continue to use that but we were clear that's not the ideal interface, not the ideal environment for authoring content. And the last thing is performance where we've tried to make it as fast as possible both for an editor and when serving up pages front end.

DANIELE PROCIDA: And what aspects of Django have made it easiest or actually have made it hardest for you to implement these design approaches in practice, where has that fallen down for you?

IACOPO SPALLETTI: For me it is easy to say what Django made easy for us to do, so I basically Django CMS uses among the other things but the 2 main points is the REM(?) which we use sometimes in a very fancy and non-trivial way and actually I'll talk about his in Django {inaudible} because we do a lot of nice things with the REM and the admin. Django CMS is basically built - every editing feature of Django CMS is built on top of the flexibility of the admin of Django admin. You especially in the last few versions, you will see really the admin if you don't go into it, but you are going to use the admin whenever you double click on a content to edit it, you are actually using the Django admin so I guess this is what actually Django made easy for us to do.

The only pain point for us is the URL {inaudible} which we abuse a bit with our way of dynamically hooking up into the project url(?)con and in some ways it's been tricky to actually make Django pick the dynamic thing we're doing with our URLs basically. We love Django and - Django CMS, it can only be built on Django basically so - that's our point of view.

TOM DYSON: Similarly, I would like to come up with some devastating criticisms of Django but there haven't really been any and we're just trying to make the most of all the great tools that Django gives us, I think one point which we might come on to later is around deployment and I think there - it's still not as easy to set up a Django based web-site as it is to up load some PHP files and that is still a point of friction for early first time users I think.

DANIELE PROCIDA: We've had a number of talks over the last few days about front end development and how Django developers need to be able to work effectively with front end tools and frameworks and ultimately with the web visitor or the web user in mind and CMS very heavily involves web users. Does Django need to do something about the way it embraces front end technology in order to make the life of the CMS programmer easier?

TOM DYSON: Again I don't feel there is much friction here. I think this is an interesting direction for content management systems and probably Django applications in general. I imagine this split between front and back end will carry on increasing and we've got a really big user of wagtail who we're not allowed to talk about yet who are using wagtail just as an authoring tool and are not going to use it to serve out pages at all. They've got a very sophisticated segmentation thing for their campaign which is going to be primarily mobile driven and they're using it just as a tool for gathering content then distributing do wagtails API and I think this could be a trend and also there is this concept in a similar way head less content management systems where the tool is about aggregation and moderation, work flow and authoring but doesn't handle so much the distribution of content, that's taken care of by Java script front end or RMS app.

IACOPO SPALLETTI: {Inaudible} Most of the traditional views of CMS is just build a traditional web-site for web engines, search engines to index and pop up in the first place in a Google search and that's pretty much right now, as of now, as of the state of technologies, this can only be achieved by using plain HTML and the usual front end and stuff, but more and more especially by being CMSs in the Django world which means you have access do lot of applications and different environments, CMSs are a tool for the editors to create content that doesn't not necessarily goes into a traditional web-site but I think that from that point of view we are very good position as it's really easy to integrate different Django applications picking content from CMS and dispute from whatever front end framework you choose and however you want, so I think we are in a pretty good position and there are a lot of ways to make Django interact with other fancier tools. I think later - no, tomorrow, there will be a tutorial about React and so it is - I think all of us has some interaction with this front end tools and yep probably at some point think will be easier but is still - we are still a good way.

DANIELE PROCIDA: Speaking now about your project more widely is there some advice that you'd have for the other about hard learned lessons that you've learned during the past years or - and wagtail is much newer so you've had less time but perhaps you've been able to look at mistakes that have been made and avoid them.

IACOPO SPALLETTI: I think that the lesson number one that valid for any project especially open source projects is communicate correctly with the community. If you know a bit of Django CMS you know there has been some quarrels about the removal of some features from 2.4 to 3.0 et cetera and this might be a pain point for an open source project to move along and innovate without leaving users behind and leaving users in - without letting them know what's the road ahead and what's the - why certain decision has been taken.

TOM DYSON: We're a much younger project so I don't think we've got any lessons to tell Django CMS but we certainly have learned lessons for ourselves and it's mainly around how to manage open source project, we've been contributing to open source for a long time but it's the first open source project we have launched half a mile away at Django weekend last year, that was a really exciting day for us and Paul who is giving us the next talk submitted the next pool request within half an hour of the launch and I remember going home and we did a launch on hacker news and I got a beep on my phone from a github issue from a guy in Sweden and I woke in the morning and some guy in Mongolia had fixed it and I thought this is incredible, people round the world doing stuff for free for wagtail.

But that didn't you know, that was to get, I guess the first flush of enthusiasm. We lost, a few really excellent contributors in the early days because we didn't respond quickly enough and we didn't handle the pull requests in a sensitive way, I really regret that, we are trying to formalise that now and doing a better job of making sure people know about the roadmap and know and understand how the work they contribute is going to get processed.

DANIELE PROCIDA: I am also a member of the Django CMS core development team, that has been an issue for us, how to nurture and develop the community. This is something that every community has to find its own ways of doing.

I think the million dollar question, how far away are we from having a word press level smash hit success for Django in a CMS? I am sure that is what would be the tidal wave of new Django developers? What do we need to do, to knock word press off its perch? Can we do it.

IACOPO SPALLETTI: As Dan said earlier, the problem that any Django project place is the deployment, which is the main hurdle that any known Django developer will encounter when trying to working with Django.

I think that I don't know, if this is going solvable in the near future at a technical level, probably more easier to solve at service providers and that will create more and more tools, not having to deploy Django projects but working with them in a yes, in some kind of platform the service and up load the code and get it running without fiddling with a configuration of wsgi and whatever.

TOM DYSON: I agree, this is the main sticking point, although Django CMS is doing work with Aldrin, a platform, seems the only way to go.

I think another way it seem to me of accelerating a CMS, advancing themes, that is why it is successful, install it somewhere in a couple of clicks and pick your theme. Maybe the answer is having standard themes but I don't think that is particularly the direction for us, that doesn't feel where we want to get to with wagtail.

DANIELE PROCIDA: Shall we have questions?

RUSSELL KEITH-MAGEE: Extending on that point, I think I have a hypothesis of why it is successful, not that the templates are just available, but eco system it is ... it is easy to monetise the change, not only books telling you how to build but telling you how to make a business by plugging in and up loading this to this commercialised service. The Python community doesn't have that or any analogue of it at the moment. I am interested as to whether having that commercialisation angle would in any way improve your prospects or in your opinion, the prospects of Python more broadly.

TOM DYSON: That is an interesting point. I feel like there is a hesitancy in Python and Django world to adopt that type of model. That is one of the nice things about the community perhaps. But, I think you are right, it is, probably does slow down the commercial take up and to be honest we as an agency building we are not even sure what the kind of monetisation route it is ourself, we are building as a tool to help us as an agent, build it more quickly.

IACOPO SPALLETTI: I agree with you and Tom, making easier for people to monetise out of services not, based on our platform but not directly building code for our platform is a way to make easier to attract more people and there are non-developers that is way to drive the adoption to outside the, not really small circles but in any way.

DANIELE PROCIDA: We have got literally two minutes, we will finish in whatever we can in two minutes.

FROM THE FLOOR: I wanted to comment one of the differences between word press and any Django is the ease of administration, to update the installation, click a button on the page in word press, Django at the moment has nothing like that. I think that is a big barrier to why adoption among amateurs.

IACOPO SPALLETTI: I agree, I don't know if this is been taking solved at Django level, probably this might be solved at in terms of service providers.

TOM DYSON: Anyone seen that type of feature in a Django app?

FROM THE FLOOR: I don't know, okay, the thing that let's call it all the PHP application, all they have, like press a button and it updates cool on the one point but I don't really want to see it in any Django application because it involves some modifications to the code where you probably don't know what is going to happen. It might make you migrate the database, which may take a while depending on how many data you have in there. So I don't really want to see that.

DANIELE PROCIDA: I am afraid we will have to wrap it up there. I am sure Iacopo who is representing Nephila has a sponsorship stand and Tom is on the wagtail stand. They will be pleased to talk to you, thank you both very much. (APPLAUSE).

NEW SPEAKER: Hello? Yes, so our first after lunch will be Paul, a developer from London, he will describe his experiences building RESTful API's.

PAUL HALLETT: Thank you.

Screen don't fail me now!

Okay, cool. Hi everyone. So if anyone has seen any talks I have done in the past year or so, I have spoken about why you should use Python and Django to build RESTful web API services. Spent the last seven or eight months doing what I was trying to advocate, I want to be my devil's advocate, great to build it from the ground up. What is it like taking a big technical debt ridden service, adapting it to be a HPP service instead. So hopefully I will share some interesting insights with you, so you can take back or maybe you can correct me and say, you are doing it wrong and there is another way of doing it.

Let me talk about myself, I am Paul, I look like this emoji, I work for a company called Lyst I consider myself to be a bit of a fanatic when you see API services, you see the companies, API products I love building these things and love using Python and Django the do that as well. Over the last few days we have seen lots of projects for science and maths. I have been doing cool projects not that socially good. I built one for Pokémon, I did that in a weekend, it got popular, it has Pokémon data and Star Wars, anybody who wants to know what ships Luke Skywalker piloted, it is all there.

Why is it important? When I started to work at Lyst joining with the intention of helping them to take their services and adapt it to work with http. Well you know your stuff about Pokémon and Star Wars, we are a tech company in London, aggregate all the clothes on the map. Do the image mapping and data science stuff. We let you buy the clothes through the website, Amazon for clothes, they have a mobile app as well. As I said, when I joined, what they wanted me to help do was to take their existing mobile API service and adapt it to that, so that it stuck to the rules of http much better, what it did look like, keep in mind it was developed quickly, had to get things out of the door quickly.

This is using, how you would get a product for the mobile API. So pass the parameters to the url with the post hpp method. If you want to post to a user, take them, send it to url with the post method. This is remote procedure. The problem is this, it doesn't map well to http and I know it is this thing everything can do what they want with. It is like the wild west. There is no approval to get something on the web. So people adapt it to work properly.

But the problem with this, well there is a few problems, I won't tell you all of them. We are using 1url. We are registering users, getting lists, buying products through the same url, but doesn't map to the idea of http in your eyes, post method for getting products we are say, we are performing this unsafe action, we want information about the product and beyond http as well, it is hard to debug, so this is a central log, I have no idea what tissue. We know there was an error on the server.

I looked at the current API I came back and said, RPC doesn't work well for http, but it doesn't work well for this, we should not use this. What I really wanted to do was to rebuild the entire service using Django rest framework, we spoken about it a lot. For the benefit of those who are not aware, I was asked, why is it useful?

There is this popular phrase going on, Django is a batteries included web framework for building everything you need to build websites.

Well I like to use that idea when I describe the rest framework, it is a batteries included framework, it gives more than you need to do stuff with API services and Django, they go together brilliantly. Most tend to load it alongside Django, maybe it should be part of Django, so I went ahead and started to re-implement the service using the Django rest framework, I want to share some of the things I learned specifically to do with Django rest framework some of the nuances you may encounter and some of the things that were holding us back, more broadly to do with http, API services.

The most important thing to realise when you are trying to take a service and adapt it for http or anything else is that it is a really good opportunity for you to look at what you have and think how can I rebuild this? At Lyst the original idea we had five years ago is not what the service is today, along the way we have done monkey patching and gained some technical, learning how to get the structure into the nice representation on the website.

I wanted to take the approach with the rest API service, how are we going the provide -- the mobile app it wants products and register users and group the products into collections.

I spoke to the engineering team and I said, we should design this first and then code it. A lot of engineers don't like designing thing, they like just coding which is great. If you need to do things quickly.

You will probably end up with an external API that is just as bad as the internal services that you have. If you have internal services that are great, ignore this advice, go ahead and just do a direct map. But for us, I realised we should design this first, we should definitely think about how it is used and obviously REST as this idea of resources these are the thing that is owe need to focus on, what is the resource for us and we want to share. Often with Django, I want to share the model I have. But really what you should be doing, your models don't have to be what your resources are. They can be less or more. I want to share with you an example of where this happened with us. We have this internal model called cart. This was built, we have a load of data on, the shipping and billing address, as well as the information about the cart. There is a separation of concerns here about these various different resources, cart information, there is shipping information and billing information. So what I wanted to do was to logically separate this out into separate resources so we could address those various parts when we needed them and not get back the whole lot. We can do this using serialisers, this is probably the most powerful tool. Some people just install it for the serialisers, it works with the forms, except with the html form component. You can serialise it and have an external version of it and do it in reverse and validate on the way back in.

This is what it looks like, can everyone see this okay, it is quite light in this room and it is dark slides.

We have this model and the fills we wanted to expose., the extra attributes the number of items in this cart, serialise the method fields, wouldn't recommend doing anything heavy or complex.. This allows us to take that internal model that we have and serialise it in three different ways the exact same model but serialising them differently, to make the models, you can go a step further and change the attribute names and map it through the serialiser.

Serialisers are a good tool, there is lot more than taking the model and validate it. You can do more with it.

Router, a component in the REST framework, which some people find useful for ... they regenerate your url configuration for you. I thought, we may as well do it ourselves and have better control. For a lot of the nuances that I will speak about in a minute, routers just weren't applicable so we just threw them out of the window. Heard a lot of people say this, okay for a crud based service but for other http services you may not want to use it.

Permissions and authentication, Django rest framework ... we switched something that was ... allow any, anybody can access the end point, somehow they managed to find it. Set your default permission class to admin only. You ... if someone does something, you probably know who they are because they have admin rights.

This allows you to have explicit control over your views for your REST API and you can say which views you want to have. I won't go into much into permissions you can have expressive ones, I want to allow users from this country, looking for these type of products, access to the API. You can do this window accessing the framework views.

The other part of the Django rest framework, this is permissions, this is proving where you are, whereas permissions is saying whether you allow access to this thing.

People is v said, they have an internal consumer and internal server on the same network, even if you have one consumer of your service, you should have authentication, even if it is token or a string you have written yourself, Django rest framework comes with a bunch of different authentications you can use, I recommend token off, which is a random string on your server, give it to your client, they have access because it matches.

If you want more control over per user and per application, you should use or (NAME - INAUDIBLE) the. It is a head ache to set off. The trade-off is much better, to be able to make tokens expire after 30 minutes means you have greater security.

Django rest framework recommends you pip install this tool kit, migrations, set your default authentication class to that thing there.

I think this is because it is the same component in itself and if it was part of Django rest framework, then it would come this plus the tool kit thing.

Anyway those are some things I spoke about related to specifically Django rest framework and Django when you are building API frameworks that you might not realise or that is useful.

I want to talk about http in general and things you might encounter, we have covered this a lot already it is great because I want to fit into the gaps that other people spoke about, give my opinion. The thing that everybody needs to be aware of http is not being used today for what it was intended. Even when http1.1 was standardised it was used for things it wasn't intended. We have ducked taped it to get it to work the way we want it to. We have to be aware of that, when we adapt our services in software but don't work in http I think you need to take a grain of salt. You have to bend a rules a bit. I have a really good example actually, so we had this one, we wanted to be able to provide our mobile app with the ability to take a cart and check it out. So to process it and purchase the items.

Initially, I was like hey you know what, I will make a new http method it is a string, but servers don't support it and clients might not support. That is a bad idea. The way, the solution I came up with was this. I have had some people give me feedback on there I would love to hear what you think of this after.

So the solution I came up with was to use a method that I knew matched what was happening on the server, post means something unsafe is going the happen, something is going the change, we are checking out and checking out some items, we shall making a specific url not a resource k but a sub resource in a way, I am giving the cart to the check out. Worked brilliantly in my head. This is how we are currently doing the checkout method. We passed the idea to the cart and it handles all the rest.

So the last thing I want to talk about is asynchronocity.

Http is inherently stayless, each request response shouldn't care about other requests and responses they time out after usually like 10, 60 seconds.

I want to demonstrate to you one of the ways we do it. David talked about having a poling on the client side. Another way of doing this is to have a web poke, I avoid to use the word ambiguous, it is hard to demonstrate what it means.

The method I had, the url that checks out my cart, we check back this, or for a mobile application, we pass the push back attribute, these are optional, don't always have to be passed together, do both or one of them. We receive these and store these and make sure that we keep these related to that process in celery or whenever it is happenings. What we do immediately is we send back a 202 accepted response. This closes the connection which is nice and done that part of the connection is done and http 202 accepted is how http says we've accepted what you've asked us to do we're trying to do that right now and when that process is finished whether or not it's failed or succeeded we have our service send you our request to that URL like I had previously or drop it into our push notification cue for apple and android devices so this is reverse to the idea of client polling constantly, it means your client doesn't have to keep asking is this finished yet, server sends the request saying that things you want us to do we've done it now. And we also check to make sure we get back the response if we don't get back a response say if you are in a tunnel or server is off-line we'll do that request again in 5 or 10 minutes.

The last thing is documentation I regardless of how good your API is how tested your service is how great and rested and monitored it is if you have bad documentation you have bad APH sorry if you don't have documentation you have a bad API! There are good restful tools to develop documentation automatically one called swagger which will build it for you but sometimes you are bending rules of http and rest to adapt your services often we found we may as well write it ourselves, figure out how we want the service to work and be explicit. If you're not a fan of documentation at the bare minimum provide users with a way to get their authentication tokens it's the biggest pain point for API services having worked previously for an API company the biggest drop off was people finding out how to get tokens, to make sure that process is nice and easy is useful. And if you have time do card examples. I can usually turnout how to turn curl into a python request thing.

To wrap things up there are a lot of cool things you can use in Django rest framework and be aware you can use serialisers because I really like serialisers they can replace forms if you don't like forms. If your building a service that already exists try and figure out how to adapt it didn't just dive in because it's probably going to be bad. Use this opportunity to think about how the service works internally and how you non-expose it and where possible change the rules of http just a little bit, stick near the beer path to get it to work and if you are using RPC for http please don't it makes me upset and it doesn't work well. That's all I have for you if you have questions I can take one now but I'll be around the rest of the day thank you very much {applause}.

NEW SPEAKER: Hi, just one quick question, I noticed on one of your slides you had G zip compression and I'm wondering if you are still using it for your API.

PAUL HALLETT: I had no idea I had that - I literally just stuck it in.

NEW SPEAKER: It's one that said this is what debugging responses looks like and I thought oh I wonder if he's doing the breach thing.

PAUL HALLETT: Oh that's the old one - the developer who currently works on that doesn't work there any more so I haven't been able to ring them and wring his neck and say what have you done?!

NEW SPEAKER: So the new one doesn't work?

PAUL HALLETT: Not at all. Anyone else? Cool I'll be around rest of the day if you want to grab me. Thank you. {Applause}.

NEW SPEAKER: So before our next speaker a small announcement. Because of Moses couldn't attend the conference we'll shuffle the talks a bit basically Adrienne's talk is going to be before break not after, so that's it, and our next speaker is Ana who love open source and soon to be potato member in London and is going to be talking about mixins {applause}.

ANA BALICA: Hello.

NEW SPEAKER: Hi!

ANA BALICA: So this talk is about understanding how mixins work and how do you create your own mixins using Django framework will in the context of Django framework? I hope I don't bore you to death because I don't have any kitty or gifts up my sleeve. I am Ana and you can find me on twitter to ask a question or simply say hi again!

So, mixins are a controlled way of adding functionality to your classes. But, mixins are not special language constructs. This means there is no key word in Python that will point to some construct and tell you this is a mixin. In fact mixins are ordinary Python classes. Take a look at the short example. It's a class called some mixin and it has one method which does nothing. Well, this is a useless mixin obviously because it doesn't add any value. But the important part the thing to pay attention to is the fact the mixin adheres from object if you are using new style Python classes. Basically what I want to say by this is it's preferable that mixins are not derived classes.

OK. Mixins offer modularity and that's the main reason we'd like to use them. When to use them? Well, when we want to re-use a particular feature in a lot of different classes especially when that feature doesn't belong in any of the existing classes. That's why mixins should be very narrow in scope, should have one single responsibility, they're not meant to be extended and they're not to be substantiated.

I mention mixins are good in making modular designs so let's take a look at the decoupling concept. Imagine you have an encapsulator that has 4 colour-coded features orange, blue, green, yellow, so far so good. Over time the project grows bigger and we create new classes that encapsulate their own features. And at this point, you might see that his code smells. Well, there is a problem with these 4 classes and the problem is the yellow feature. I bet someone copied and pasted it.

What do we do in this case when you extract the yellow feature and put it in a separate encapsulator so the current and future classes can benefit from it. And here comes the question. How do you actually use mixins?

In Python and not only the concept of mixins is implemented using multiple inheritance and when we talk about multiple inheritance in Python the order of inheritance matters. It's also called method resolution order but most of the time you don't need to care about it if you are not dealing with multiple inheritance.

So here is how we use a mixin say we have a foo class and it inherits from some base class and some mixin. You may think according to naming conventions base foo acts as base class and some mixin is indeed a mixin. Well, might be might not be not. Nevertheless the way we read the following code is like this. Some mixin is a base class extended by base foo extended by foo. This might not be a problem if some mixin and base foo are not sharing the extending or overriding any of the methods. But if you are conscious that some mixin will be extending a method of base foo you should reverse the order so right now it looks much better.

An easy way to remember the Python inheritance order is with one single arrow that goes from right to left. And if you start chaining makes sense. Be especially careful about the order.

OK. Who is using or has ever used Django class base views? Ooh a lot of people. Great. So this is a snippet from the official Django documentation, it is a class based view about inherit, subclasses template view and template view helps us render a template, that's all it does and the minimum configuration we need to do is set up template name. Perfect.

Well, class based views in Django is this one typical use case where mixins feel very natural so we're interested in plugging in over there a mixin.

So at this point we know 2 things: we know what are mixins and we know how to use them. History moment drum roll please ... thank you! Let's write the first mixin.

Common use case in web development is super text some of your pages so we want in force our end-users to authenticate themselves before accessing a page. Let's create a mixin that will check if our users are authenticated.

So what we do is the following. We choose a good class name, log in required mixin. It inherits from object and is placed inside some app user profile or it might be placed almost anywhere depending on common sense and conventions in your project. So we'll like to check if user is authenticated before we start processing the request. You might know or might not know but I'm here to tell you that one of the first methods to be called when processing a request is dispatch method. So what we do we just use the definition of the dispatch method from the base use. Now to do the proper check. Look at lines 8 and 9. I hope the code doesn't look too scary to you. We say the following. If the user is not authenticated please raise a permission denied exception and since we don't right rat any Java code and report back packages let's go on-line and specify number 2 important statement for permission denied from Django card exceptions. And the last thing to do is handle the use case where users are actually authenticated and we want to allow them to access the page therefore, let's call the super method and resume the work. I'll leave it up to you to provide the dogs and testing.

Let's go back to our about view. We plug in the log in required mixin and tah-dah, it works. There is nothing else we need to do. Now, all the anonymous users are not allowed to trespass to the land of the super-secret about view info. It is a trivial example but it demonstrates a mixin in action. Here is how we can describe it graphically.

Once again - the about view inherits from template view and also from log in required mixin. What if at some point we decide that the about view is not a template view but may be it is a detailed view? Well, no problem. And if you didn't see what I change on my slide I basically swapped the template view with the detailed view.

That's a good thing right? Minimum effort is required to use a different view with no additional cost. This looks simple to me. I hope it does to you as well.

So I'm preaching here about how cool are those mixins but what if there is another way? There are tons of different ways but it doesn't necessarily mean that they're the best. This made me think how would I do it another way? So I might check if the user is authenticated writing the view but that's obviously a stupid idea for obvious reasons so I came up with this and is obviously a stupid idea. What I do I subclass the template view and create log in required template view then I inherit then the about view inherits from the log in required template view. Doesn't look that bad at the moment but it actually does because every single other view, the number of which I can extrapolate to infinity, that you want to have an oaf(?) check can be extended the same way. That's exactly the same problem we discuss at the beginning. We didn't solve anything. We just copy pasted it. Then take a look again at the mixins diagram and it's elegant and simple so that's why mixins are better.

OK since we all agree mixins are fantastic, how about looking a bit more inside class based views and some common methods of the class based views?

If we want to add a feature to class based views using a mixin we need to know some of the internals. Once a request hits a template view one of the first methods to be called is dispatch we've already seen that right, it figures out which method should be called whether based on the fact whether the request is a get or a post or something else. It's useful to override to the extent dispatch in order to check if the user is logged in or user has permission to do something. Another useful method is get context data. Sometimes it might be useful to extend it if you want to add new context to the data. Basically what it does, it takes your key word arguments and sends them to the template context.

Get template names is called by another method rendered response and basically what it does is listing all the template names. It can be extended to add more flexibility to your template names. If you want to have a different mechanism of identifying them.

But you might be thinking, well, that's kind of a lot because what I show you at first was the simple example, the template view which had one single attribute and then this happened. So obviously I need to tell you where do you find all this info? My personal approach is just go ahead and read the source code. It is quite easy to get your head around Django source code, thank you core developers and all the contributors, it has doc strings and the code is readable, but there is also a different approach, you can go, you can start with the Django documentation and read a little bit more about class based views. Also not by accident Django users mixins to build the class based views and that's why I felt I said that it feels so natural to use mixins when you're working with class based views in Django and you can read about them at this URL. I highly recommend as well to look at the source code on class based views and mixins as well because it's a good example if you want to get your head around mixins and how to start implementing them.

And the most awesome source. Andrew shod me yesterday so you are lucky it's this little web-site - CCBV.co.uk and aggregates in one place all the class based views their methods, where they come from, mixins, heroku source code, so good it will probably breach my idea of hunting through source code of Django and just go straight to this web-site.

I made you go through all of this log in required mixin code but in real life you probably won't bother implementing your own log in required mixin unless you really want to. Because there is Django braces.

It's a package that aggregates common mixins that you will need in your project, structured in 3 particulars. Access, form and other. So, just to let you know there is something already done for you, so use it.

Be careful when you decide to write and use mixins because with great power comes great responsibility.

Be aware of incompatible mixins of chaining too many mixins and over doing them because it will become a nightmare to figure out the execution flow.

So to recap what we have learned so far. Mixins in Python are classes and they have one single responsibility. Semantically we think about mixins as some plug in reusable functionality. Although mixins are mixed in using inheritance they're not creating a sub type relation{inaudible} this means they don't link to the Lyst principle of solid, the principle says the following. Instances of base class can be replaced by instances of derived classes and it will make perfect sense. This statement doesn't hold for mixins, we don't do that with them; that's why they don't create sub typing relational(?) -

They act as plug in functionality. I hope I convinced you that mixins are easier to create and are supercool to use, I strongly encourage you to go back to your views, take a look, with a pragmatic eye and maybe start writing mixins to clean up your code. Thank you.

(APPLAUSE).

NEW SPEAKER: Thank you very much Ana, a great job. We have time for a couple of questions, the mic is in the same place as in the morning so you know where to line up.

RUSSELL KEITH-MAGEE: You suggested looking at source code best solution which one is probably true not ... (INAUDIBLE) documentation is good. Is there something we can do to actually improve that or is there a way we can document mixin that is you will be able to understand them. We have had a couple of swings at it but still not got there.

ANA BALICA: The sprints are coming! so I will guess you we will try to get our head around, get the documentation easier to understand, at this point. It is more like use cases and more like specific use cases, so you propose a solution, then you say there is a better solution, then there is even a better one, so this is slightly confusing for some people, maybe we should start with something simple and then grow into user cases.

FROM THE FLOOR: Thank you very much for your talk, awesome mixin and extending the get contest data method. You apply that to many, but let's say two views, you will like to pass more extra data with this method whilst the right solution for that? Thanks.

ANA BALICA: One dry solution is to use the maybe the template stags if you are not relying if your context doesn't rely on some of the specific grounds so if you strike that to a tag, that might be a very dry solution, but talking specifically F you rely on your url and other specific params, think for the future, you probably know what the project is structured you know what will happen in the future, if it makes sense, at this point you will get the mixins in two views, you suspect at some point the project will grow, allow it to grow naturally, by actually dry like you have already repeated yourself twice, so make sense.

FROM THE FLOOR: Oh Hi, thanks for a great talk, I want to go back to this based foo and, you said in here like, yes that was great that you put the order in the right direction. But, don't you think it is better if some mixin is already composing or using some data or changing some state in base foo then it would be better to compose some mixin to base foo and foo class can inherit from base foo without the mixin. When you use the nixen class configuration of foo ... you wouldn't expect to change or access anything of base foo in that sense?

ANA BALICA: But that actually really, like do you want to do this? But in a more like better way? Do you imagine this to be like the template view to be the foo?

FROM THE FLOOR: The base foo would use the mixin to conform, then ...

ANA BALICA: Depends on your specific context but, try to think, try to keep things decoupled. Maybe this mixin, maybe the functionality that exists in your mixin might live in the base foo but maybe it might not. If it is possible, sometimes it is hard to decouple things, sometimes we put stuff like all in one class, keep them small.

FROM THE FLOOR: Thank you.

NEW SPEAKER: Right thank you very much Ana again (APPLAUSE).

We are setting up for our next talk, so please bear with us for a few seconds.

ADRIENNE LOWE: Okay ... you can hear me? Hi friends!

How do we plug it in? You can shut it off for a moment if you would like.

Okay we have found a phone, so check your pockets if you have lost a phone., check at the registration desk.

ADRIENNE LOWE: Okay?

NEW SPEAKER: When we asked the next speaker how she wanted to be introduced, she said in Welsh. So!

[welsh]

Adrienne talking of coding with knives.

[rough translation]

ADRIENNE LOWE: Hello friends, can you hear me okay? Everybody can hear me? Way at the back, Hi, I am Adrienne, thank you for the marvellous introduction, today I want to share some stories from my journey learning to code and teaching others to cook and discuss how lessons learned in one subject can help in the other. I gave a talk on open day that was aimed at anyone just getting started with Python and Django, where I shared three lessons I hoped would be helpful to the folks new in the audience, today I would like to turn my attention from those who totally new to the subject instead address those in the audience with a little more experience, since many of you may be somewhat removed from the experience of first learning to code, I will share some stories that will perhaps remind you of what it was like to feel that way and to give you a sense of the challenges and opportunities confronting the new coder.

My hope is that in sharing these stories you will hear something of your own experience and will be inspired to offer your help and expertise in ways that feel authentic and meaningful to you.

But first I want to offer my sincerest thanks to Djangocon Europe and the software foundation for making it possible for me to be here by providing a grant. I am proud to be part of a community that values diverse perspectives with such thoughtfulness, intentionality and accommodation. Thank you so much for making the resources available for me to be here. (APPLAUSE).

I am visiting Cardiff for the first time from Atlanta Georgia, they say that nothing rivals southern hospitality, since being here, I have been overwhelmed with the kindness that has been shown me.

Back at Atlanta, I organise, this is our mission. On open day we had some fun, I read this in my southern accent, I am sure you don't want to hear my southern accent? You do? You see you want to hear what I sounded like when I was knee high to a grasshopper? All right if you insist.

"our mission to s to promote, educate and advance a diverse Python community, throughout reach, education, conferences events and social gatherings!

(APPLAUSE).

So that is really what I sounded like when I was little and would run through the yard with an armful of cats. Back home we meet regularly for the things you would expect, later this summer doing Django girls for the first time. I am a regular attender of our Python meet up group, I gave a talk last month and also travelled to our regional conference to speak in February point I give talks about the experience of the new coder because I still consider myself quite new. I started my first Python tutorial last winter and only got know Django earlier this year, one place I am not so new in the kitchen. I chose an old photograph. This one is over 10 years old, issued jacket, because I have been doing this for a little bit.

I grew up on a farm in the country, in northwest Georgia, we had a big garden, this is the garden that my parents planted together and this is my old house. Started cooking for myself at a young age, went to college for history and philosophy, my first jobs in kitchen, double full time, 88 hours a week, that is a lot of hours a week. I cooked for a 100 community meal and wrote a thesis.

Finishing graduate school, worked as a chef serving throughout the community, pop up dinners, this was an example. This was a dinner I did for a non-profit and offering private and group cooking classes.

I came to Python through my long time use and passion for open-source software and I just love the branding for this year's free software, I thought it was cute.

Because I want to help women like myself from nontechnical backgrounds get started on their journey with programming. But first I was a cook. I started a Blog, coating with knives, to marry my passion for culinary arts with my interest in learning to code.

I share cooking lessons and recipes and alongside updates of where I am on my journey with Python.

Since folks I have been asking, I thought it would be useful to go over the three things I talked about in my open day talk.

So I share three lessons from my drive from writing I did with coding with knives. I suggested that new coders source their materials from a wide variety of perspectives and teaching styles, not take it as a reflection of their own skill organiser’s self-worth if one particular resource doesn't work out for them. I say this because when I started to learn Python, I did myself an unintentional but huge disservice by only using one tutorial. When that tutorial wasn't working for me, I thought I was the problem, I perceived the author as the expert, I figured I must be wrong, felt discouraged.

I doubted my ability and I gave up. But, when a cook doesn't find a particular cook book useful, he or she will grab another one, it is not a big deal, so likewise I advise new coders to give themselves the same flexibility, if something isn't working, you can find something that will. There are too many great resources out there. Just as we had the one teacher in school, inspired and motivated us more than others, you will find a lesson that will help you get to where you need it to be. It might be this one?

I really like this one. Second I emphasise the importance of asking for help. I didn't want to seem vulnerability or pathetic for asking for help. I didn't want it to look like I didn't know what I was doing when invited to speak at conferences. I didn't know what I was doing and I really needed help. One of the reasons learners don't ask for help, they don't want to feel they are burdening or wasting their time by asking so called basic questions they feel they should already know the answer to.

I think that in asking these questions and getting them answers, mutual learning occurs that is valuable. The learner gets with what they need to some extent but the person doing the answering gets to flex and improve their teaching skills. Answering helps me as someone who teaches cooking classes, revisit and rethink my own processes and positions and I think in so doing I become a more effective educator and teacher.

Third and finally I showed how having an intentional goal purpose organisers a project you would like to build can help keep you going in the inevitable dark times, just as there are many opportunities to tear off one's apron and storm out of the dangerous professional kitchen, loved it!

New coders can get discouraged really easily. Start to wonder if there is any point in trying, I know I was there, from talking to you this week, many of you were there as well.

Since the times of discouragement and frustration is inevitable, have a goal, like maybe winning hell's kitchen! Or making a cool app right?

Hopefully these three brief examples give you a sense of some of the things that come up for new coders and perhaps remind you of your experience as a beginner.

So open day as I said was meant really to inspire new folks to get started with Django and Python without hesitation, we did Django girls, but the goal to inspire you who have more experience to help the new programmers.

I thought this one was sweet. It is a ... having tea. Since I been at Djangocon I felt an amazing sense of welcome, encouragement, the tea has been really good, from all of you. There is a real sense we are all part of a shared creative endeavour, that we are sharing and learning together and likewise in the cooking world we have a really lively exchange where people are constantly posting their cooking adventures on Blog, asking for feedback, asking to adapt recipes, I am a member of several recipe groups, some people post, I want to make X but don't want too much Y in it, where Y is typically a major constituent ingredient, something that has been always included. Like these, excuse my French, macaroons is how we say it who don't know how to pronounce it right.

Somebody will post and this was one of the hugest discoveries I would say in the culinary world in America. We figured out to make French macaroons without eggs, if you are not a cook, I should point out chicken eggs are a major and up until recently, considered an irreplaceable ingredient in these, these have no eggs at all which remarkable how they figured it out. I will, figured out, I will tell you if you come up to me later. It will blow your mind, so revolutionary, I didn't think it was possible. I didn't believe it for weeks, talk to me later.

So when someone posts a question like this, tonnes of comments will pour in from cooks saying, try this or Paddy try that, this one time, I used this one thing that would work, I used it this one particular way and I know as someone who has been in this community that that is really an ethos of our community as well. We are guided and I feel since I have been here, a spirit of collegiately, of sharing our knowledge but for many just getting into programming, it really doesn't feel so immediately collegiate or like we are all in this together. I am using sad [dog], because they so often learn to go with the coding alone, afraid of asking for help and afraid of being less intelligent, lacking the confidence to join us in collegiate spaces like this one.

They don't come to these conferences, they shy away from the local meet ups, in doing so they miss the chance to see how well we support one another, with our teaching, welcoming sprints, I will contribute to open-source for the first time this week, that something I always wanted to do (APPLAUSE).

So, this is why it is so important to invite them into our spaces to provide grants so that they can attend like the one I received to help them become speakers in short to make thoughtful efforts to get their involvement.

..To help them become speakers, in short to make thoughtful efforts to get their involvement.

Now there are many concrete suggestions, these are a few that have been explored by other speakers this week. I think we all who heard Jamie absolutely loved his open day talk on making spaces like this one more inclusive and of course Lucie's wonderful talk about reaching out Django in the social sciences where she pro-actively asked for our help, she said we need your help. But the suggestion that I want to make today in closing has more to do with attitude and posture.

I believe that one of the most significant ways we can help new coders is by bringing our full authentic selves to this enterprise. We must reject their natural tendency to lyonise us as heroes and instead help them see us as people who learned at practised a useful skill.

Perhaps in seeing us as the humans that we are, they will feel invited to bring who they really are to this table with all of their self-insecure in security hope and questions. I experienced lots at Djangocon this week which is why it's critical we get them to these conferences and I just wanted to share a few of the ways I experienced this power of authenticity.

When Russell gave a lightning tall on his experience of a major depressive episode and when he invited everyone listening to check in with themselves and seek help immediately if they even thought they needed it -

When I met the incredibly brilliant but also completely humble cofounder of py ladies Catherine and I just said in a reflective way thank you so much for all you have done and she said don't thank me, we're all in this together.

Or Craig admitted to me and John at lunch in Monday he was interested in giving a lightning talk but was nervous and unsure about it and after talking it through with us he went and signed up, he gave a great talk yesterday.

Or when Corrie shared an open day talk about his reaction to a report of a major security flaw and how he might handle it differently next time and he loaned me the slide. Or when {inaudible} explained web accessibility is not an option and he finished his talk with this, and neither are we.

And when anyone got up here and stood where I'm standing and said you know what I'm super nervous but I'm doing this any way.

Our community is of late getting a lot more comfortable with the practice of owning our own in security and doing in in a very public way and as surprising as it may sound, because it is scary to do so, this is probably one of the best things we can do for new coders and it constitutes my final piece of advice today.

We really need to be aware of the fact that new programmers find it all too easy to lionise well-known speaker’s core contributors and so on and place themselves in contrast at the very bottom or sink into the background like the cubs here.

But even when you have been around for a while it's really easy to have the same mind-set so this end I want to share a nice story from our dinner at the museum on Monday. That's us. I was sitting around with a group of incredibly bright Django noughts getting to hear they stories for the first time, some identify themselves as new to this work, relatively new, others had a few years of experience and all told me stories of really interesting projects they were working on and the more I learned about each of them the more I wanted to know and I said, why don't you guys give lightning talks? I always met with universal resistance - oh no, no wouldn't dream of it and there were a number of reasons right, the size of the audience is a common one. It's really scary to get in front of a big crowd and I just showed you all of my slides I'm terrified! It's hard to be up here. But the one that struck me and was shared by all of them was the idea that this conference is full of luminaries of people at the very top of their game. My dinner companions shared a common fear that if they were to get on stage in front of the people they respect the very most they might say something silly or uninteresting or lose their esteem. And yet these are all truly bright people. They're already accomplished in their work. They're already respected by their peers and yet ... Jacob Caplan Moss commonly regarded as one of the founders of Django gave a keynote at Python you might have seen where he called himself a mediocre programmer. In his talk he challenged the talent that he believes keep people out of TEC and arrests the creativity and ambition already in it. He explained how we have lionized individuals in our community that we've created an architect of what a programmer should like if not at least look like they've accomplished and was inspired by shadow canes work specifically her work at tenex engineer which dissects some of the same myths and I see the same thing in cooking as I teach others to cook I really fight the influence of the rise of the celebrity chef with his perfect kitchen and 30 second recipes which for the new cook never do. People who are perfectly capable making their own delicious home cooked meals I feel get very discouraged by all of this. They wonder if folks like Jamie Oliver, Gordon Ramsey, Nigella Lawson have some innate skill that they simply lack. So when teaching, like cooking, - when teaching I emphasise like cooking like coding is a skill that takes practice it is something you can learn to do Jacob Caplan Moss calls it programming an exercise collection of acquired tools or skills. I can't remember how he put it but I put tools.

So just as you are not born with the ability to cook you are not born with the ability to code. So back to our dinner party to wrap up. I decided to do a little thought experiment. I guess it's the philosophy major coming out in me so I asked my dinner companions to imagine a single person currently at this conference possibly sitting in this room who they fear peaking in front of more than anyone and when I asked why that person, the answers were relatively similar, so and so gave such an outstanding talk, they've done so much for the community, they're a core contributor do project they care about and so on.

And then I asked them to imagine and really get it into their head, to imagine themselves going up to those developers and saying, hi, I'm so and so, I really could give a very interesting lightning talk but I'm terrified to do so because I don't want to lose your esteem if I say something less than marvellous.

What do you think here she would say, I ask?

I think we've all been part of this community long enough and felt its support clear enough that the answer is very plain to see. They would say, get up there and give that lightning talk.

The point of this little exercise is to get some more practice sharing how we feel especially self-doubt in productive way and bringing our authentic selves to even a dinner town conversation and to publicly undermine that little voice that says you can't or you shouldn't or you aren't good enough or aren't enough. The fact is you are all of these things.

And if this is what you want new coders to feel about themselves, if this is the expectation that you are going to set, this is what we must model for them.

I have seen a lot of powerful authenticity already this week and as someone new to this work it has meant the world to me. I encourage you to continue sharing who you are with new programmers trusting it will inspire them more than you know. Thanks {applause}.

NEW SPEAKER: Thank you very much. Incredible talk. We have time for one or two questions; I see Russ is rearing to go.

ADRIENNE LOWE: I kind of halfway took off my mic so I'm sorry.

RUSSELL KEITH-MAGEE: Thanks so much I can't agree more with everything you said there I put my pants on one leg at a time, what everyone doesn't know how much time I spend putting both feet in the same leg. Yeah absolutely I'm not fantastic, I'm just one guy and I'm remarkably persistent if I'm leading anything it's because I'm persistent not because I'm excellent.

But to that end how do we - have you got any suggestions about how to communicate that message? Is it just a million small things or is there anything we can systematically do to make sure we don't end up making icons of people at the head of the community when frankly I don't want to be I'm just -

ADRIENNE LOWE: Can you still hear me? I'm just going to hold it thank you so much for that remarkable question and I do think in many ways it is kind of a collection of a lot of different things you do. But, I do agree with those who say that providing financial assistance to new coders to attend conferences is really helpful. A lot of you have been to many, many conferences before over many years but this is the first time I've been to one of this scale and I can tell you as a new person I feel so incredibly welcomed and so incredibly inspired. I'm about to spend the next 2 days doing something that I've wanted to do literally for years which is contribute to open source and I wish there were maybe 50 or 100 other like me so often the financial burden is what keeps them from being here so that is a very systematic thing that could be helpful. And otherwise it's just the kindness, it's the stories that I lifted up, the 5 examples from the way people really brought authenticity to the stage including you. So thanks.

RUSSELL KEITH-MAGEE: Thank you very much.

NEW SPEAKER: Hi there. Thanks very much for your talk. It was really good. And my thoughts before I gave the lightning talk was what if I'm advertising some coding examples of something which is wrong or bad, not that I'll be mocked but that I'll sort of teach someone something negative or bad that is sort of a bad practice and I suppose sort of getting the opinion of maybe not everyone here who knows better than me maybe most of them, it's a really good way of opening up lines of communication to how to do things better perhaps because I have seen 2 or 3 people come to me and say oh how do you do this, we're doing it this may and it was a fantastic way of opening up those lines of communication.

ADRIENNE LOWE: Yeah definitely, thank you so much for giving us a bit more insight into why you hesitated that's really helpful.

NEW SPEAKER: Thank you.

ADRIENNE LOWE: All right thanks so much everybody. {Applause}.

DANIELE PROCIDA: Very briefly before we stop for coffee, Vicki and Eleanor, do you want to come and stand at the front please? This is Vicki and Eleanor from the university's counselling service. Vicki helped us set up the arrangement so that we'd have these sessions so thank you to them and their colleagues for doing this. {Applause} and we are hugely grateful to the counselling service and to the counsellors for providing this.

They will be there tomorrow, both for the counselling sessions and both to talk to people about perhaps developing some kind of application which is what we are quite good at actually that might help them. So can you just raise a hand if that's something you are interested in doing tomorrow perhaps working with them? OK so perhaps you would talk to them in the break now and work out a time and suitable place. I'll leave you in tomics same hands and perhaps he'll guide that. Come into the room by 20 past ready for the next one to start at 25 past so we can catch up time we've got a lot to get through still.

(BREAK)

DANIELE PROCIDA: Can we do that thing where we start clapping to get everyone in please?

If you could sit down so we can get started. No not you personally Florion.

So. May I have your attention please?

This is the raffle box for the prizes that we have here. We have got a long board, we have got the original watercolours from Ola's slides that are down here if you can't see them. We have got some books and a subscription to plural side’s services worth 500-dollars, this box is going to make its way around the room. If you would like to be entered into the raffle, put some money in it. When we draw the raffle, it will be drawn from the list of attendees in the conference, we will ask you if you put any money in the box, if you say no we will go to the next one. Don't be caught out in that position.

All the proceeds will go to supporting the Django community and its members in Wales and internationally. So just let this pass around the room, the idea the put money in, you can take money out of the Django community in other ways if you need to.

NEW SPEAKER: While is box is travelling around the room, let us welcome Gregg, Gregg is a software developer from Dublin and he is going to tell us something about conformity. Please give a big round of applause to Greg.

(APPLAUSE).

GREG CHAPPLE: Cool good afternoon everyone. Yes. So, as I said, I am Greg from Dublin, today I am going the be talking about style conventions. I have titled conformity and you, to sound overdramatic and hopefully generate some interest, let's face it is not the most exciting topic. Briefly about me, from Dublin, that is how you find me on the internet if you feel so inclined

I guess the main kind of things I work on outside of my job is, kind of 50, 50 between Django related and rust., I started with a text editor in iota, you can find that out on my page.

At the moment, I am working for a company based in Dublin, we propylon we build legislative and legal compliance, with most of them written on top of Django, we are hiring software developers if you are interested in working with Django and a lot of open-source projects check out the website and you know, feel free to reach out to me here after if you want to find out more as well.

Before I get going, I would like to say thank you to the conference organisers and the volunteers. The whole process from submitting the proposal all the way to stepping up on the stage, as a first time speaker it is encouraging to come across such an accommodating and welcoming attitude and it is encouraging to see I am not the only one to notice that. Just to say thanks before we get going.

So yes. Style guides and code style and all the tools and rules and stuff that go with it is not exactly a very exciting topic you might think. Like you probably never hear people ever say any of these things when they get up in the morning, "can't wait to fix some PEP eight violations -- it is not something that people get excited about. We want to work on the real programming things and use real tools! whatever that means.

So, as everyday as programmers we have choices. So we have when we start a project we have a choice on what language to use, what framework to use, what library we want to use if we want to write tests and then you know, if we are going the follow some sort of style convention. I think for most of us, me included style conventions is not something you really take into consideration kicking off a project like you know, for a language choice or a framework choice, that sort of thing can have big ramifications down the line if you make the wrong decision in those choices, so if you pick the wrong framework for the job then you find yourself having to redo a lot of stuff maybe in a couple of months or years down the line, you regret the choices you make. We don't give the same sort of consideration to the style that we apply to our code base.

I want to explore a little bit why maybe that is and then some give some reasons why we might want to change our thinking around the whole thing.

So, with code style, every line of code that we write comes out with some sort of style applied to it whether it is intentional or not intentional. The way that I kind of think about it is, compare it to the accent that you speak with. I am from Dublin, I have a Dublin accent, I don't have to try, this is what I sound like when I talk. I can try put on some other accents, I won't insult anyone by trying to do any other accents but you know, I could, you can put in an effort to speak with a different accent but you have to make a decision for every word that comes out. You make the decision to say it with that accent. If you don't make that decision, then you go back to your default accent which is whatever you have.

So, the same thing is with our code you know, we can smash out stuff on the keyboard and then whatever comes out, comes out. Without thinking about it is going to be our default style. You can make a conscious decision to you know, name things a certain way, use you know, double quotes over single quotes, tabs or spaces or use certain patterns rather than some other patterns and some of the other things, so we can change our style but our default style is what can be used to identify us.

Who here could look at a piece of code, maybe in their work place and have a good idea of who wrote it?

Right. So that kind of speaks to what I am getting at here, you can look at a piece of code from the style of it know who wrote it. The same way as if you listened to a voice recording of someone, you would get a good idea who it was that was speaking.

So before I get in well, I will step back. Mentioned about our default style, so our kind of defaulted accent, so where do we pick these things up? How sit that we have a default style? Let's go back to talking about your accent, that comes from your upbringing, who you are around, what those people sound like and your culture and a couple of different factors in there, so again, from Dublin, grew up around a lot of Dublin people, so it is what I sound like now. I have this annoying thing where when I go places I tend to pick up accents really easily, so in the States a couple of weeks ago like less than two weeks and I didn't realise but when I came home everyone was slagging me for starting to say words with like an American twang instead of what I am usually sounding like, phrasing sentences differently. Part of me won't be surprised if I leave here with a bit of a Welsh twang! but like, so, you are able to change your accent and like over time your default accent changes to be, to match that of your surroundings. Some people it is more, more active than others, like for me I will probably change my accent in a year, where other people can go 10, 15 years and see not much of a difference.

The same thing with the style of our code, the defaults style that we have comes from our code up bringings. So where did you learn to code? What examples did you follow? If you pulled out stuff out of textbooks especially early on, you will copy one for one the examples in textbooks or stack overflow or documentation or whatever. You don't think about it. You just write it the way they have it. So, you are without realising it starting to follow their code style and then after a while of doing that, you might notice someone else doing it another way, prefers that, seems more natural. Then you start doing it that way. Then you know, 10 years later, you don't think about it anymore, then you have your own style that is it. Other people look to your code and see, how did he do this feature or this thing and then they look at your code and then all of a sudden your style is being passed to them and it keeps going, but if we can change our accents. If our accents can change from our surroundings and who we are with and who we are spending time with, surely our code style can change as well?

But it takes some hard work, it takes some like conscious decisions to, to make those changes. In this talk I talk a lot about style. By style I mean anything that contributes to what your code looks like, so this kind of encompasses like code patterns like using like list comprehensions over four loops, you know, landers, indentation, quotes, spacing around operators all of this stuff and some of the things that you might see are like some programming constructs that might have particular benefits for using them. Like performance benefits and I am not really going to touch on any of that stuff. Like there maybe examples here, it makes more sense to use a certain construct or a certain pattern over another. That is fine, but I am just when I say style, encompasses code in general and I don't really consider performance or any particular choices why you might take one approach over another.

So here is a, what I mean about patterns over style.

You can't see my mouse, that is annoying. So anyway, you can see we have a list of two dictionaries here, in the first item in the list, you are using all double quotes and it is nice and consistent and spaced out and then the second one is using single quotes and missing a space around the comma. This is like what a code linter would pick up on and say, you are missing a quote and a space after your comma and that sort of stuff.

There was actually meant to be another image there and I just realised it is missing but it was just taking those, taking that list and using two list comprehensions to pull information out of it rather, so that would be patterns, so list patterns whereas two is not necessary.

So for moving from project to project we encountered different programming styles and different code styles and there is a bit of a switching cost even though we might not realise it from going from one style to another style. So probably the biggest switching cost, does anyone here switch on a regular basis from one project to another?

Yes good for you. So, right probably the biggest switching cost is figuring out you know, how does this project do XYZ and you know, how does this project work and then try to build up the whole model in your head of how everything works. That for some people that can be really quick and for others a bit longer but code style can, can have an effect in this area as well. If you are moving from one project that consistently uses something ridiculous like seven tab indentation with single quotes inside functions within for like class variables uses double quotes or something just really not consistent then you have to get used to doing it this way. Then switch to another project and then they do it another way. Then adjust how you write your code to fit that project. You have to remember to do it.

But there is a lot of existing guidelines and tools that can help towards this. I say help, I don't know if it really does help. Some of the popular ones, PEP eight, JS, I don't know how you say this, I say go-FMT I hear go foompt, sounds like you are slating someone. Google would publish or has published a guidelines for like Python, C plus plus, shell script and even vim script. I wonder if I should make them publish a style for vim script then I looked at vim script. But there seems to be this big demand for these guidelines, a while back I changed my text editor from vim to Adam and the first thing I did was look for a linter plug in to install, as of yesterday the plug in I used has over 340,000 down loads and the sub lime text equivalent has over 600,000 down loads. So, you know, there is a big need for this. That is just two text editors, if you add in the ones like ID's, pie charm, eclipse, whatever other text editors, the fact that the text editor either with support for these linters or have some way for you to really install them, says something about the need or the want that there is for these tools.

So then, these guidelines typically come you know, with these sort of stuff, if it tells you how you should name things in your project. Should you use globals, should you use landers, indentations and that kind of stuff. Obviously the language specific stuff will change from guide to guide but the general content is the same.

What I have noticed over time is from going from project to project projects will take like a base set of guidelines like PEP eight then they will say, oh we stick to PEP eight we along lines, once they are under a hundred or whatever. They will change to suit the organisation.

I don't like, has anyone submitted a request to a project and then someone commented saying, oh the style errors in your code, your commit message isn't right or missing a quote or a comma and that feeling of range that comes over you and you are like! (LAUGHTER), you can't believe that someone has been picky enough to point this out. Then you curse yourself for not noticing something so small and you kind of begrudgingly resubmit with the changes. It does happen to me loads of times. What I notice is, seeing it happen from other people, people take criticism of code style readily from an automated tool rather than from a person. Yes, I guess you guys would agree.

So like, I wondered why it is like, what is really the difference between a tool telling you and the person telling you the same thing? Does it come back to what I talked about accents? If someone started ripping off my accent, I would get annoyed. I can't do anything about it really. Then someone is then ripping the code I write and the style I write and then you know, it is like a personal attack sometimes.

Sometimes it is this applies to a lot more, code review in general, we can address it in the way we say things but what I am getting is that having a tool in place to, to inform you of these errors is extremely valuable because it removes that possible area of confrontation that could come from a person making a comment.

Let's do like a quick survey. Who here consciously adheres to some sort of style guide., when they are writing a code, is this compliant or whatever? If I notice a libter error, I will fix it, other than that whatever comes out, comes out. I am probably one of them at the moment.

PEP eight never, this is my code! yes!

Okay. So most people are sticking to some sort of guideline, are there many who like have used some sort of customised guidelines. Customised guidelines? Not very many, it is base guidelines.

I found this interesting from the PEP eight page, under the rational about why it exists and says one of the key insight is that code is read more often than it is written. The guidelines intended to improve the readability of code and make it consistent across the wide spectrum of Python code.

I didn't read PEP to but it says ... consistency within, with this style guide is important consistency within a project is important, consistency within one single module or function is the most important.

I think this is really good. I hadn't read PEPeight when I prepared for this., I looked at the tools I had installed but I think this is really important. Yes it is important to be consistent with PEP eight as a whole. But more important to be consistent within a single function or module. Then it decreases in importance from there. Who has ever looked at a function and seen on one line it is using single quotes and then the next double quotes?

Yes. Mainly see it in java script projects I don't know what that says?

So, I went looking for some inconsistencies in projects to have some examples, I went looking in Django first, I thought that would be the most relevant and Django's code base is pretty good, annoyed me, was hoping to find clear examples and go ha you are all inconsistent and then you would hang your heads in shame and you would fix it. Django is pretty good. There is a component calmed form tools, called form tools and that got pulled out recently, pulled out, this is where it has come from. These two methods are inside the same class actually, so the dock strings in one is using the triple quotes and then in the second one it is just using single set of double quotes.

So, these sort of inconsistencies is what stands out. If you are looking as a new contributor to your project, okay, how does this project lay out its dock strings because dock strings are going the be a big thing. Some document all the parameters and all the form types where others have dock strings that tell you what the method does.

So, you kind of look to see in a code base what kind of convention you should follow and if that convention is not there then you are just going to do whatever comes naturally to you.

Here is another one from forms tools. So, the first example lines up arguments with the first argument, second one puts them on a new time and yes third one puts them on the same line as well. And I just realised there is a big syntax there at the top of that but we won't talk about that ...

Another one from PIP. In the method declarations so all the arguments are stacked but then there is one line where they're not stacked and in the send method they're all laid out but then there is one on a new line so I guess the last one was made like that because some tools said oh this line is longer than 79 characters and developer went huh and went enter and put it down and that's how that was there and that leads me on to when is it OK to break these guidelines because it OK to break them I talk about pep 8 because it's a Django conference but other guidelines say - they don't say you must stick to this its important but it's most important about readability and consistency.

Here is an example from Django's security middle ware. The first example is actually how it is in there at the moment so it's just that list comprehension all on the one line and I just broke it in the second one so that it didn't go that long and personally I think the first example is way more readable. Does anyone agree with that? Anyone think the second one is more readable? Well, you are wrong. {Laughter} I am joking! I'm kidding!

But this is a really important thing to know. Good looking code means something different to different people and we'll talk more about that in a little while.

But the important thing is use your judgment. If you think it looks better the first way and you can stand by that and say yeah that's more readable do it that way if it comes a second way and you are like you know doing it this way it just makes it a little bit a uglier so I would rather have the pep 8 and break it on 2 lines. Just use your judgment you know.

There is another one here so the second type error towards the bottom goes pretty long. So, could have broken that string open put it over 2 lines but strings especially I find really hard to read when they're broken over a couple of lines. So I'd say in that case it's OK to go against pep 8 but in the first type error you see there, they use single quotes; then the second use double quotes so that's an inconsistency that could be cleaned up. It's not massively important doesn't make it less readable but if you want to be consistent across your project these are the little things we should look out for. It's not just little projects. Projects like I'm the only one committing to and I still somehow manage to come out with different conventions of doing things. I think I must have an alter ego who writes code when I am sleepy or something.

This is some code from the text setter that I'm writing. This is 2 examples of a match expression in Rust. So in the first one I've lined up all the little arrows nicely and in the second one I just didn't bother I think the first one is nicer. Some people might think the second is nicer but I haven't been consistent in picking a style and going with that. It's see what happens when I hit the key board.

I did bring notes with me but I somehow have jumped way over everything I was going to say.

So take this example so like it's not consistent. So you ask the question, I've spotted it now, should I go back in and up-date it and clean it up? So it kind of poses the question like: will it help with read bill readability and if so it might be worth doing it. If not and you are doing it for the sake of doing it, probably leave it alone.

Anyone heard this before? Write your code as if the person who will maintain it is a psychopath who knows where you live?

I think code style has a lot to feed into this. Who has inherited a project before and opened it up for the first time and just gone like, yeah ... I've been there... {laughter} and then you wish you knew where that person lived {laughter}.

It's important to make our code readable for those who will inherit it down the line. The general thing that I stick by is if someone else is going to work on it take that bit of extra time to make it look good so if you are like setting up your project at the start, take those extra few minutes to decide OK this is what code is going to look like in this project and then be conscious of that when you are writing it.

I talked about pep 8 and JShint and whatever - they layout guidelines then they give you a tool to run and it will spit out references to specific areas in your code that tell you where problems are.

There is this other group of tools which go FMT comes under. Has anyone worked with go? A couple of people. Go has these style guidelines like any other language but it also gives you this tool called Ggo FMT that will run over your base and enforce your style guidelines so if you write your code all over the place doesn't matter, you run your tool over it and it will format it correctly according to the guidelines so the idea is you have consistency across all go projects that you can move from one go project to the other and the code will look the same like all functions will look the same, be the same length, lines will be the same length, all that sort of stuff so it removes that switching cost that comes along, but then it doesn't leave you room to deviate from that so you are stuck with whatever they give you.

And to be honest I prefer this approach like personally I would rather just write the code and let some automated tool take care of formatting it and making it look the way I want it to look. I don't really want to have errors popping up or my editor saying this line is too long it's really annoying and I'd rather it wasn't there at all so if we had this tool we could use to do the work for us so I could get back to writing the code and not have to think too much about single or double quotes like really it's not the most important thing in this scope of a project, and so when I started working on rust I started working with rust like this time last year may be when it was - long before the better version and long before 1 op0 release and APIs were changing like every day like literally every day I'd open up the terminal and have to rebuild stuff because certain projects hadn't updated to the changes that came out in the latest nightly, and it was just an absolute mess from a maintainers point of view, but - and there wasn't any guidelines around style at the time because the language was so new and it was still changing quite fast that no one knew what rust code should look like but then as we got closer to the data and documentation got built up some people started talking about style guidelines on IOC and read it and I was kind of just a witness to the whole thing I was curious to see if they'd go down the line of just providing a set of guidelines like we have with pep 8 so it's like this is what it should look like or if they go down the route that go did where they give you a tool where they format all your code for you and they went down that road. Like there was plans for the Google summative code to work on a rust FMT project but then Mozilla didn't get into code so that didn't happen directly, but there are people working on this at the moment and I think that says a lot for what developers want that this is a new project a new language that started with nothing like basically so they could do it however they wanted it and this is the way they chose just to give a tool that will format your code for you and that's all you have to worry about.

There is a similar thing for Python from Google and I don't know if this is meant to be said as letters or if you can just say it but it's all Y APF and it does a similar thing to go FMT that is, you can run it on a code base and it will apply some - it will reformat it for you and you can override certain things so you can say lines can be this long. If you want to use pep 8 as a base and then change these things and then you end up with code that looks the way that you want it to hopefully.

But like I said, code looks good to some people might look awful to some other people.

I'm sure you've seen these sort of different variations of code. So first one has the opening brace on the same line as the function. Function name and the second one has it underneath. Same thing with the if statement, opening brace on the same line then underneath.

Personally the first one is what I would do. Second one drives me insane.

Before I started working on the text editor project I had a look through some of the vim and e-max source code, seemed like a good place to start, and it quickly realised I could never be a C developer because they all use the curly brace on the next line thing and I just couldn't cope {laughter} something in me just gets really angry when I see this and I just have to fix it and I have a feeling it wouldn't be really appreciated.

Same things with declare and variables this is all the way I have seen Java script variables defined. First way is how I do it and other way is how I've seen people doing it line break after the var key words commas at the end then at the start or just loads of var key words and there is nothing inherently wrong with any of these ways. It's just what looks good to some people doesn't look good to other people.

Again with the rust match statements there is lining up the arrows or not lining up the arrows. I've seen the same thing with variables, lining up the equals with your variables so you have this nice long line of variable definitions.

And so the formatting tools that are there and the ones that are coming like the one for rust and the app from Google, none of them are perfect right so I am yet to find a person who has written perfect software, people who think they've written perfect software but not people who have written perfect software that works every time and these are the same so there are times when it will eat your code and it will break at you have to look through DIF to find exactly what it deleted and add it back in again so it's going to take time to get these tools in place so we can use reliably on our code. So until then, I think we need to challenge ourselves a little bit because especially in the open source community, such a wide variance of projects and the associated styles with each other's projects and it's just really, really inconsistent. The analogy that I heard once was you know if I was a builder and I just built a big 4 or 5 bedroom house and from here it looks great but then as you get closer you start to notice like that wall has a bit of a bulge going there and why is the floor like this and you know so you start to see these little things, there is a crack there, it's missing a brick like stuff that you know - you don't see it from far back but as you get closer you start to look, you see these things and may be is the same thing with our code that style errors or style inconsistencies are easy to fix and if they're not fixed, well, does that show a lack of attention to detail, are there things further down that may be have also been ignored, some more complex things may be and it's just something to think about, probably if we're following all these other things like test driven development or even if we have just test coverage chances are we won't miss big things but how your code looks is like a window into the code base.

Like I've done a couple of events where I've been running the live sound so like missing a band - running the actual event for when there is a speaker up like this, then finishing out, so the mark of me doing a good job, what I've seen, the mark of me doing a good job is when no one mentions the sound at all if the sound was really bad and the band didn't sound good and there was feedback all the time people would say oh God the sound was terrible, like so distracting whatever, but if it was good then they talk about everything else that went on, they talk about the band, they talk about the speaker. And it's the same with code, like you open up your editor and you are oh spaghetti code, or it's like this is awful, but if it's good code you don't talk about the style, you are just like looking at what it is doing rather than what it looks like.

Yeah so as a community I think we could all benefit from a step up in readability and consistency its not to guilt trip anyone into saying the way you do it is bad and you should do it this way but it's just to be more conscious of it as we're writing code. I'm sure even this week, I know this week I've written code that looks awful, but we don't intentionally sit down and say am going to write some bad ugly code today, you just sit down and write some code and it comes out the way it comes out but it's to add extra time in there to be conscious of it and the chances are may be no one will notice, may be no one will notice you've taken the extra 30 seconds before you do git commit to format things properly, may be no one will ever notice that you've put in that extra effort, but maybe that's the point: if no one ever notices, then maybe they also won't notice ugly code because you've taken that time to put in and to clean things up and make it look good. Because your code speaks about you and you as a professional and you as a developer.

I don't know if you guys have seen this at home there is this ad that will come on in the cinema before a movie and it was like an anti-piracy ad and would start off with all these flashing images then this big text superimposed like: you wouldn't steal a car and everyone is no, no I wouldn't steal a car. Then it was I wouldn't steal a hand bag, of course not I wouldn't steal a handbag. Then you wouldn't steal a TV, no I have a TV, then you wouldn't down load a movie and everyone is shifting uncomfortably in their seats going ooh no definitely not ... {laughter} I'm sure no one here has down-loaded movies but {laughter} may be some of us have written ugly code. But we haven't it is not an intentional thing we go out to do. You wouldn't write ugly code, well you did. What I am getting at is, we need to be conscious of this, as a community I don't think we are. So yes. Whether it is PEP eight or you know, CSS lint or JS lint or go FMT whatever you want, I would encourage you to do is to be conscious of code styles next time you open your editor, don't update all your repositories in one go, be conscious of it and then gradually we will see this, we will see this increase in quality and increase in readability. There is no one who won't benefit from that. So that is all that I have. Thank you for listening. (APPLAUSE).

NEW SPEAKER: We have time for one short question?

So whoever is first there.

FROM THE FLOOR: It is easier to come when it is one last question, you said sometimes you didn't want to get the line and I agree with you, especially when there is one character and how do you under this with automatic linked in? I know that you can add the ... at the end of the line, but it is not really a solution in that case?

GREG CHAPPLE: There are, are ways to, with ... you can do with certain lines, ignoring it is not changing the guideline. I would say if it is one or two characters over, for readability it is not going the make sense to cut that and put it on the next line, so keep it there. Tools like the auto formatters need to take it into account that we might want to override these behaviours so like, if it is one character over, you know, don't break the line, you know, take into account I want to preserve readability and that sort of stuff.

FROM THE FLOOR: Okay thank you.

NEW SPEAKER: Thank you very much.

(APPLAUSE).

DANIELE PROCIDA: Hi folks just a few very last things before we head into the last lightening talk of the day. Peter Finch is in the house, it is finally a beautiful summer day, you will enjoy the walk even more than you did the last two times. So, let me find the right place on these slides. Let's try that again.

Don't forget we have the prize raffle the box is going around the room. Be generous or at least be fair. Pablo has taken away his arrow, we will have to use it by Python script.

There is long board from, books, a plural sight subscription and ... the original watercolour art work that Ola painted for her slides. Yes.

Those of you wanted to collaborate with the universities wellbeing service, there is a meeting tomorrow at 12:00 o'clock at Cardiff University main building in the foyer by the statue. Somebody passed me a message about the Django for social change or Django for social good group that is forming, an e-mail, I can't remember what the message was, can you remind me?

I was supposed to say something? Okay.

There is a Google group started. And we are going to twitter the link thank you.

Don't forget Cardiff print workshop are out there with their art works and you can have a chance to at least say hello, thank you for all the work they did for us, help yourself to it, put a donation in the box.

Last thing: anyone who has seen me at a conference over the last couple of years will have heard me going on and on about a proposal to help start a Python conference somewhere in Africa. Last year Prof. Judith Hall from Cardiff school of medicine where I was working approached me. She was one of my colleagues one I particularly liked, she was a trouble maker like me and she said well, we're doing this project which might be in Namibia and I have hurt about what you are doing and suddenly the project took off and it actually happened in February this year with the help of the Python Django community the Dutch Django foundation. DBO. Git hub. Django France organisation, Providence - is there in the audience. Daniel {inaudible} activate, so we have fantastic support with these people coming so thank you for the support in the community; because we left London in the start of this rather miserable February in the middle of a rather long and miserable winter and we met {inaudible} and Tom from activate in London and in Frankfurt met Loek from the 1 per cent club in Amsterdam. Then Namibia. Met Jason and Shai in the {inaudible} and we had 70 or 80 people most of them students from the university. It was organised like any other Python Djangocon. We got to find out what kind of things they were doing the computer signs and other students. Some of them were doing amazing things. We met one who was collecting light pollution data from sensors in the desert in order to help establish a proposal for a telescope array. There was one who had built an exchange of goods web-site so people could give away things they didn't need. We met we met a student who learned Python on the first day and on the fourth day gave a lightning talk with a live coding demonstration in it. {Applause} yeah.

So we all worked really hard and we formed some very valuable relationships. That is the very important picture. That is the moment that the Namibian Python programming society was formed.

So, everyone who was in Namibia please stand up. Loek, Jessica Maria, Michael, we've got 3 of our visitors from Namibia, there is Jessica who is the chair ... {inaudible} {applause} ... Michael, Maria, Jessica ... {applause}. So we've started something. Jessica is the chair of the Python Namibia society who did the live coding demonstration. Yeah. {Applause}.

And they want to organise something for 2015. It's for them to organise it not for us but we as a community will be there. I hope you can come if you are sick of Europe's winter perhaps by then, you might find that you have a really lovely week in a really lovely city with really lovely people. So, if they supply the will and the energy to organise his, the rest of us, many of us will be there to support it and I hope to be there. So maybe we will see you in Namibia for another Python conference. Thank you very much everyone who came, everyone who was there. Thanks. {Applause}.

Are we ready for lightning talks? Who is our host for lightning talks? It's you and Baptiste OK I leave it in your capable hands and I'll come off.

VINCE: So we are going to keep the number of mac books that die to a minimum today. We had so many requests for lightning talks we're probably not going to get through everybody but we're going to start off with people who've really made a big component at the conference and this is Sheryll from the speech-to-text transcription. {Applause}.

DANIELE PROCIDA: It's a mac book but it's running Microsoft windows {laughter}. I didn't even know that was legal!

VINCE: All right we're going to start with someone else. If you give a big hand to Ben who has been one of the volunteers who has worked tirelessly throughout the week. Big hand to Ben and everyone else too {applause}.

NEW SPEAKER: Hi everyone, my name is Ben, I volunteer here and I don't know a huge amount about Django. So yeah I am definitely a beginner but what I'm wondering is why I who'd bother using Django because I know that's a slightly controversial thing but I want to tell you a bit about me and what I want to do with web apps and may be someone can after this give me a little few reasons why I should be using Django. So, my name is Ben like I said. I want to build web apps and have been building web apps. I'm a medical student so that means 2 things I'm not. I'm not a doctor - yet - and it also means that I am not a web developer. So don't be - just try and understand where I'm coming from with these things.

What I want to do with web apps is build prototypes for some of the ideas I come up with that I think might help patients or help me or help other doctors doing their jobs and their training. What I am looking for with a web framework is something that lets me develop things really quickly and I know it kind of controversial here but something that is more magic and less code is also good for me because it means I can get the same thing done quicker, I'm not that bothered about what's going on underneath. So, I tried Django because I started learning Python in May last year thanks to some guys from potato who encouraged me. But when I tried the tutorial the tutorial was I thought really hard, there was lots of components I didn't understand, a lot of assumed knowledge even though I felt like I had a good grasp of Python there is a lot of stuff missing there that I didn't know what was going on so I didn't get any progress there. Someone suggested web to py to me. Kind of some sniggers in the crowd, and Python any way, I know the guy who runs that is here somewhere so awesome for making my life easy for that.

So, I actually did start building web apps with not too much difficulty. It was really easy for me to get started and I didn't have to configure anything thanks to Python anywhere goes. It's a web based IDE, yeah that's probably not good for you guys but for me it meant I could pick up any machine anywhere in the hospital on my own, even on my phone and be writing code wherever. The documentation was awesome for me and easy to read and MVC actually meant MVC so a view was a view and a controller was a controller.

I'm going to show you something I made - I have time? 2 minutes. Awesome.

So it is really like boot strappy, so just be gentle ...

Doesn't come out that good.

What I built was a sort of on-line notice board where student can sign up to sessions that are going on in their local hospitals to keep them busy and make sure they're always making the most of their learning opportunities.

Students can just see a list of sessions going on there.

These are sessions that the students have been to in the past and sessions they got coming up. People can edit sessions and it's an undergraduate component for helping the team in hospital manage students as well. I build all that in web py and PDFs as well and what I found is that there wasn't anything I couldn't do using web to py and from my point of view I really enjoyed it because I could go from an idea to a prototype in just a couple of ideas but having been here all week I am seeing that this Django community is absolutely massive and I am pretty sure there is something I must be missing.

So if you think there is something I'm missing, come and give me a shout and let me know what I'm missing out on really and how I can get involved with Django with a bit more success. Thanks. {Applause}.

NEW SPEAKER: Sheryll: unfortunately I'm using mac book with windows on it because my software will not run on a mac side unfortunately so I'll just give a brief overview of what we do. You've seen the output on the screens, we travel round the country and we support deaf people mainly in AGMs, work meetings, round table meetings, conferences I have a top 3 of hardest conferences to support and I now have 4! {Laughter} {applause} It's been a pleasure actually supporting you guys and I am so pleased to see so many people using the screens.

Basically we're working - it's machine shorthand so we're working off machines, I'll show you this ... there you go. So we can't get it on the screen but it works by a system of phonetics. So, you have your key board and my colleagues key board works in essentially the same way. So we have a bank of consonants on the left, we have a bank of consonants on the right and the vowels down the middle. So to do a word, you are playing it like a piano. You are pressing chords of letters all at the same time so each key is a letter. So you start learning the key board, you start learning where the letters are and then you start making short words like"" bat, rat, cat" and then you go through a process of learning how letters, by putting them together with more letters make more letters - stay with me! So the bank on the left here, these 4, if you press together, you'll get a J. But on their own, S is Is. K is can, W is with, and R is are - if you press them on their own. If you press them together you'll get a J. {Laughter} ... as you do!

Then you have the vowels. You have AOE and U. You don't have an I. You have to press the E and U in combination to get your I. So if you wanted say judge - bearing in mind what I said about the key board, you can do bank of consonants this side and the other side so you have an alphabet on both sides. To do the word judge, you would press those 4, U from the vowels then this block of 4 here is another J, so to get the word judge, you actually are pressing - you are pressing 9 keys but you press them all at the same time so it is like playing a piano.

So, you can - unfortunately I can't show you the software which I know you would be really interested in, so the software matches the chords that I'm pressing on the key board and it matches it to a dictionary entry on the software and you can pre programme words but you can also get things wrong - during the conference we've heard the term whiskey! Thinking that's good ...! Then we realised that whiskey was not W HIS KEY - it was wsgi! So we can alter our dictionary then to reflect that when we type whiskey we want it this way (wsgi) so it is spelt correctly. So you can do clever things with it, unfortunately in a lot of conferences we do have speakers get up on the podium and say Aha, I am going to have some fun. So they come out with words like - Llanfairpwllgwyngwllgogerychwyrndrobwllllantysiliogogogoch (Welsh!) {Applause} Now, I told Hilary we were going to do that so she pre- programmed it. And I have a whole bank of really long words that I can insert so if a speaker decides to try and catch me out I can say well forget what you said, try this. So it's very good. You can pre programme a lot of stuff which is why when we go to conferences we ask for as much information as they can give us so that we can get terms.

I did a literary festival the other week and there was a section all about Shakespeare, so I got on Wikipedia read around the subject and put as many names as I can into my dictionary.

So, to have an idea of speed, people would say why do you use electronic shorthand? The reason we use it, on a normal lap-top key board you can only get to 100 words a minute if you are a fast typist. On our key board we can do about 250 so you can pretty much keep with most people apart from the gentleman Russell - I do apologise! {Laughter}.

So we do have limits. It is human interaction. So {laughter} and I think that is it I think my time is nearly up. {applause}.

NEW SPEAKER: Sorry... it's flashing in weird ways ... let's see, no wrong screen ... sorry it's flashing ...

So, it's not best way to view this but I just want to say 2 years ago round these days I was in Warsaw and I was under a circus and I met with the community and on my way back I told why not try and do something like this? Not like that but similar in Florence. And in the start of July, we organised an event in Florence which was called Django beer. I recognise some faces because during euro{inaudible} and I guess a lot of people attended and I guess they had nice time despite my very long talk but any way.

This year we rebooted the event and had monthly meeting about all things web so not just about Django but anything web related and we are - sorry very low resolution - we are having talk of very different kinds from how to be a bad developer, HTML circumstances but I love it or things like discovering Django rest framework from theory to practice and I guess this is first time for Python related events even beer chasing lessons from a real brewer. That's the key of Django beer, we love Django and we love coding in very many different ways but we also love lot - especially me - beers and especially craft beers and enjoy beers responsibly so we have street code of conduct, this is not event just to get drunk but to code and to taste beers at the same time and I especially say beers because my beer chasing guru which is one of the best beer taster inactivity always say there is no beer there are only beers, every beer is different from the others and you see it's this guy so he must be serious, he must be. With this face he is a serious guy!

In my opinion we organise this because I see very many matches between code and beer both thrives with a good community, both thrives by people experimenting with it, tasting S exchanging ideas and are very good at socialising people and get people together with the same - people with the same attitude together.

So, if you happen to be in Florence at the end of each month last Tuesday of each month, go - visit the web-site and see if you see, if you have any events, any talk interesting or if you come back just propose your own talk and we'll be happy to share some talk and beer with you. Thank you so very much. {Applause}.

NEW SPEAKER: I hope hits not been killed ...

VINCE: What is the chance,.

NEW SPEAKER: You never know ... oh!

NEW SPEAKER: Don't move!

VINCE: OK next talk is Ana then we'll have Geraint ... {applause}.

NEW SPEAKER: I would like to spread the word about systers. Currently we have tens of thousands of organisations that encourage women to gain technical skills to name a few Django girls {applause}. Py ladies {applause}, girls who code, rails girls, there are plenty, you don't need to clap for everyone because there is seriously like a lot of them, rails girls, the other initiative, girl development, black girls code, girl geeks, she plus plus, girls in tech and those are just a few I know {laughter} believe me there are many more and this brings me to systers a community that unfortunately is not very well-known here at the Django con. So systers is a community which is part of a big organisation which is called the Anita Borg Institute, it's a community for women involved in stem - science, technology, engineering and maths; it welcomes women technologists of all ages at any stage of their career or studies. It has over 5500 members. Spread in at least 60 countries all over the world. It is indeed the world's largest email community of women in technical roles in computing. It was founded in 1987 so it is pretty mature by a fantastic woman Anita Borg an American computer scientist and it started as a small email community with 12 members and grew to what we have today. It's very simple to become a member of systers community. Anyone who identifies itself as a woman can join the mailing list so ladies, seriously, all it takes, you go to this web-site and subscribe to your mail map. The vision of systers is the following: increase the number of women in computer science and make the environments in which women work more conducive to their continued participation in the field.

Over the years systers is us private space for women to discuss the challenges they face in their professional lives, make good career decisions, even find a job or collaborate on a project, receive guidance for difficult professional situation or simply share the joy of being part of a world community same as we feel here at Djangocon and it's not just - seriously take a look at the mailing list and you'll see a lot of support. Go to the systers web page and you can read more information on it. Thank you {applause}.

VINCE: After Geraint we're going to have munier(?) so if munier could come to the stage please.

NEW SPEAKER: OK so Geraint.

NEW SPEAKER: Hi. I apologise for the rubbish title, I gave this talk 3 months ago and it had a much better title. I'm so I am a PhD student and I wrote my first line of code 1 and a quarter years ago so here is my journey to code. So in February 2014 I wrote my first line of code of VBA because I was studying a module of my masters and I could do loads of stuff in VBA, a 4 loop and an F statement, it was brilliant {laughter}. So zoom forward to October, I grew up and I discovered Python. So, {applause} - this is the only bit that counts! {Applause}. So what were the big turning points that changed me from thinking a mac was something I ate in MacDonald’s to being stood here in front of you? The first thing I did which was really important I taught Python to other people and I didn't have a clue what I was doing. This was the best way I could teach myself Python was making sure I knew enough of it to explain to other people the concepts and this taught me all the basics and I can't recommend it enough.

Next, I googled tings, I discovered that the internet is full of all the answers I could ever want.

So whenever I became stuck on a bit of code, or, whenever a student asked me something, I didn't know how to teach, I Googled and I always come up with the answer because this was a sort of basic programming at the moment.

I was still using idle at this point, so the next thing I did, I found a nice text editor, flipping between area Tim and sublime, some people like to use vim, I am happy with this, part of this is learning how to command line worked which sounds really basic but it revolutionised the way I wrote my programmes.

Next, I was most proud moment of my recording career, I installed my first library, this is before I knew what PIP was. Then I discovered PEP and then I could install as many as I wanted and discovered that if I googled something, the answer was usually, find the library, import from bla, bla, from bla, bla, I didn't need foo write code it was lovely.

The next thing was a step up, I pair programmed so with my PhD supervisor Vince here, we sat down and wrote a simulation of a queuing network, this is where I learned vexer control, git, testing, documentation, I learned that just writing code that is not structured into classes is nothing, and it is not really code.

This is where I went from yes, I know what code is, to actually I think I am writing something that is quite nice.

Then I what I have got labelled as discovered as a mathematician, whenever I come up, I don't come up with them, when I read about new thee rums and algorithms, coding you rip to pieces the maths and you look at the pure logic that really helps me in think maths.

Then, as you heard before with the invitation of Daniel, this is first time I met some of you lot, I discovered this there was this big community and editor opened up lots and lots of doors for me, led me to be on the committee of Djangocon Europe and me talking in front of you today thank you (APPLAUSE).

NEW SPEAKER:

No mac is have died so far on my watch.

NEW SPEAKER: ...

I really wish I didn't say that now!

The hello everybody. I am from Tunisia, this is my first Djangocon here, I have been a software developer using Django years and now working in Germany, I want to talk, why do we need to update and upgrade our Django project with the latest version, so, first thing why update matter? I think the first thing is the security issue as we see, I have seen many talk about security and it is an interesting topic to worry about and the second thing is the bug fixers on Django project and, also the new feature that get added on every version.

For this security, I think many talks, talks about the security issues, so, this is only the list from Django project.com for the security updates for 2015 and we need to consider the deadlines as you see, even the Django one for which is the long for release, will not get new updates since October so I think many people need to consider upgrading.

So, the good news is that current Django version is long time support so, I think it is a good thing. Another thing to consider is the Python version, so if someone upgrade from Django 1.4 or other lesser version you need also to consider to update Python from 2.6, if you are using 2.5.

We need to preparations, data backup, data migration, updating, troubleshooting.

For preparation I think, we need to read, for every version carefully, the release note, the deprecation list and the new git branch or fork or copy of the project and create a new environment with all the upgraded Django version.

For example, some removal in Django 1.8 like the ... the function closed connection and now C port for ... so all this are well described on Django project.com.

For the data back up like the databases, the source code and the ... we need also to consider that is now integrated in Django 1.7 and there is small steps to be able to upgrade and for the code we need to replace the replicated function replace and available packages and you will have to write if you don't have because it will be easier.

You have to troubleshoot your project if everything is okay, you can be able to deploy my advice is, also to try deploy on other than the life server because maybe they some update you need to make, that are not related to like con fig files organisers something.

Also it would be great if you can support Python three, I think it is a good idea and there is also a tool called can I use Python three, show you like the requirements for it, your package is dependencies are the supporting Python three or not.

You can also plan to support, if you are maintaining the third party app, to support both Python two and three, it will be a good and nice idea. So thank you very much.

(APPLAUSE).

So this is the penultimate lightening talk, after that we will have Russell.

NEW SPEAKER: Here is Baptiste.

BAPTISTE MISPELON: Hello, it is me again. Okay, it is time for Pythonic crime story. First off disclaimer, this is based on a true sometimes.

Some of the people involved in the story are here in this very room but I have removed their name to protect them.

So, let's start with the crime scene. This as you see, is the normal Python class, two method, foo, returns, hello and bar returning world.

But let's see what happens when you try and run this code?

First off we have the class, foo. Then it is foo method, everything is okay as expected it says hello, then we are called bar method. What is happening now? Oh no. The object has no method bar, there has been a murder!

(LAUGHTER).

Let's investigate and go back to the crime scene, well the method looks like it is there, this is the code.so, what happened?

Well, yes, this is the method that has gone missing, luckily, we had a well-placed logger and we managed to get a picture of the killer. There it is. You can't see really well. Let's enhance, ah, much better!

You see though (APPLAUSE).

The problem is that the killer is invisible. But, let's try, the mug shots to our face matching database, yes, we got a hit!

Let's get the Interpol file. Last name, T, sorry. First name, back slash some weird Polish name. Number nine, ... back slash ...

Also known as tab by.

Python quiz time. How many spaces does Python see in a tab in eight, correct. The answer is eight.

So. How did our tab do it? You see, this is what we see but, we need to turn on wide space vision, sorry.

Should turn on white space vision.

So then, please turn on Python vision, Python sees this, you see our method there is our code, it is sitting after the return statement.

So, that is how tab did it. So now you are wondering this looks dangerous, what can I do to protect myself from this killer tag. First, Python has significant wide space, what should you do with it? Highlight it.

Get it? Yes.

So, then you can turn this, into this and there you see the problem.

The other thing you should be doing properly is stop living in the past. We are in 2015 if you are not using Python three you are missing out. If you are stuck in the pass ... Python two has flags, fixes tabs and spaces into warnings or errors.

What is the moral, don't let friends ... tell their friends use Python three, friends use Python three!

(APPLAUSE).

One more thing, I see I have some time? No?

Django girls T-shirt are back on sale. The picture is not showing up, but if you have seen, this is what a programmer looks like, you can get them at [on screen]

Thank you.

RUSSELL KEITH-MAGEE:

NEW SPEAKER: Final speaker is Russell.

(APPLAUSE).

RUSSELL KEITH-MAGEE: Hi everyone, I am here to give you a gentle introduction to print r sprinting this is not what I do to torment the live transcribers over there. This has to have a minimum of 10emoji there is one.

A sprint is a collection of people, who have finish add conference, turn up a space, to basically, basically hack on code, there is no specific plan or specific activity that is going on but, they get together and hack on some code. We have got a sprint going on in the next two days following the conference.

So, what do you do when you are at a sprint? Well all sorts of things need to be done, doesn't matter what level of experience you are at. Bugs that need to be triaged, Django have a bug tracking system, people report them we have no idea if they are seeing real problems or if they are smoking really good drugs, we need people to tell the difference between the two.

Some of them will have patches associated with them. Attributed by someone in the community, we need someone to write the patch and review it. Sometimes people will contribute to a patch, this fixes a problem, they won't have provided the tests won't come back in the future, we want tests, Django as a procedural thing won't accept code unless there is a test to back it up or.

We need documentation, we don't put anything into Django a new feature unless it is room to im...

The complaint that normally comes from people at Djangocon Europe, I don't speak English, I can't write -- that is rubbish, don't get hung up on the exact text, the hard part is not the English, we have, have many, many English speakers some of them from England, that can review the text, what is important is working out the example we can do, how to structure the document takes, if you can write the first draft that is a huge help. If you are a begin, you are the best person to write it, you know what you didn't know when you started, we forgotten what we didn't know.

What should you work on?

You can work on Django's core, sorry, there is obviously Django exists and you can ... you can work on the libraries around, so there is a huge community of libraries that support Django in various ways, also have bugs, they also have features they want. They also have things they want to get done.

You can also work on resources, Django has a website, we have things we like to do on the website., come and talk to us, we need warm bodies to make those things happen. Completely aside from the official website there is, is in... resources as a community, if there is something community should have as a website, build that website. Get it out there.

So, pick an area that interests you, look at the open tickets and scratch whatever, the view particular that you have.

If you are a begin, report a bug, triage something, test a branch or a feature, if you are an intermediate user, write a patch, eat some dog food, take a feature and try and break it. If it is a new feature, write some documentation, write some tests for that feature.

If you are appointment advanced user, work on a big ticket item, see if you can ... you have a lot of the core team here, we are more than happy to kick around design ideas or see we can shape where you are going, to see if you have a feature that is big, big in 1.9, maybe propose something controversial.

Any questions, ask a core defined, the core team is here, we help you, you help us make Django better, that is ... ask the person sitting next to you, we are not all heroes, everyone's opinions matters we need everyone to contribute.

There is an IC channel for hash django sprint. This is supposed to be a fun activity, come along.

If you need any questions, the docks have a contributors guide, link at the bottom of the page, please read that and that gives you a good place to go.

Don't be afraid to commit workshop, if you are looking for training on how you get into the Django contribution process, there is a, a workshop that Daniele is running to help make that happen.

NEW SPEAKER: And two more sentences because Russ forgot to introduce me, we are also running a sprint for Django girls you may not know this, the tutorial for Django girls is maintained by the community and we encourage you to work on the Django girls website and Django girls tutorial, if you are beginning, it is better, just find us, it is me, Ola, you will find us thank you.

(APPLAUSE).

DANIELE PROCIDA: Does anyone have a kitten to sacrifice for their gods before I plug this in?

We are okay, I strangled the kitten earlier!

So. This has been Djangocon Europe 2015. We want to tell you about, we want to tell you about Djangocon Europe 2016 and the tradition is that next year's conference organiser will be introduced by last year's organiser, so Loren and Mathieu ... please.

(APPLAUSE).

NEW SPEAKER: I am sure you are all dying to know where it is going to happen right? How many of you ready to pay? Just joking.

Next year, it is going the happen in the country of emoji lover, can you guess who that is going to be?

NEW SPEAKER: Russ?

NEW SPEAKER: You want to come around, so it is going to happen in Budapest and I think Baptiste has a few words to say about that.

(APPLAUSE).

BAPTISTE MISPELON: Thank you Mathieu, so yes, the secret has been revealed next year, next year will happen in Hungary, in the capital city of Budapest, I have assembled a team of the finest experts on emoji out there and I am completely certain that the conference will be a huge success. I didn't really prepare for this, but, I am going the be really happy to see you all next year I hope and enjoy the rest of the conference, I hope to see you at the sprint and I hope to see you next year, thank you very much.

(APPLAUSE).

DANIELE PROCIDA: So, we will see you in Budapest? Is the machine ready? It is time for the raffle. So, do you want to press the magic button?

NEW SPEAKER: Prepare the machine!

DANIELE PROCIDA: Stand up here so people can see the.

NEW SPEAKER: It is a fix!

DANIELE PROCIDA: It is not a fix. This is the Chairman of Djangocon Europe 2015. Ilya, I think he has left. Sorry.

Jessica, yes!

There is a the board, the watercolours, there are high performance Django book. You would like to go for the book?

(APPLAUSE).

Maria.

(APPLAUSE).

It is not a fix.

This is, really random things often do not look random!

NEW SPEAKER: Look good on a long board?

NEW SPEAKER: No I think I will compliment Jessica's book, I think I will ... (APPLAUSE).

DANIELE PROCIDA: Oliver.

You did fantastic, what you would like to help yourself to?

Thank you very much.

Thank you.

Also have a 500-dollar plural site subscription.

Marosch. Duncan.

Duncan parks. No.

Ben Cole.

Alexis?

Connor Selwood?

He is not here.

Ola!

Yes it is you.

(APPLAUSE).

You choose whatever you like to have.

The book, the ... dangerous wheeled thing.

(APPLAUSE).

Mechala ...

You probably go flying, this has been -- Christian ...

Ha ha ... Loek. Where are you?

Where is he? He is hiding!

Again, someone else you don't want it? Okay. Alehandro.

Every penny raised from that is going the go back into the community, so thank you for everyone who put something in there.

(APPLAUSE).

So this is it.

All that is left now is to say some thank you's, to our sponsors, we wouldn't be here without them. We certainly wouldn't be paying the prices that we did. Yes. Thank you.

(APPLAUSE) Everyone of them is also a part of our community, thank you to the speakers and the workshop leaders and Sunday and tomorrow and Friday.

Thank you to Cardiff University for all of their involvement (APPLAUSE) and the individuals.

(APPLAUSE).

Please do mention them it does help. They gave, made numerous contributions members of our committee, volunteers provided facilities, scholarships, tickets and prizes, funded students, we have had speakers, the counselling service was here and they helped bring our visitors from Namibia to the conference as well. They have been fantastic for our event.

Thank you to Peter Finch our conference poet.

Thank you to Cardiff print workshop for providing the art work. Elinor and Clare could you come up please to the front, I would like to introduce you to, yes.

(APPLAUSE).

Many people provided art work. Clare organised everything. Elinor is the artist who produced the Djangocon logo and the Pelican, the wolf and the raccoon, so it turned out that she was the artist we used for all of the conference art work, so thank you so much.

(APPLAUSE).

If you have enjoyed the prints and the art work, take a look at their stand and say hello to the other artists there.

Thanks to Alex and Rob at enthusiasm events for managing the AV all the three days. (APPLAUSE).

And to Daniel and Christopher who have filmed this, we will have this published on-line about when? Tomorrow night? (LAUGHTER).

Sometime soon, so thanks.

(APPLAUSE).

The Action on Hearing Loss speech to text reporters Hilary and Sheryll of course.

Brecon mobile crèche who looked after our two youngest Djangonauts.

Cardiff City Hall and all the staff here, it has been fantastic here, really lovely.

(APPLAUSE).

Thank "you" for coming all this way to Cardiff, I went to Djangocon in 2011 in Amsterdam and I thought, wow manage having this in Cardiff one day, so here we are, thank you for coming. This has been a dream come true for me, thank you for coming and being part of this.

(APPLAUSE).

Thanks to our supporters many of you bought your own tickets and some of you bought a ticket for someone else, you bought a diversity supporter ticket, there are people at the conference because you bought a ticket for yourself and a ticket for someone else too. There are many other people who just added an extra amount as a donation, that also has gone to providing free tickets for people. So thank you, the generosity of this community is something that will keep it going and keep it healthy for a very long time.

I would like all the Cardiff University volunteers, Tom, Ben, don't hide please, Gwilm. Come on.

Gwilm.

Where is ... where is Girish where are you? They said can we be volunteers and they turned up and volunteered. Thank you very much.

(APPLAUSE).

And the Djangocon Europe 2015 committee, please come on the stage too, they are also unpaid volunteers please.

(APPLAUSE).

People asked me what is Django? The easiest answer is the one, our tag line it is a web framework for perfectionists with deadlines. But for me since I first encountered it six or seven years ago, it is an adventure with friends so thank you my friends for being part of this and I hope we will have many more adventures together, so thanks very much.

(APPLAUSE).

Just briefly thank one person who deserves it the most, Daniele, most of you know, I consider him as yours, he is part of your community, in Cardiff, he is the kind of person, spend 30 minutes in a room with him, you wish you were more like him. We have a few gifts to give him. Come on stage please.

(APPLAUSE).

We insist that you open them now please.

Can you read that?

It is a one direction autobiography!

(APPLAUSE).

DANIELE PROCIDA: We have a lot the thank one direction for! we have at least one speaker who was here because of one direction so, I wonder what this small package might be? Because well! who would have guessed. So there is our sound track!

Thank you all so much.

NEW SPEAKER: VINCE: Don't go, everything that you guys have been through over the last couple of days and is thanks to Daniele, he has put in so much effort. Everything is through his vision, we have serious gifts for him, Geraint is going the bring them up. A bottle of wine and some tickets to the opera. Everyone should thank Daniele I know this has been a dream of his for a long time.

(APPLAUSE).

DANIELE PROCIDA: It has truly been my pleasure, it is time to go now, if you are sprinting or go the workshops we will see you bright and early tomorrow at Cardiff University for all the things we are doing there for two more days. Peter Finch is ready to take you on a Cardiff walk, I hope you enjoy that. If you are coming to the Clink it is at half past seven so thanks once more.