Djangocon 2015.

Tuesday 2nd June 2015

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

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

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

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

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

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

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

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

Take advantage of that.

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

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

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

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

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

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

(APPLAUSE).

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

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

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

So are you ready?

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This slide is very Polish!

(LAUGHTER).

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

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

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

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

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

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

She finally reached a definition of model choice field. She decided that the error must be somewhere there and without even thinking she decided okay, I will override the model choice field too, I can do that, I already override the iterator, why not do it with the model choice field?

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

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

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

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

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

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

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

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

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

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

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

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

What happened? Asked Alice.

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

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

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

So the rabbit hole is kind of metaphor of entering, disorientating, confusing land and a moment ago you could follow the Alice steps following the rabbit into this strange world of query set, model choice field, and the strange thing is nothing seems to work as expected and he got more and more confused and more and more frustrated.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

So, how one can fight with rabbit holes.

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

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

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

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

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

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

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

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

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

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

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

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

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

And if you don't work with people who willing to help you or will judge you because you do weird stuff sometimes in new code you probably should change your employer. If you don't have colleagues because many of you are self-employed or freelancers you can ask your friends or ask on internet. Sometimes even explaining things to somebody who is not a programmer is very, very good idea.

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

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

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

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

Thank you. {Applause}.

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

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

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

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

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

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

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

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

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

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

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

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

FROM THE FLOOR: Thank you.

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

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

(APPLAUSE).

OLA: So our next speaker is Benjamin, works at (NAME - INAUDIBLE) works in Copenhagen. He is going to talk about best practices and how to give your pony wings. (APPLAUSE).

BENJAMIN WOHLWEND: So Hi everybody, my name is Benjamin and I have to make a confession. I am a lazy programmer which I mean what I mean with that is I like my code to be lazy or in other words I like my code to do not too much work. That is something I learned after a few dozen Django projects I am not a fast learner so the thing is, making your code faster or making your code not do too much work makes your code faster.

To illustrate this, I would like to take you on a journey on a project that may or may not have happened in 2009 my boss said to build a stack overflow clone, but with a twist, should be for pony owners. I have Django, should be a piece of cake right?

Jump in, create a few models, question model, easy stuff with the foreign key to a user, a title, body. Then an answer model can with the key to the question, again with the foreign key to user, body, because it was 2009 and it was cool back then, added a tag field, yes, then let's move on to views. There we didn't really have class based views back then, I never said that this story was true. (LAUGHTER).

Yes, really simple stuff. To finish it up we added some templates, again really basic stuff. Here we have a template for a question list, we looked through the questions, we displayed the title and the user name, the body, back then we had the mark down filter then we show how many answers the question got and the text of the question.

So I guess we are finished with the stack overflow clone, let's put it live, so everybody is happy, fist pumping and clapping and partying until this happened. Somebody linked to us from this cool site called pony news, where people hang out to yell at each other. So yes, what happened?

Apparently we have a performance problem. Or a more realistically have several performance problems.

So, problem now is how do you even find where your performance issues are? That is the really hard work, everything before it was just child's play.

What I usually do, I run tools that people run that are way smarter than me, different kind of tools, tools to use when you are developing locally and tools to run in production. I am concentrating on a few local tools on this talk. Especially Django debug tool bar and Django deverser.

If you not heard of Django debug tool bar, I use it all the time, gives you insight into the cache access, but gives you a nice overview of all your sequel queries, it is more the visual impression that you can see here, lots of sequel queries, it is an indicator that something is wrong.

If we zoom in, remember we just are displaying a few questions, so why should it take 121 queries, that is insane. Something is wrong here. Let's look at what Django defserver, it is similar to the tool bar, but with the more bare bones interface, outputs everything to the console, really nice having it running on the site. Sit on the side while developing what is going on.

What is really cool, there is 59 queries, we queried the same thing 59 times, something is wrong there. Found ourselves an issue. What are we going to do now?

I usually have to, big three-points I am looking at when I try to fix performance issues, first to reduce database queries and round trips database, then what seems obvious, do less work with the code. If everything else fails start to cache stuff.

Django gives two really cool tools, select related first.

Select related is nice if you have a foreign key and you want to pre-fetch or preload the data in one single query. In our code if you remember we had this question model with the author and we are looping through the questions to display the user name. So, when we don't use select related what happens in the database is, first we get all the questions in one query, and then for every single time we access the user model we do another query to get the user. So if you count it you can see we queried first user three times and the second 12 times, that is where the applicants come from. Did a lot of round trips that are necessary.

Select related there. Select related and the name of the field and what happens then is we do it all in one query, so we get all the objects that we need at once and Django maps it for us and everything works beautifully. Back in Django devserver, we are back to sixty one queries, we had 1,211th, half of them are gone, we don't have duplicates anymore. That is great, let's move on to pre-fetch related.

This is really cool for reverse foreign key relations and many o to many fields, we have examples of those in our code and the tech model which is a many to many field. We loop over the questions in the template and display all the terms for that question. Without pre-fetch related. This happens. Again we get all the questions in the first fetch and then for every single time we loop over the questions again, we go back to the database and then get the text related to that question, which again is not ideal.

So let's add pre-fetch related. Pretty easy, cool API, what happens then is we do two queries first we get all the questions and then Django magically figures out which text needs to do, gets them all in one query and matches them back to the questions we got before.

So, in the end we end up with three sequel queries, 121 first, then back to, we can say we accomplished our mission to reduce the queries.

The next chapter is do less work, which seems less obvious, different ways to achieve that. One way is to move work out of your request response cycle because that is our priority here to reply to the user or to sent cob tent back to the user as quickly as possible, so you can move work that is not necessary for the response out of that code path we are can answer to the user way quicker like API calls to track accounts or stuff like that.

Another one is to not, another way is to not fetch data, Django by default fetches all the columns for a model when you query for it. There is a cool thing, defer, you can tell Django not to fetch data that you don't need. If you have a huge column, which contains the body of the question, you can say defer, and then the data won't come over the wire.

What I like the most is to let other people or other layers do the work. One thing that we can do in our code that we looked at is to calculate stuff with annotate or aggregate, if you remember, we are showing in the question overview or in the question list how many answers the question got. To do that we are fetching all the answers from the database and then counting them which seems kind of not very smart to get all the questions and just count them. So let the database do that count. That is really easy to do with annotate if you look at the code there, we just tell Django to count all the answers and put it on a property called answer count and all we end up there in the end is a number which we can display to the user and we are done.

So, this actually talk about this 20 past noon I think, if you are interested in annotate and aggregate, check the talk out.

Meanwhile let's move on to caching. It is an interesting topic. If you do it properly, you can like scale to infinity basically but, doing it properly is really, really hard. Cache invalidation is a hard part because the danger that you are running in if you do caching is that you end up serving stale data to your users which might be okay in certain cases like in a Blog or in a stack overflow clone, there sit doesn't really matter if the content that you are showing is couple of minutes old but if you are running a web shop organisers perhaps an e-banking website that might be pretty bad if you served account balance from a few minutes ago, so caching is dangerous, after experience, adding caching to your code can make it hard to test and to reason about how your code actually works so, my advice is to use caching only when it is really necessary.

I am not saying that caching is bad not at all. It is one of the more important tools to have at our disposal but only start using it when you think it is really necessary. Let's say it is necessary for our case. I really easy way to get started with caching is to cache page decorator. What you do here is you just wrap your view with the cache page decorator, Django will check if it served that request in a, the last 60 seconds in this example. If it did, it return it from the cache if not, recreate the content and put that in the cache. Really easily cache the content of that view. There is a few problems with that, there is a couple of box open in the Django box checker, but generally it is a nice way to get started with caching.

There is way more options to do with caching. The cache template tech, wrap complete pieces of code of your template in a cache tag and then Django caches the content of the template tech, what is nice, there is ORM caching, really cooed tools for, that one is Johnny cache which is the best named cache out there. Another one caching machine. They give you a quick win. Then there is the big ones like varnish for instance which handles a lot of caching up for you, but it is difficult to set it up correctly.

If this has peeked your interest, look at high performance Django which came out a few months ago, there is a lot of stuff in there about caching and a lot of ways to get your Django side faster, I recommend to read it.

So in retrospect, really watch out for those sequel queries they tend to come back and bite you. Being lazy is good, don't do too much work in your code, try and reduce the work that your code does. If everything else fails bring out the big caching guns and start caching. So, yep. That is all. (APPLAUSE).

OLA SITARSKA: Yeah we've got time for a couple of questions.

NEW SPEAKER: Great talk, thanks. You mention defer. I don't think you mentioned only which is a parallel to defer with the logic going that way. I wondered if you had any thoughts about when you draw that line, when it is worth deferring because obviously there is a service side cost you pay for constructing a slightly more complex object and there is a long term engineering cost weight you've got to carry in terms of if you accidentally start using an attribute that you said you were deferring then you introduce additional queries so do you have any suggestions about when to make that decision, what sort of criteria you use to make that decision to use defer or -

BENJAMIN WOHLWEND: So my brother is a carpenter and he told me always measure before you cut so benchmark, try it out with defer, try it out without defer, see what happens. Yeah, that's I guess most important point. Always benchmark before you try it out.

NEW SPEAKER: Sure thanks.

NEW SPEAKER: So the devserver looks like a very good top level profiling tool. Do you have any favourite one that delves a little more, gives you more data, more finally carrying data?

BENJAMIN WOHLWEND: Well, it depends a bit what you want to do. I mean the Django debug tool bar for local development gives you fine grain data about what's going on in your request, for production tools there is stuff like nurelike or upbeat(?) which this company - which gives you fine great data about your production code so yeah there are so many tools in the market just find to write one for your task.

NEW SPEAKER: Thank you.

DANIELE PROCIDA: I don't think you have to feel shy about mentioning upbeat because it's exactly part of this endeavour so do go and visit the upbeat stand they're one of our sponsors one of the companies making this possible so thank you very much. {Applause}.

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

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

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

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

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

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

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

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

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

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

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

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

The Django rest framework makes it easy to build on top of the orm and get cloud working for resources very, very quickly and then allows you to continue from there to create customer actions.

If you've heard me talk before you know a live celery. I feel it's a vesting class application of synchronous work cue which pretty much any non {inaudible} application needs in a function. The fact that you can have this pretty much out of the box with python and Django puts it leaps above lots of other solutions.

But for all of Django's strengths it obviously has weaknesses. The most I guess talked about weakness is the C 10 K problem. For those who don't know what it is it's handling 10,000 concurrent connections to a single process. And Django just does not do this very well. Python or at least Python 2 wasn't built with massive concurrency in mind and Django was built on a Python 2 world so your alternatives here are to use something like A sink IO, or twisted which gives up a lot of the library support which makes Python strong in the first place or it's monkey patch the world with something like G event which gives you all kinds of edge case problems. You know it's patched event loop.

So with the proliferation of client side tools comes the necessity of proliferating Java script as well because Java script is a de facto language in the browser so when you are developing on client side Java side tools you enter this community that expects Java script everywhere, expects Java script on the server too to take full advantage of the tools so you are hamstrung if you are not willing to run Java script on the server.

Look at problems you could run into trying to run the ocean with Django. Pushing the web clients is a very comment problem for modern web applications and Django doesn't have a good solution to it. You can try to use long pulling but during the time your connection is being held by the server you're consuming a precious Python web worker and this is a really good way to deal less your server really quickly.

However if we look to the wider community we see there are lots of options for solving this. Socket IO is the most famous one for web socket communications. There is also built in web socket support with go, which uses not blocking IO. Native solutions like web sot {inaudible} hold the web socket open. Things like using tornado where they sink OI or twisted like I mentioned before.

So, let's say that you your IPA has a point that issues a celery task but for your clients who continue with its interface it needs to get the results of that task before it can move on.

This is common problem in like search engines. Let's say building a travel search site you want to execute a search but you can't show anything to the user until you have the results of that search back. How do you solve this problem with Django?

What I'd do is I would organisation meant my API response with a list of celery task IDs that were launched as a result of that end point and then create a separate end point to allow me to get the status and result of each of those tasks then set an interval in the client to check you know check over and over again for the results of those tasks and move on once I have the information I need but there is a more elegant way to do this.

If we build on the ability to push the clients, then we can treat a web socket server as a bridge to pub sub and if we are standing up pub sub on the service with something like Reddis our ancillary workers can push direct to it using {inaudible} as channel name then our client can using web socket's subscribe to tasks they receive through the same augmented API end point with task ID then have information streamed down to them as if they're a pub sub subscriber to that task and then all of a sudden you've got this complete push loop going on where your client can in real-time know updated information from asynchronous processes on the server.

As I mentioned before there are a bunch of Java script specific tools which are helpful when building rich clients not least of which is server side rendering of your initial page state. This became possible with reacting has been adopted by ember and some other web frameworks but to do this you pretty much have to have Java script running on the server. In the early days of react Eric was telling me he tried to sub possess out to node in order to render react components to grab the HT mode back and send it down through Django and this is obviously just not an optimal solution. So, I would not recommend doing this, it's very slow.

If you on the other hand are willing to run no JS to serve directly to your client as well as Django API side by side, then you can just use the native reactor render to string and have your node web server actually serving your client application. Remember you are treating the browser as just a service consumer so it doesn't really matter where that application pay load comes from, it just matters that it's responsive, it matters that it is fast and you can work with different services side by side. So in this world you would probably want to use something like cores to be able to have a node on may be a dub dub dub sub domain and your API on a separate sub domain and have node render the initial pay load of your page, sent it down with all of your static assets then treat the Django part as simply an API provider for the client.

So as soon as you have more than one service serving web traffic you have the problem of shared authentication with the client and there are 2 mechanisms you can use for statefulness in a web browser, there is cookies and local storage.

You can refer your off to Django contrib and just using it willy-nilly happily. You can go to an off load in page for your client, get a set cookie back in response, set your session ID on a domain that again will go to both your other public services and to your Django service and then any requests you make to your other services they can pull that session ID out of the cookie header and actually defer over to Django to ask who is this user and get an authentication response then be able to have authenticate user for their service.

If you don't want to use contrib auth because you don't want multiple methods for KPI you can use it directly same sort of mechanism, issue and access talking to your client, your client either stores it in cookie local storage sends it up on API requests and also request to the initial page though if you are using local storage you can't take advantage of authenticate service side rendering because you need {inaudible} pull it out of local storage and send to it node that's something to keep in mind.

Another option is defer authentication to a separate service which is the {inaudible} SLA approach because all of your public services are treated equal and but there are moving parts involved and it's more complicated to get this up and running so you may prefer the lighter approach.

In conclusion, say no to full stack by python I know a few years ago there was an idea we should be pursuing ass a community full stack Python, Python everywhere, this just isn't feasible for the web and it will cause you pain if you try to solve all your problems with Python. Choose the right tool for the job. Not all nails are - I don't know what - mice, no, not all nails are mice.

HTP is your friend. All of your web services know how to speak it how to listen to it, use it to your advantage. Talk your services in http and when you need something more flexible than direct one-to-one communication use something like publisher subscriber model for your letis(?) for your post stress for your cofca(?) whatever.

And think about web browsers as service consumers rather than as special displayers of content or something. Web browsers are sufficiently advanced enough at this point and the tools are sufficiently mature that you can build much more modern web application architecture by treating your client as an API consumer or as a consumer of your public services rather than trying to make it more like a dummy terminal.

So a little pitch, heroku is hiring got employers all over the world, if you are interested go to the careers page. Now I will take questions if there are any. {Applause}.

FROM THE FLOOR: Hi, so, the polyglot stuff is really cool and great but screams often over-engineering to me, how do I go about working out when my website is big enough worth, to be worth bothering to do all of this. Writing the API's, all the services, front end, is a lot more work than is necessary to whack up a single Django only site, do you have advice on how to get, when is it worth starting to moving a service ...

Dave: If you are building a Blog, you don't need the real-time responsive stuff. But if you are building Google docks, then obviously you want something that acts more of an application, to what degree does your project look like a website versus a app. There are shades of grey in in between, the more it looks like a app, if more people expect for a native application.

Be thinking about is, my product also a platform play, like is it going to need an API anyway, if it needs an API then you get a lot of advantage of dog feeding that in your web client. You are using your API all the time, you will feel the pin points and make it better.

RUSSELL KEITH-MAGEE: I would like to get your details on the assertion at the end, Python can't be used for that right now the tools are not there, not anywhere near as mature, do you think that is a fundamental limitation of Python that it shouldn't be like that or never develop the tools or something inherently good about java script, I can't believe I said that word.

DAVID GOULDIN: You are writing java script on the client, if you want to write V eight in Python, I don't want to.

Python can do all of this stuff, Django itself is not particularly well suited to things like you know, web sockets and A syncI o, there are a lot of ..., weren't built with the con currency in mind. A sync io are promising and it is possible we could get to the point that the only benefit of doing the ... is for isomorphism.

FROM THE FLOOR: Making dynamic responsive applications is something that is obvious for us but for example, in my company, each time we are trying to do something new, right front end ... java script, somebody from a department that is responsible for (INAUDIBLE) comes to us, no, no we need to render everything on the pages, needs to be traditionally.

DAVID GOULDIN: I am having trouble hearing you.

FROM THE FLOOR: Closer to the mic? Better now okay, so, my question is, how do you like handle the situation when you have to choose between making responsive web app and like classic website where everything is rendered from the controller view template and stuff like that for this co purposes?

DAVID GOULDIN: Are you saying when you would choose which?

FROM THE FLOOR: Yes, Google robots don't have the capability for now to grow the web size that are rendered by java script, I think.

DAVID GOULDIN: They have (INAUDIBLE) I see the point in that, you are departing from web standards right and so, like you are going to lose some of that goodness that comes along with like really simple just like full responses in the original page, is that what you are talking about yes., I would point to isomorphic java script, get the goodness of the client application, ... fully rendered server page, have things like push state to keep track of your urls's, the user fields like they are using a regular old website even though like java script is powering it on the client. Copy, paste ...

Shipped down fully in html to the client and then the client uses the Django API as a consumer, does that make sense?

FROM THE FLOOR: Thank you.

FROM THE FLOOR: Thanks for the talk, my question is, how would you go about like integration testing in an environment that has lots of services or you just have like, the unit test level things where you test like the service in isolation and you maybe use marks and how do you make sure that the mark for a certain service stays or has the same API as the service, it doesn't change over time, and you are testing something that is not there.

DAVID GOULDIN: This is edging in on the world of distributed computer, where service contracts are extremely important and they should be versioned and they should be enforced. I know a lot who use the Json schema to document the ..., as long as you have strict "contracts between one service and another., you can test a service in isolation, know that the integration will work, you always want to have integration tests, to run those in like a sand boxed environment probably want to use something like docker or ... or something like that, to spin up virtual machines so run your services in is way that has party with ... I know ... launched with what they call dev cloud, the 20 something services they will spin up that number of services in order to test all of the integration points, the more services you have, the more difficult it is, as long as you have strong contracts most of the time you don't need the integration level and you can mock out or stub out calls according to the contract established between the services.

FROM THE FLOOR: Thank you.

DANIELE PROCIDA: Thank you very much David.

(APPLAUSE).

Our next speaker is going the need internet access, we are having a few problems with connectivity today, so, unless you are doing something life or death right now on your phone or lap top, just close it for the next 20 minutes that will be greatly appreciated.

DANIELE PROCIDA: I am specially pleased to introduce Stefan Foulis from DVO and it is I had the privilege of having him for a colleague at DVO, so a great pleasure to work with him and pleased to have him here speaking at Djangocon.

(APPLAUSE).

STEFAN FOULIS: Thank you Daniele, so today I will be talking about local development and local Django development with docker, so I will briefly introduce what docker is and what the benefits are of using docker. I will look at some basic docker concepts and then we are going to create a dockerised Django product. Then use docker compose to define our stack, so run the stack locally, as a bonus, if we have time I will show you how to use nice urls's with a proxy and docker.

Most of this works without internet, I think just to make sure, I think we maybe sacrifice something to the demo gods, for that I have kittens (LAUGHTER).

So let's delete or sacrifice one or. (APPLAUSE).

Hopefully that is enough!

(APPLAUSE).

I still have some kittens just in case, so, let's see if we can do this.

First of all, I am going the just want to show you a scenario, so got a new project just checked out the repository and you want to get this up and running. You are go the docker directory, the docker composer comes up. Start all the containers needed for the application, we are starting cob stainers and the web application, then the familiar run server. So everything is running. I can switch over to my web browser, open this IP and there we go. That is the web app already running, in this case, just language containers running in, that is for debugging purposes and already using the cache, 3 second cache and a longer cache, so we see every 3 seconds it is updated so show you how the cache works.

I want to guide you through the way this. I want to find my presentation again, dam kittens!

There we go.

So what is docker? To make it really simple kind of like a virtual machine. If you understand how to make virtual machines work to understand the analogy. It is efficient, run them in a host. Not the overheads, you can launch new containers within milliseconds, not in seconds and minutes like with a virtual machine. The main selling point is the set of tools around it to build and share your applications with others and we will have a look at that with docker images.

The party between development and production. Because you can really run the same code bit for code, the operating system, every library that is installed, the development and production.

So some of the common things you will hear, combination with docker, here is the image, this is you can think of a template. To go back to the virtual machine analogy, a virtual machine, packages then make a snapshot then use it as a basis for a bunch of other machines you run, those will be containers in the docker world.

Then there is the registry, a useful place where you can share and distribute those images, those reusable images.

Here is some of the operations you can do in docker, at the right hand side the machine, you can docker pull an image which you would get from external registry for example, or build locally with the help of the docker file, once you have it you can run it.

On docker hub, there is ready to use images, like we are using in this example. Django image in docker that is just insanely easy to use. So, if you, images work in layers and that is how it does this with these different images so, there is for example, a debian8 docker image and then the Python image. Then there is the Django image that builds on top and adds it stuff. At the end your app which adds whatever is specific to your application and of course your source code.

A docker file to run a Django app is just this one line. In this case it is using the own built trigger, automatically detect your ... copy your source code into the image, there is not more needed. To see an example of how docker files work, I will make an example of using the Python ... so this is a docker file, start off using the Python image, install some packages, we want the mysql client, couple of things, create a directory, make this the default directory, copy in our requirements, install them. Copy in the rest of the source code, tell docker we are expecting to expose part 8,000., requirements file is simple. Just Django, now if you want to start a project from scrap, use docker build to build this docker file which is created dash T defines the tag, the name the image is going to have. Then use start project which basically we do docker run, IT means run it interactivity, ... -- V mounts your local source code into the container and user is needed for permission reasons for mounting that folder. Then we run the general admin start project and you are all set.

We can run the container now with docker run, the same thing, the only thing is we tell it to map the port 8,000 on our local machine, to the port 8,000 on the local container.

... default command for this imam, but listed it here for completeness.

Docker compose, a command line tool that adds a yml file which describes the whole stack, a combination of multiple docker images and build them together to a whole stack. So I will switch over to demo again and show you such a composed file.

So, here at the top you see it is, we all our main container we call web, we tell it to, if we want to build it, build in the current directory, add this to your project. We do the volumes like we defined before in the direct command as well. Mount the directory for our up loaded media, so we don't lose that when we recreate and destroy the container. Map the port again.

This is where it is interesting, link other containers into this one, take the container and map it in this container with the name postgres., this is straightforward, pulls it from the docker hub and you are good to go already, all we do is define the password. The database here, the host is postgres. ... Makes the host and matches that name to the IP address of the other container using postgres.

We have extracted the most important ones here, of course you will have to update your settings a little bit to follow this convention, so here, you see, we are setting secret key based on the environment and using some reusable tools like Django database and create settings out of it.

I got the docker file is a simple one, just the Django template we have built. So basically we are ready to try this out.

This is the app I started previously, I will kill it. So all containers are stopped again. Actually completely delete those containers. So this is now wiped postgres container, wiped the data, starting afresh.

This time, I am going the do compose up with no recreate and just a web container to show that it will automatically detect I need these other images because I need the links. Start them up, no recreate, tells it not to rebuild the containers if not needed otherwise we will lose the postgres data every time we start.

So we have our up and running again. You see, Django is already complaining oh, we forgot to migrate our migrations, this is pretty straightforward, so here is the command, docker compose run, which will with the web components tell it, well I want to run this in the context of the web component and run the command. I will run, now I will end up in this container and I can run multiple commands so it is a bit faster than running, than less typing than always typing the whole thing, so I can just Python, manage, migrate. Here.

There we go.

While we are at it, we will create a superuser.

Great. So, let's switch back to the browser. See if it is still working? Yes, we are still up. Now we can also log into admin, with our newly created user.

So, let's say we have this extra little application, myapp doesn't have any migrations, you go about this the same way, every time docker compose run, web, your commands or step into the container and just do them one by one, we will make the migrations for this app.

There we go. And we will go ahead and run them as well.

So, another thing that is a bit different than when you normally develop in your machine, when you need to add dependencies, in order for dependencies be picked up by all of your containers you are running, add to requirements and rebuild the whole container, a trick there is you can also just add them. If you have a single requirement and don't want to rebuild everything, you can use caching mechanism to your build. Install that one package you need for experimenting but before committing this to code, you should probably put it in requirements. Py.

Once you have rebuilt the container so the command to rebuild it with docker compose would be build then the name of the container. To be really sure that it actually gets the new image, it's a good idea to delete the container and recreate it. It's also good to do that when you change for example environment enables because it doesn't always pick those up automatically.

Another slightly different thing that you have to take care of if you want to use tools like PPB(?) with docker move up you won't have an interactive console it's just a log output but you can run data in another way that it's because to have an interactive console so run docker compose run instead of up and you tell it to service ports extracting which means also maps your port into a container like it would do when you normally do up and when you start the container like this you'll actually have an interactive console here and can do PDP debugging.

There is another neat way you can change stuff, for now every time we do docker compose run it creates a new isolated container that is not part of the web container already running but there is a command to step into, an existing container. And that is docker exec. So I can do docker exec IT for interactive, I choose a container, let's take our web container. Run bin bash. May be have to sacrifice some kittens. No don't want to kill more kittens - what you do here is step into the processing space of existing running container then if you do PS for example you see the same processes running that you see when running the run server.

Another neat thing is a package called - or docker image called N gin X proxy. We run this on our developer machines on port 80 and port 1 1 3 so does all the proxy for {inaudible} run locally and this is smart enough to watch docker and see any containers that get started up and if you have an container that has environment virtual set like this it will run any request for that virtual host to the correct container.

So what this gives you is that you can actually run your web-site like this which probably won't work now. So you can have real URLs for your web-site. This alleviates problems you might have with cookies when you run stuff on different ports, makes things way simpler in that regard. It also has the functionality of doing SS L determination so for our older stack we have SS determination for older development as well and that allows us to really test the different services how they interact much better than if we have just http.

Okay I think I am running out of time.

Let's go to questions. Thank you. {Applause}.

NEW SPEAKER: How do you keep track packages installing your dock images and -

NEW SPEAKER: Can't hear you?

NEW SPEAKER: How do you try stacker images make sure all your images are to date with all the latest secret batches?

STEFAN FOULIS: Updated images on docker hub and make sure you get those locally?

NEW SPEAKER: Yeah or that images on docker hub web are updated with latest open SSL or mercury or ability -

STEFAN FOULIS: Basically it's to you if there is a dependent image you are using that has a bug you have to rebuild your application based on that. There are tools on docker that make this easier so if you have docker hub build your image as well a service they have you can have build triggers that will automatically rebuild your image if one of your dependent images changes which means you always have up-to-date version in that cases you still have to remember to employ that to your live system but that alleviates it a bit.

NEW SPEAKER: Hi there. Why should I move from vagrant to docker as a sort of development environment?

STEFAN FOULIS: Well you think it's a good idea or?

NEW SPEAKER: No - I never heard of docker much and I just wondered.

STEFAN FOULIS: It's much more light weight specially if you have a lot services that are different, if you have one part it's only a no dap or Django app and have different dependencies what they need on the system. It's easier to spend up 10 docker systems with different operating systems than 10 vagrant systems with different operating systems.

NEW SPEAKER: Thank you.

NEW SPEAKER: Thank you very much. Do you think there is any benefit of using Ansible with docker?

STEFAN FOULIS: You mean use Ansible to build docker images or -

NEW SPEAKER: Yes -

STEFAN FOULIS: I don't think so because the nice thing about the docker files is basically you don't have to maintain your image over time because whenever there is any change you just rebuild the whole thing and you redeploy a whole new artefact so what Ansible is strong on I think is keeping servers up-to-date with what you've defined to be your stack over time which isn't really needed with docker any more so the docker file I think is perfectly enough to build those docker images.

DANIELE PROCIDA: 2 very quick ones.

NEW SPEAKER: First question. Do you use docker in production. If yes how do you manage to keep user data and put database on this containers if you need to rebuild them? If not why not?

STEFAN FOULIS: OK so we do use docker in production. We make sure toe that the containers are state less so any up loaded files for example on the sites me host go into a shared storage back hand. The postgres database is running in docker but not running in those containers, it's running separately as a service and we're using that and with postgres for example if you want to get your data outside of the container you mount the directory and have the data part outside the container so you reconstruct them.

NEW SPEAKER: If I may, how do you manage different versions if you have to all {inaudible} container NEW SPEAKER: SF of the what?

NEW SPEAKER: For example postgres data 3, and 4, have to change data of underlying files -

STEFAN FOULIS: It's the same process you do traditionally with a server so make sure you migrate the data when you upgrade.

NEW SPEAKER: OK thanks for your talk. Say you have your own project and have gone down a rabbit hole writing your 5,000 line bash script to build everything up and only just heard of docker, do you have any advice for converting your existing projects over to docker?

STEFAN FOULIS: Really depends if you're having problems with your current set up. If you're not having any specific problems for them I don't see a good reason there but of course it's useful for new developers starting on a project because the only dependency they need to install is docker so if you have a lot of new developers working on your project probably until they have that set up it will take a long time if you don't have a good reproducible set up.

DANIELE PROCIDA: Thank you very much Stephan. I won't keep you but I've got an important announcement before we go into our break. Let me just quickly do this. OK so I'm about to put through the order for the sprints catering on Thursday and Friday. According to this, we have got 248 people coming to lunch on Thursday. 199 on Friday. But only 137 are coming to the sprints party on the evening of Thursday in-between. So, this is why we need you to give us accurate information because we don't want to have a party expecting 137 people then have 248 turn up because some of you will go hungry and go home disappointed so what you need to do is go to your teeto ticket. You have one of these on every single email message that came to you from teeto practically and just answer the questions. Hit change to do that and say yes I will be coming to this meal or that meal. This will be on your individual or corporate or sponsor ticket the one for the whole event the first ticket you bought. If you can do that straight away we'd be very grateful.

I'll let you get your coffee now. Be back promptly at 5 past, an extra 5 minutes because we have a special announcement for you of something that's been kept secret until now.

(Break)

NEW SPEAKER: Hi, I am highjacking the schedule for a brief unscheduled lightning talk. Thank you to Daniel for organising the crèche which is allowing me to attend the conference, unfortunately I can't make the lightning talks so I have slotted in here. So it's back for those who don't recognise my branding what it is Django under the hood. It's different conference to Djangocon which we ran for the first time last year. If you were paying close attention to Baptistes keynote you'll be familiar with it. It's conference aimed at high level Django developers where we get long deep level talks on various areas of Django and other parts of the system. It was a huge success last year. We had 100 people come and it sold out quickly, talks went really well, great feedback and I'm pleased to announce we're doing it again this year and we're going to make it bigger, so with a bit of luck you won't have to be so quick on your key board to get a ticket, we're going to make it longer so we're going to have 9 amazing talks every day aunt half, start at lunch time on Thursday, 3 talks on the Thursday then 5 talks on the Saturday and sprints on the Saturday, one of the great things about Django under hood are the prints and we're going to make sure we're there properly this time is we're going to have at least 18 members of the core team who've signed up to say they're going to come already so there is really good ratio between those of you trying to contribute to Django and those who allegedly know what we're on about. The other things it's going to have is lots of sponsors. At the moment however we have none so I'm looking at you. Please come and talk us if you'd like to be involved with sponsoring the event, there are tickets included for sponsors, please talk to myself, Baptiste or anybody else on the team. I'm also excited to announce a lot of health duel today so hopefully get you excited about what's coming up we'll start with everyone's famous Australian. Russell Keith Magee is going to give a keynote cryptically entitled once upon a Django. I don't know what it means either and Jacob Kaplan Moss explaining about how Django gets from a wsgi server to your http request and how it goes back the other way. We'll have Lacopo Spaletti a core member of the team talking about Django works for it. Florien a Django core dev and is one of the people who receives your email when you email at security at Django.com to explain what measures we have in Django and how they work. We're going to have Josh Smeaton who is responsible for the enormous expressions patch that landed at Django under the hood last year. For those who haven't used 1.8 yet {inaudible} is going to talk about this today and later today. Following on from her talk yesterday we'll have Ola Sitarska talking about the admin talking about how the admin does things and we'll have James talking about files in Django and how it deals with static files, media files and technical constraints there. There will be a couple of more talks announced to I'm finalising what's going on. We'll be in Amsterdam from 5 to 7 November this year tickets approximately 150 euros. They are likely to sell out fast follow us on fitter find the web-site and look forward to seeing you there thank you very much. {Applause}.

VINCE: Our next speaker is Xavier who started using Django 2 years ago {inaudible} opportunity to get involved with Django. Talking about web accessibility. {Applause}.

XAVIER DUTREILH: So, hi everybody. My name is Xavier Dutreilh; I am going to talk about web accessibility during the next 30 minutes. I would like to say 2 things before starting my talk. The first one is I really appreciate different action on hearing are doing for the conference because otherwise I would not be able to understand anything of almost all talks.

The second thing I would like to ask you is much more a favour than anything else. So at the end of the talk if you want to ask some questions, please, try to be clear and simple so I can read them here. I am very sure that some of you have some questions to ask me. Some of you have been very honest yesterday about issues in life. And I think some of you may have some questions about accessibility.

Accessibility or web accessibility, we can ask both questions.

So, first of all I would like to clarify why web accessibility is not an option.

We are here at the conference on Django. It's all related about the web. So, is anyone in this room ever considered life without the web? And all the smaller opportunities provided to you?

Before this talk I never really thought about this too so it's not surprising you may not have think about it.

But still even if you've never thought about what the web did for you and what happened in your life, it did, you may have obtained work, studied, you may have learned about it, maybe make friends, from what I see the Django community is very friendly so people make friends thanks to the web.

May be today you do not see any issue with the web, you use the web any day and it's fine and may be for you it's perfect but this issue exists for people and the issues are very real. So I make this talk to explain to you what are these problems and what you can do to solve them.

One of the biggest reason why some people have real problems using web-site is because we do not design people for people in general - we design web-sites for non-disabled people. So maybe you are shocked by such statement but it is what it is. We do not include everyone when we design and build web-site and exclude a large part of our community and the world in general.

In particular, we exclude people with disabilities. So, may be the world of disability is for you very fuzzy but with this talk I think you will get a clear idea of what it is.

So, let me tell you a few things about web accessibility.

So, it starts with an inclusive practice of removing barriers for people with disabilities. In fact, we know that many web-site we'll be on every day may be difficult to use if we have some kind of impairment. This issue, this kind of issue keep all people with disability apart, on the side and they cannot use it. And such practice is very important because if you have some or no disability you get access to information on the web-sites and functionality. It is not linked to the kind of disability you may have.

So, disability may be hard to understand if you never read something about it document yourself about or impairment or disability.

So disability is the consequence of an impairment. And you may have one since your birth or you can get some disability later.

So if you get one from birth you can do anything but sometimes when it is acquired later, it, you can acquire the disability because of an accident, of genetic condition that start later in your live and affects your abilities.

Also can be progressive so it means that not all impairments have the same effect on you when you are five years old or ten or 20. Sometimes in only a few weeks or a few months from being fully able to being disabled.

Often we split all this impairment in to four main categories we can see other categories but it is much more important to, to focus on this one to start.

So, visual impairment is what you see with your eyes, you can be blind, so have issues with light and perceiving, understanding what you see. Blindness is not as we may think when we don't know what it is, blindness is not when you do not see at all. It is when you cannot see in fact when you go to a doctor and you test your eyes and what you can see, if your view is rated below a certain threshold you are considered blind. Even if you can still see something.

Low vision, or also called poor vision is something that a lot of you have in some way, if you wear glasses. So a lot of this people in this room, never consider yourself being someone with disabilities? Now you know you are.

Colour blindness is when you have some trouble making differences between colours, it is not always about making a difference between red and green colours, it is also an issue with blue colours sometimes, completely different. It is not only about making difference between the red and green colour. Sometimes a few people have issues with colour and they have issues with red and green colours and blue colours.

Impairment maybe also hearing and most of the time it falls under two categories; deafness and hard of hearing which is the category I am following.

When it comes to motor, I think most of you imagine someone in a wheelchair and that cannot move or walk by yourself, it is much more general, people with genetic condition that makes them having a lot of trouble to move or to coordinate the movement following this category, so that is why we call a limited dexterity. It is also a loss of limbs when you have an accident and you lose your arm or two and you cannot use a computer as you used to before.

Cognitive. It is not visible at first but some of us have issues about learning things. For instance, remembering the things on websites. This is two possibilities, this is category is very large and they cannot learn everything.

From there what can we do to make accessibility?

The first thing is simple. Every time you put an image on the website try to describe it because for instance people who rely on screen readers to read a page, won't get any information. The screen readers, scans the image up and then nobody might be able to understand what it is on it.

So, this example is very bad. So, as you can see the name, it should be in front of me. So, if we try to add a description or may use something like image of Xavier, which is also bad, better to produce this example, we add in description but it is bad because it is a lot of, a lot of redundant information, in fact, one way to describe as, there is an image, there is an image of Xavier, if you never use the screen reader on the website, you may not know, but every time you add a lot of text it becomes longer for people to read the page.

Before this talk I tried to do a screen, voice over which are still on macro six ...

To give you an idea of how long, if you put too much text on the website, it can make difficult for people to follow you and to browse your site.

So this one example is short. It is clean but tells what it should be.

What I do on the photo is important. You have to describe it. Because otherwise it is, there is no possibility that these people, the people who use this screen reader can understand what is on it. So only if it is important you have to specify the action on the photo.

The second, to use html but in a good way, so you have to stick with standards and structure your information. So some of you write documentation on a daily basis, so they are used to a structure of writing but it is not always the case with the web. To give you an example, this is an example from a oversimplified on the website. You would use structure all your page as such. So if you have header in the page and some part of the navigation you would use appropriate types like header and not choose what a lot of people use in their developments or in fact have no semantic at all. So every time a screen reader it does not know if it is a header, navigation or the main content of the page. It makes the things a lot harder.

If you write at all, you may try to use some kind of example. You have a data which h1 a subtitle for h2 ... screen readers pay a lot overflow clone attention to this type of thing, every time on the list for instance they say, nine elements and element one and it is, element two reads the text and so on.

Advice and guideline, build proper tables, so often when we build tables, we put data without presentation to the name of the elements. As you can see here, for screen readers it is very important because it helps when it reads the table to make the distinction between the header and the column in the table.

The fourth advice is about making proper validation from validation and error recovery.

Most of you use Django and most likely only Django, so this issue is not less, not sorry presented in our web apps, much more prevalent when we do a lot of java script in the front end, as you can see here, what is important is that we have a field name input and a label which is, which explains what it is to field before that. The label is attached to the field by filling the attribute, the form attribute, so when you say name here it refers to the input with the right name.

At this level it helps a lot of assistive technologies to make a matching, allowing people to fill forms and it is also good practice to use all attributes you may have in html file because if the field is required and for some reason the person is not able to fill out the web browser will help the user to fill in properly and then instead if we do not make, if we do not fill the, we do not add the attribute, the user may submit the form and then get on without understanding what is, what.

Also this is very important, not also for accessibility but also for search engine optimisation, every time you put a link on any page, try to be sure it is clear. If you have these kind of thing where you name, you name this link as clear, click here, it is very problematic because a lot of people may not understand what it does. There are a lot of assistive technology which scan the page, extract all links and leads them to the user. If you have a lot of click here link in your page, once they are presented to the user, which is any, no links, no links which are named, click here will be understandable by the user.

So if you are, the link which moves the user to the article page, just name it article. But even better, something like see all articles. So the user can understands where you want to move him.

This is guideline is offering ... I think it is just confirms what it means, you should have to every time you put some kind of audio or visual media on the website, you should put transcript. Transcript is what is here on the screen, tells you what the speaker says and eventually some information you may have missed, if you have a hearing impairment like people when they laugh and when applause, caption is not about providing subtitles to the speaker and the subtitles are often displayed at the moment the speaker says the words.

So, I don't think you need a clear example since we have one from the very beginning of this conference, but if you want the slide of the talk or the video on-line, you may want to watch this talk from, this is one of the keynote and suddenly not so much talk from Py Con that contain subtitles, I hope following, the videos will contain everything.

After video and audio, you should ensure like I did for my slide to, if you are unsure, make sure that all of your binary files should be accessible, to do that, most of you probably use some kind of some kind of software like Office or ... to build binary files you just need to use the same guidelines, just need to structure everything, for titles and put things where it would be, it would make things a lot easier, but if you can use just the web for explaining yourself, use the web, because screen reader have made a lot of around the time with web pages not always the case that where all the binary files.

This one, again, it is very important for people who have a lot of issues, a lot of motor issues. Sometimes we have very complex pages with a lot of stuff and we want to help people to move between part of the page really easily. One example is from one link most of you never saw on any website but even though the, you have this like kind of link. This is, it is this one. In fact, at the very beginning of your document you have most of the time a non-visible link which allow you to skip to the main content of the page. So you do not have to scroll to the page to go to where you need to go to get access.

You can do this kind of quick link for everything if your page, for instance, you want to switch directly to each part.

This one is about not relying on colour to convey information. Most of the time since I started to work with back end developers they would use things like label to convey information but as with the classes imply, it is labels which should be boxed with a colour and not extend it. So if you have issues to see the colour, you may not know and understand what it means. A lot of e-mail clients a few years ago were doing that. So labelling all the e-mails, if you cannot see the colour, you don't know what categories.

Simple information, put some label, so if you have trouble to see colours you may, you just may put the text in place.

This one is probably hardest guidelines to apply. Because even if you master one language, if all of you, I assume all of you, as native language in which you can express yourself probably easily, but writing for the web is often very complicated. Sometimes to explain things we use a lot of complex things and we often, sometimes people have to understand what we call nonliteral communication. You have to be explicit about what you say and you will not have to expect people to have all the information we have to give you an easy example to understand this.

If you learn a new language and you talk to someone from the country that choose in language you may not understand the expression, you may not have the same culture in order to understand this.

Sometimes it is also good things if you cannot find someone around you or, I don't know, if you cannot find someone who has some with this issue, you maybe work with someone who is mastering the language, ask them for help.

One advice when you have to do this, you can also try to in fact a lot of this work, I don't know a lot of this word before making this talk, so you have to be explicit, clear, so if you think some of the talk you have heard in the past few days we are expressing and clear about what it is I want to tell you, it means a lot of people have spent a lot of time on it to make it clear. It should be clear and positive, it means that, write explicit clear and positive, there is two way to have sentences, for instance, if I write a positive sentences I would say, I eat at, I ate at the city museum yesterday, it was really good.

So instead of if I will not choose a positive form I would say, yesterday, the city museum has offered me to eat free.

So this one connective issues, much more about visual issues, the first thing is to use clear fonts. Some people for instance live with a dyslexia and much more the font we use on presentation website are not readable by some.

So, if you try to make your web-site accessible, you may want to of course build your pages with one set of fonts but also provide fall backs because there are fonts that have been made a long time for people with some issues with reading, a font we use most of the time.

Also use relative units. If someone may have some issues accessing web-site need to zoom in most of the time if you use pixels to size everything when you zoom in it become a mess and when using relative units make sure when you zoom in on the page everything stay at the similar position.

And strong contrast, well you see the slide, you have to be sure the colour of the text is very different from the colour of the background and it's not magical and a lot of web-site have been made on this and if you need a tool to help you choose colours I will provide you with one at the end of the talk.

And this one is something that has been removed from HTML and it should of course - it's not about toning anything, it's much more about you don't want your friends if you develop a web-site - you don't want your friends to make some sort of epilepsy quizzes during the - but it happens. Some of you are smiling but, yes, for instance we often say that sweep flushes in one second may start epileptic crisis so you don't want your user to have some kind of episode on viewing your web-site and also avoid to move things on the page here because if people who have some issues with co-ordination when things they move will have hard time on the web-site and will not spend a lot of time reading things. But OK the thing was here, now it's here, and will spend much more time in other things.

This one is no simple thing to fix this issue. Java script is accessible to almost all web browser, most work just above screen browser so if you think Java script does not work with a web browser or screen reader especially with a screen reader you are wrong. 10 years ago it was not the case but today it is. I think 98 per cent in studies it has been shown 98 per cent of screen adjust users have Java script so you can make Java script application in top of your Django application.

The only thing you'll need to be careful is that not overlook people using keyboard over mouse. So if you have motor disabilities or if you cannot use your arms or do not have hands at all, you won't use a mouse and most of the time people when they built Java script encryption over {inaudible} for instance they will pint all {inaudible} over mouse and they have no click so may be your app won't work at all. You have to be careful about that much of the time the key board part is still supported so you can stick with key board.

And the last one, guidelines. It's very similar to the guidelines about tutoring content. If you use HTML 5 do not put HTML 4 element into HTML 5 because you'll have deprecated tax and it will be a complete mess. Avoid in line styles, put your access into complete files. Do not miss some attributes for example images, the other attribute is not an issue - {inaudible} it makes your pages very inaccessible.

So, from there, if you at least implement those guidelines on the web-site most of the people will be able to board the web-site you will just start to exclude a large part of our community and people in general. And after that I would like you to spend sometime to first document yourself about impairment and disability. If you're not very - I think for a lot of you it's not very clear what it means in reality so you have to document yourself. I can't talk about it during presentation but you have to document yourself. This subject is very, very broad so you have to spend some time.

You also have to talk with people who have some kind of disabilities. You cannot just talk to one person. You have to talk do lot to understand what it means. Because not only one way to live with disabilities and if you want to improve your web-site you have to talk them and listen to them. It's very frequent in our community to hear from people who have no disabilities and asking people with disability to making more effort include themselves.

They do they already do, so may be at the moment, maybe you should be.

Once you get a lot more fine and comfortable with the disabilities you should also read some guidelines like WC AG, but if you understood some of the things I said you are done with WCAG, the other ones the 2 other ones ATAG and UUAG are about software that allows you to write a page. And there we are come up last once you must ARIA and it's very useful if you build {inaudible} - {inaudible} WCAG is sufficient.

Also I think a lot of you haven't tried, haven't yet any kind of accessibility tools and most of you have a smart phone or computer at your disposal day or night so if you have some recent computer with windows or I don't know 1, 2, any other Linux application activate things like the screen reader the colour blind tools and everything else, it's very important that you try them, may be at some point it will disturb you because it will change how you look at web-site and how you use them to interact with them.

But you should have to try them because if you never try one you may not fully understand what it means to design accessibilities on your web-site.

To get things here a lot of people have started organisation like web aim and they build a web-site which can call your web-site and make you some things that tell you which things work and what things do not work so we can tell you if you use tone contrast and if you have issues with contrast it can tell you why and pose some solution. It applies to the same thing for the links and in fact most of the guidelines I gave earlier is covered by this application. So, just try it. It's wonderful.

And you can try it on the Django web-site.

Also if you don't like using web-site and prefer some kind of extension for your web browser you can use this accessibility tools. Google provide a lot of tools including this one to help you in this.

I don't know if that is something similar to fire fox but my view you should try this one with chrome.

And finally once you master everything or at least a bit of it you should provide your skills to any accessibility organisation that you may have met before or you may meet today because most of the time they like the technical skills need to make great application and you can provide them with these kind of skills.

So in conclusion, web accessibility is not an option. We may think so but it's not. And none of us are optional. So we should get back to work and fix this.

{Applause}.

VINCE: Thank you Xavier. If we have any questions could you make your way to the microphone please.

RUSSELL KEITH-MAGEE: Thank you Xavier. Just something you said at the end there. Did you say that the Django web-site does not have good accessibility at the moment?

XAVIER DUTREILH: In general if you use screen reader for instance not that bad, some areas for improvement and if you want we can talk about that after that but for screen reader it is really good. It is longer to pass and to read so maybe there is some areas for improvement there because some pages were really long, but I think, yes, on the colours, yes, the colours are really - you may not see the issue between the text in white and the green for instance but - so, maybe you do not need to make, to rebuild all the web-site in fact, you can keep your current CSS on the web-site and provide a link which - off CSS which applies some kind of stronger contrast. This is something we can do.

RUSSELL KEITH-MAGEE: Absolutely. If you are here for the sprints we'd love as much assistance as possible to improve any accessibility problem.

XAVIER DUTREILH: We should do it.

NEW SPEAKER: Fantastic talk so thank you very much. I just had a quick add on actually as a person with a visual impairment myself, I don't use any specific tools, but what I find sometimes is the most useful thing is for developers and designers to get out of my way. The best example I have of that is using meta tags to lock the zoom on the view port because that means that when you are trying to pinch and zoom on a web page in order to read text, that optional setting that designers like to set means that you can't zoom in any more. So I think there is a lot that you've said which is fantastic about using tools that are available, but also some of it is about not setting options which get in peoples way so I just had that to add but thank you very much.

XAVIER DUTREILH: Thank you.

NEW SPEAKER: Thanks for the very informative talk I was wondering could you recommend a command line that we could run like unit tests that would validate accessibility. Could you recommend one?

XAVIER DUTREILH: I do not know any kind of tool. I think may be there would be something to build - in fact maybe we can have some library built up on top of, - for instance if we want to make some Django forms more accessible and may be build something above the Django forms to apply accessibility more easily but at this moment, no, I do not know any kind of tools you can run from the common line -

NEW SPEAKER: May be a good sprint project? OK thanks.

NEW SPEAKER: Thank you for your talk. A quick question, from your experience do you think building 2 versions of the web-site one with accessibility and one for normal viewing is better than having the same like version of the web-site respond depending on the accessibility of the user accessing the web-site?

XAVIER DUTREILH: From my own experience and a lot of organisations - accessibility organisation - I think you should deal with the same version and should not put too much like just stick to what I said to Russell before I think one version in fact, yes, and not building 2 version because you want to provide the same kind of information, sometimes you may want to mark some information on the page as optional so it may be skipped by some for instance screen reader but no it is easier and from a dev viewpoint it is important not to have 2 version to maintain. But this one is acceptable because there is no value added to provide 2 version, 2 different version.

NEW SPEAKER: Thank you.

NEW SPEAKER: Thank you. I have inherited a design have you project and I have 2 questions. Where does the responsibility of the designer come in in designing and picking up proper colour scheme? How would you work in accessibility features in a legacy project or in a legacy code base?

XAVIER DUTREILH: So the designer is really responsible. In fact most of the time most designer I worked with in the past are not trained in accessibility so they use funny colours because they like contrasting of colours. There are a lot of tools like wave I suggested before that can help you to, well, you choose one colour or a set of colour that you like and you want to use for design and then after that you can just check that for instance if it's about contrast that the whole contrast are still varied. After that if your question about colours is about dealing with colour blindness, I think you should make your colours optional. In fact you can still use red or blue or whatever colour you want but it should not be important to use your web-site so I don't know what kind of computer you use or operating system you have on it but most operating systems allow you to switch to a grain mode to see how your design your web-site will look like: about the second question which -

NEW SPEAKER: Legacy code basis, working in it -

XAVIER DUTREILH: Oh yes so on new project it's very easy because you can just - at the moment you design your web-site, includes all this stuff and all guidelines. On existing code base it's very hard. I suggest you may be to start using one or 2 guidelines for instance describing non text content may be in {inaudible} for instance using retinue needs things like that may be hard because your designers not been designed for that. I would suggest to take one guideline at a time and apply to your web-site. It is really hard because most websites when they have not been designed for accessibility are hard to integrate.

FROM THE FLOOR: Thank you.

FROM THE FLOOR: Question on colour blindness, could you recommend specific tools for validating our presentation slides web based or pdf's, so validate them on colour blindness, on all the types that exist?

XAVIER DUTREILH: I mentioned a few slides ago. This is this slide, mention WAVE and mention colour contrast checker, wrote a few years ago and you can, you can use it in fact to validate your presentation. Before this presentations on some of my slide I was used to use some different colours like on ... because of this tools I make some adjustment to ensure that if some people in this room had some issues with colour, they wouldn't be able to read. So.

FROM THE FLOOR: I have just tried ...

XAVIER DUTREILH: If you have sometimes ...

FROM THE FLOOR: I think WAVE is really good if you have web based slides, what about pdf's and images?

XAVIER DUTREILH: If it is just about colour, if it is just about colour, you can use WAVE and because you can provide a colour, in fact there is a colours as specified in the tool I use for building my slide can be taken and injected into. Because you have fields to type, colour code you use and assert you can still validate. It is still a lot manual, you can have to take the colours from the tools and import to the tools but yes. It is yes, about colours it is very complicated.

FROM THE FLOOR: Thank you.

(APPLAUSE).

NEW SPEAKER: Just a brief announcement we are going the take a group photo before lunch in the lecture theatre here, if you want to be on the photo, please stay here for a minute or two, if you don't want to be on the photo, you can go straight to lunch.

Thank you.

VINCE: Next speaker is Theofanis talking about aggregations.

THEOFANIS DESPOUDIS: Look it is the guy with the windows!

So, thanks for having me today. My name is Theo, I am a Greek man, I am living in Cyprus for now. Today we are going to talk about like one thing that, for you, for the developers is a set of tools for doing aggregation, and annotations on your data. So, a little bit of thing about me, I used to be the PHP developer, a pretty horrible pages developer I think a lot of people so you started.

I recently, I mean about six months ago starting programming with Python. I have this you know, sound that the snakes do ssss and I work at social airways which is a platform for managing your flights, you can set flights and you can have friends on the flight and you can also do a lot of things with your account and stuff.

So, we can, it is going to be a small talk, not a big one, we are going the talk about aggregations. Then we are going to continue with annotations and then they complement the aggregations notion and then we are going the maybe say a tips and tricks about this and then we are going the answer some questions.

So what are they? What are the aggregations? Basically, database functions, they are functions of the database. Django has a nice layer on top of it. You only have to deal with the effect you know. You just call the method aggregate or annotate. If your parameters are okay, it will return the result for you. So it is useful for producing summaries and collections of objects. You will have a look at collections of objects. It will produce something you say that you need to produce. So, we are going to see some examples later.

It is similar to the reduce function, you have a collection and then it applies the function and then it returns the results for you.

As a single volume. These are some of the functions, that is, those are all supported in Django. They are probably more in the database level but that is, the database limitations specific, some database support, some database don't support. But the average marks means, in this, on the right hand side is the return value. Return the float or the same field or any, or you know.

So we are going to just for this purpose of this demonstration, I have devised like a small database which is we have like this company so this company has like a mini games, on-line games. It has this you know, database schema, then we have gamers, the gamer may like have a gender or a name or whatever he has. We have a game, let's say the game is like on-line risk or on-line strategy game or whatever which is like strategy or an action, whatever we need to categorise it.

We have a game server, where the gamers register themselves. Might be the chance that the gamer registers in multiple servers that is why we have the field here.

The game station, records the games played. We have the game, when it is started, ended, linker to the server and then this mainly we save the result of the game. Then we make sure that the session and the users is unique, so we have this session, this user has won or lost and these, he has accumulated like some points. They might be a chance that we have like a game with like ten users, ten gamers? Maybe some of them have lost, some of them have won and they will just record the result.

Just for a simple minimal database for we need to work on.

Then say let's say okay, how do you use this? So let's say we need to count how many unique this is probably, how does it work? We generate the query that is we apply the function and then we return the result. It is 55 mate.

That is probably the yes. So let's say we need to filter the results so we can apply filtering on the query sets and any kind of filtering, all the tricks and the things you know. You can apply the filters about the user or find the user one or whatever. Then we need to find the number of wins. So, we remove the you know, the losses and we need to count only the wins.

This is the actually, query in the database. So you want to do it by hand you probably do something like this. But the nice thing is, that Django for this functionality for you, you don't have to do this, the other parts. So left to use the API, the interface and then it will, it will help you build the query and look at the results. That is a bonus for Django.

Also you can do like, you also have a standard deviation and variance, they have a lot of functions so you can while you are there, just compute the average of the scores and the session, then you can also add multiple you know, multiple results so you can have the average standard variation, with the sample and the variance.

All comes one the single results for its function you asked for. So, let's continue with annotations. So annotations helps you also, it is another way to have a summary. This is a method that maps the differences, the aggregations aggregate, but these maps a summary value for each object. So, any query set equates it.

So let's say, want to find the sum of all scores by each user, so we need to go for each user and find all the collective summary of all of these scores. So how do we do this? We just annotate the sum and nothing would do here, we just the ... we find the score. So now, every scores, every gamer this score is a gamer object of course will have in addition a field game session result, because I didn't name it, it will give it to this name and then we will have this score, maybe the second will have another score. This really great tool to just helps you know, collect all of those results in your, in the same place.

So let's say, min, max, sum. Find the score of the session with the most scores accumulated. So what we can do here. We can annotate of course over the game session objects now. We calculate all the sum of each game session object across the relationship then we can aggregate and we can continue with aggregation and we can use the sum of course and apply the Max function on it. While we are there, you can also add all this game, access the free things because when we are over there, we can ask another thing.

The result is of course, the game with the maximum sum of scores, that is, it show it is result and the oldest game is like a daytime object.

Then what we can do is also slice and order our you know, annotations, so let's say you have, you want to ask what is the top five players by awarded points? So annotate the gamer object. You want to find his score, the sum of his score of all his games. We want to filter some because they will appear in our as a result. Then we need to apply ordering. Descending, so the first one will be the highest score, the second the next highest score etc. We also slice the first five.

Then we can see that you know, the first gamer is like this guy and his points are like in these extra field. Because we specified on the annotate you know, function that method.

So what else we can do? We can group that is a really cool thing because grouping is a really nice thing to do. You can use the values method of the related manager and if you do that it will just transform itself and it will group according to the fields specified. What I mean by that, see in the example the results can be used later on for annotated or aggregated queries and then we will be merged into the common groups. What I mean by that? So let's say we have like three servers. So we have, we can get the values and then we can annotate the count of the number of gamers. So the servers in the US, have users registered, the Asia and the Europe. If you want to group some of those results you could, you could make the reads the same for two values, that is probably the best way to do it but again for the demonstration, you could group the values for the Asia and the US and now if you go to the same, you can call the same method again it will merge the results so we have like six users from US and four from Asia, total ten from Asia, then the other one. So it is really, if you are into the sequel, you know what grouping means, groups the values into the common group parameter, so in practice you should always check the generator query, you can do this with, like from the Django tool bar or you can use the you know, the go I, you can easily see what is being generated over the request or you can simply print the query set query class, which holds the generated query.

So this links to another presentation, if you can do it in this sequel, try to do it in Python. If you have the option to do it in sequel, just do it in sequel. Apply the function where the data is you can do it as Python but, I mean, it doesn't look good sometimes.

It can be aggregated.

So what I, this also linkers to another yesterday you know, and to another presentation, how do we use a proxy? Maybe you can use it like this? Maybe collect all your queries related to this what you want to do, what you want to achieve, it may be applied different ordering, maybe you want to display another page with another statistics, so you can put the proxy class and apply different ordering. It maybe you can also create your own manager and then you can do all sorts of things as long as you stay with the boundaries of the proxy you know, proxy class.

So this is the last did for me, try to keep it simple. You can get easily into you need to think of what is your problem basically because you can easily get, you know, you cannot easily get away with a lot of things. You need to find what the problem is, what you are trying to achieve, how Django can help you solve that problem. For us, for developers really easy, because we have a vast library of tools to use and so we have to think about what we are doing every time. So, so all that the results, you need to think about if the result is actually worth it. I mean, can we count it? Cannot it be counted, I mean, what is the point? You have to think about what you are doing so any questions?

(APPLAUSE).

VINCE: First questions, come to the mic.

FROM THE FLOOR: Can you hear me? Thanks for the talk, so, you mentioned that we shouldn't do something in Python, if it can be done in SQL, I agree with that., we have a situation where we need to fetch the objects anyway? We need to list them somewhere. So is it better to do another SQL query, where we have to aggregate, or to use the same queries we fetch to do the?

THEOFANIS DESPOUDIS: If you have the results in hand you can use it. It is better if you reuse those capabilities so if you have like a function with, you have this aggregated function okay, if you use it one way, which will only one way, use it only once that is okay. But if you want to use it many times maybe later on in your code, you might be putting it in the function, maybe in the aggregated query, to reuse it, so you don't have to deal with the situation of how do I filter? How do I aggregate the result again? Om have to call the mb, only have to call the manager and it will take care of the results for you.

FROM THE FLOOR: Okay just another question, that is probably a question that is all over the internet, I didn't find the answer.

Can you filter the objects that you are annotating on?

THEOFANIS DESPOUDIS: I think so, you can, because it is a query set it is, you can filter the query set.

FROM THE FLOOR: I think it is an F object, it is not a query search. I mean if you want to annotate for example, only.

THEOFANIS DESPOUDIS: Because if you see the SQL generated it is a new field that gets applied so it is just like alliance, so you get this field back, so if it is like an F field you can, I think you can filter it.

FROM THE FLOOR: For example, your example if I want to see how many won games for each user was, not all the games, only the ones that you won?

THEOFANIS DESPOUDIS: If you want to find all the won games for each user, you have to count them first. Then put them in an annotated you know, field. So you have to annotate the count and find the, for each user, what is his number of wins. Or maybe if you think about what the problem S you need to find the count of the wins for each user.

FROM THE FLOOR: Okay thank you.

NEW SPEAKER: Hello thank you for your talk. I am wondering if we can use annotation with some simple mathematical operations like if you want to subscribe to 2 fields?

THEOFANIS DESPOUDIS: Well, that where it gets really complicated because you can do things but cannot over exaggerate over things because there are probably more ways to do mathematics in Python than SQL.

NEW SPEAKER: For instance if you want to calculate the distance of the object -

THEOFANIS DESPOUDIS: You can do it but it's really tricky and it's gets into the rabbit hole experience, you're trying to find something that it might be easier to solve with another method. So that's how you have to poor prioritise and say OK what is the data, what do I have in my disposal, what tools do I have and what's the end result? So, you can try a lot of things but you may be - it may be not worth it sometimes so you may have to just use Python.

NEW SPEAKER: Of course thanks.

NEW SPEAKER: One very quick last question.

NEW SPEAKER: Direct response on that. I'm not completely sure it's 1.8 or 1.9 and there will be some statistic functions in Django {inaudible} that have been launched a couple of weeks ago, at least there is something in that area going to happen.

VINCE: Thank you very much. {Applause}.

DANIELE PROCIDA: While we're getting connected to let you know at lunch time lunch will be served both on that side and in the other dining room at that end so you can go to either one. Peter Finch's city walks are both now fully booked up for this evening and tomorrow evening, so if you've got a ticket you don't want please cancel it to give somebody else a chance. If you are going to the clink tonight or tomorrow night it's not really compatible with going on the city walk so cancel the walk for that night. OK.

VINCE: Before we start remind everybody about the photo that will be in this room, we'll do that just before lunch. Which brings me to introduce Yulia who will talk about load tests {applause}.

YULIA ZOZULYA: Hello everybody my name is Yulia and work for {inaudible} we develop tools for developers to make their lives better and the most recognisable in this community would be of course {inaudible} Python but we do not only make desktop tools. We also provide a bunch of services such as tracking system, bug tracking system or CI servers or code review tools and when we started our latest web servers which was review tool named outsource we decided we want to somehow re-think the way how we do the performance testing. And this thought brought us to a lot of questions for example the first one would always be what tool to use to do the performance testing?

There are a lot of tools out there from {inaudible} ones to encryption languages which provide a lot of frameworks for you and they can be configured via XML any files so vast majority of options are really hard to cover in one slide.

So if you actually select scription language to do performance testing you might want to understand why you do that and choosing Python over other scription language might not be a question in this community because you already use that language to the web server development.

And why to use scripting language at all? It's quite easy to load servers which require some complicated logic for example for the game in our services me even use sockets for the task and it's not easy with IO oriented tools or if you need complicated data over complicated requests over performance testing it's also easy just to write code tan to fill all those strange forms with a lot of fields and stuff.

Still, in Python there are a lot of tools which provide you such ability and I will cover only few of them and I would like to start with funk load here because it is the simplest one. As you can see it looks a lot like simple unit tests here with test set up and tear down method so you get code in easy recognisable matter and has special set up and cycle methods which I'll explain later. It configured with a in you like configuration file. It may scary because of a lot of parameters but it helps you to tune to your needs. And it can be easily reduced actually. So it is not that bad at all.

So what actually happens when you run crunch runner over your written tests? it actually spawns a number of configurate number of threads one by one with a little day so you don't get all your tests run in a single moment which almost never happens in real life and in each thread test is run in a loop. During configured of amount of time and this time starts only when the last thread had been spawned so you don't get into situations where you don't have all the concurrency that you configured.

When this has ended all the tests attempted to be created are killed so this should not be a problem.

Top cycle and down cycle methods I mentioned before, those are meant to be for configuration parameters that all the users have in common for example URL, web server or some other configuration you might want to pick up from your files.

Those cycles - sorry ... those cycles form a branch in which you can configure different amounts of concurrent threads and this allows you just to compare the performance of your web server over different amount of load.

So why do you actually use funk load?

It allows you to write your tests in easy manner just like your IT or unit tests and there is a runner provided that allows you to run this in single user mode so you can use them as smoke tests for example.

And other advantage would be that it has benchmark more, benchmark - it has a distributed mode and you can easily adapt for virtual glance to your load environment just by 2 lines in the configuration file.

Other thing about funk load is that it is extremely well documented. It has a lot of examples it has a lot of tips and tricks about performance and I really advise its recommendation.

And the last thing would be that it is written in a way that it can be easily extensible. In our company we are actually over written some methods of default bench runner so it would communicate with our CI server and provide the data to our default graphics and it was really easy actually. But why not to use it? The main reason would be it uses default Python threading module only and that's quite a problem because Python threads can use only one cord, Python internal architecture, and it might be OK with IO tasks but if there is at least something you do which is CPO which loads your CPU, you will have ... hmm... sorry, just a moment ...

There might be a problem with that and you won't get to use all the resources of your motive {inaudible} glands(?) and ..., and the other thing about funk load is default reports of this framework is not really good because there is only one type of graphics here and it is not configurable at all, you just get the timers that library think you should get and you are not able to influence it in any way.

And a lot of problems of - some of the problems of funk load library can be solved with multi-mechanise framework and the core here of this framework is transaction class basically an object which should have a run method defined. And it has configuration file in your like - just like the funk load library. It has a lot less parameters here so it looks more light weight than funk load. So it would be easier to start with it actually.

So, what happens when you run multi-mechanise runner over your written class. It loads your transaction into configured number of threads which is spawned one by one with a little delay and it seems a lot like funk load up to this point but the main difference that those threads can be formed into user groups and those user groups are spawned not in different processes using different processing model of Python. Its user group is started simultaneously and if you want - you may turn up the load with several user groups and configure amount of threads so you get load you need.

So why to use multi-mechanise, it uses multi-processing along with threading so you use multi-core with your clients easily and default report graphics is more configurable in multi-mechanise, just for example you can configure your own timers and it show you the results and graphics that you really want to see and actually it has more types of graphics - more than one - it is quite easy - and it looks better for my taste because it uses different graphics back end so it's generally better.

But why not to use it? For me API is not as obvious as simple unit testing because you are not get the idea from the start which helps object would be peak topped by the runner which are not and it's quite frustrating not to understand what's happening and the other draw back would be that distributors work flow of them is not supported out of the box and you will have to come up with the solution to that yourself, for example with some other Python module which you prefer.

And the last but not least would be rather fresh and actually developed framework called locust IO. The runner actually forms randomised locust which will fled your server and the rules of how they will be formed actually described in the related tasks set class which is basically set of tasks. The frequency of each task for each locust are defined with weight attributes and weight time between those tasks are also is not fixed but in some interval so it's getting really randomised here.

And what actually happens when you run locust scripts. The key difference here is that this runner doesn't use threads at all. It uses {inaudible} and greenlets and those are proven to be more efficient in IO tasks than threads and ... that would be it I guess. So while locust IO? Greenlets are much more faster with IO tasks than default p.threads and thing I forgot to mention locust runs light weight web, web server in the background saw you can share the results of testing between your colleagues or even you can monitor what's happening in the real-time or you can change parameters of the load and it is really convenient.

The other thing would be that distributed work flow is supported is not that easy as in funk load but you just need to run slave for agent on your clients so it will, you will get distributed load here.

The other thing would be you'll get total heterogenic tests it's not like in other frameworks where you get all your tests run in a loop one by one. It is completely randomised and that's a good thing because you get something like real life experience here.

And last but not least it doesn't have any configuration files, all the configuration is done in Python code and there is not much to consider actually.

The only thing why not to use locust is it has rather tricky terminology and all this weird weighted randomised relations can be really hard to get into but once you get into you will see the bite brightness of it.

And all of the examples why not to use Python for performance testing and by Python I mean Python 2 here because all of the frameworks I've covered today can be run only under Python 2 even the fresh locust guys picked up Python 2 and if you already migrated to Python 3 I am sorry for you guys and the reason would be is level interpret to log which doesn't allow you to fully use all the resources of your load glands and you will have to end up with some solutions with multi-processing or adding clients to your load cluster.

In je(?) we ended up with using multi-mechanise along with jeneta(?) because we need a big load here and that is the reason why we abandoned Python because for the performance testing is not as perform itself.

So if you have any complaints about how the things are going in performance testing, please share them with me, I know that feeling, I've been there. Thank you for listening. Do you have any questions? {Applause}.

VINCE: Thank you very much. I see 4 people have gone up keep the questions short in interests of time.

NEW SPEAKER: Do you know of any of the tools you mention have some kind of functionality that allow you to see the synchronised impact on the server like the CP usage, memory usage and the server -

YULIA ZOZULYA: In Python no they don't but jeneta I mention that use Java it has this.

NEW SPEAKER: Thanks for your talk, I always struggle with understanding the output of my load test, I can run them compare them with previous runs and get a sense if it's faster or slower than previous ones but do you have any tips on how to get more information to that? More knowledge?

YULIA ZOZULYA: Um ... actually what knowledge do you mean?

NEW SPEAKER: For example I would like to know which URLs should I consider for improvement to make my service health better in general based on globe testing I guess that should be possible.

YULIA ZOZULYA: Actually if you see on the graphics that all the tools are or providing that some of the requests are not replying in the way you need them to, you can compare them between each other and actually a lot about impact on each URL as in locust 2 you have weight attribute for example which defines how much users will use this URL for example and if those weighted URLs are not performing really good probably you should look into them at first and only after that compare them with another for example. I guess that would be my answer.

NEW SPEAKER: Thank you very much.

VINCE: One last question I'm afraid in the interests of time.

NEW SPEAKER: .

YULIA ZOZULYA: I am here for the rest of the conference.

NEW SPEAKER: It is not really a question. Actually I'm more keen that modular cloud services team and we have the same problem each time we want to release a service and we built a tool that is called load and can spawn the test on many machines at the same time and it chooses statistics D to get some real-time statistics as well as CPU and memory and you write it is almost like funk load, you write a send are you how you want to test, what a user will do with your application and then it runs it a lot of time and it break everything. And we built second version using doc S and now you can write your loader in any languages then spawn 2000 machine running your documentation at the same time and break everything again so if you are interested contact me.

VINCE: Thank you very much. Thank Yulia one more time. {Applause} various people pointing at each other. We're going to do the photo now.

DANIELE PROCIDA: So you'll direct us for the photo. You are the photographer. You give instructions. If you want to be in the photo stay here for a couple more minutes. It only takes 150th of a second to take a photo. The Django for social will be on Wednesday not today.

(Lunch)

Welcome back from lunch everyone, just a few quick reminders. There will be a kind of gathering at about 2:00 o'clock during the lunch break tomorrow on one of the tables tomorrow, anybody interested in the Django formation for social good forum or platform or group or whatever we want to call it.

Speaking of social good, do you want to be responsible for making someone else go hungry? Or do you want to be responsible for the unnecessary slaughter of an animal?

So, I think not.

So, would you visit a web page if you could help avoid that? Okay.

Well you can.

All right. Because and you need to. Because this is our catering situation for the Thursday and Friday. Those numbers still do not make sense. Unless many people don't want to go to the party on Thursday evening? Which you should not miss because there will be a lovely barbecue with plenty of food. So, if you are coming to lunch on Thursday, staying for the sprints you must tell us, if you are coming to the party that night, tell us and the same for lunch on Friday.

This is how you do it, go to your e-mail and you find the message that is -- hang on, -- find the message that says your Djangocon ticket. Edit details, it will take you the your ticket and then you fill in the details for us, please do it right now.

Thanks to AOTV busy filming these days, put the videos on the internet for us, once they finish the task of editing for us, thank you for your team and your work (APPLAUSE).

If you are interested in taking up the opportunity to visit a counsellor from the wellbeing support team, the appointment notes are there on the board at the back, pick some up. There are some available for today and more available for Wednesday and Thursday.

Don't hesitate. Even if you want to sit down and say, I am not really sure why I am here? You might find that starts a conversation that leads somewhere. So a lot of people have gone in for counselling sessions not really sure why or not even sure what is bothering them but the fact they start talking is what helps.

I would like to welcome our next speaker, Kat Stevens so if you want to come up Kat and plug yourself in.

Kat is going to be talking about Django development in a small business, under the title the full stack octopus.

KAT STEVENS: Afternoon, I am Kat Stevens, the h web manager for, we are a fine wine and spirits merchant based in London,.

FROM THE FLOOR: Yes!

KAT STEVENS: Before, the question everyone asks me, is do I get to drink the wine, the answer is not as much as, the answer is not as much as I would like. I am here to tell you about being a full stack developer and some of the challenges and the advantages.

Here is my octopus, and, there are plenty of legs going on here. We have got all the Django side of things, on that side. We have got content we have got a design and user experience.

Admin and all the management and project planning and don't worry, our octopus hasn't come across a nasty accident, he is fine, we will come on to that leg at the end.

Okay, so how am I going the fit all of this into a twenty minute talk? Well, there is not as much code in this presentation as I would like but, that is the story of my life. [screen].

So, where do we start? When I was first joined the company, the company had did have an existing website and it was built in ".net" several years old, wasn't responsive this was 2010 didn't interact well with the companies CRM, so lots that needed improving for an expanding business, so I basically had a free slate to do whatever I wanted. My manager was very trusting and said like, if you can prove to me that you know, Django work then great. You know, you can go and ahead and do it. I decided to go with Django, it is something completely different, different to ".net," but different to what I had done before, never programmed in Python before, I had only recently come across Django because some of my friends were using it. I thought I would give it a go with a couple of small prototype projects my boss is keen on small events websites and things that would you know, branches of that business that were going the be temporary, this seemed perfect to try out Django, one of the prototypes I did was for Bordeaux enprimeur, every year around April, May, the wine industry goes crazy, everybody ships over to France for a couple of weeks and samples new wines out of the barrel. Now this is basically wine futures, you get people investing in wine that doesn't exist yet. Not going the be in a bottle for another two years, my challenge was to make a prototype site with a shopping cart and stock import for a product that didn't exist.

That meant I didn't have to worry about hooking up to any e-commerce, so it didn't need you know, a vast amount of security or a, it wasn't, we weren't dealing with purchases of fine wine that costs a thousand pounds, there was no money changing hands on here. Payment providers get nervous if you try and sell wine, products that don't exist. I think on-line selling regulations means you have to be able to return it within about six weeks and if it hasn't arrived yet then that is difficult.

It all worked pretty well. I managed to reuse some of the modules as part of the main site which is great.

Yes, I proved to the company that Django was something that we could be using and he was happy, I was happy. There was no one else to argue with, just me. So, really straightforward.

Okay. So on to the Django. This is the fun bit. My favourite part of the stack, this is why I am here. I mean I love all this stuff and you know, creating like a nice database model that actually reflects all the complexities of the wine industry, it can, it is very complex trust me.

Even getting down into the nub of the Python coding and fixing bugs, I love all this stuff. As you can see, I love writing stupid snippets of code. I love the open-source nature of it. So if something doesn't quite fit for my use, I can actually just, or even if I don't understand something, I can go in and look at the source code, that is something I could not do with.net, with Django, open up the file and say, great, here is a function, I can go in and see what it is doing.

Also custom template tags as well. This is something I have picked out, out of the huge amount of the things that I like about Django. Throughout the e-commerce site was having to use currency exchange rates because we sell wine all over the world and having to translate things into Hong Kong dollars, being able to use custom template tags and apply it to the exchange rates, it was so useful, adjust it to my needs.

I can lose track of things sometimes, I, it is a large complex site now and I really, it is really difficult. It is hard trying to keep track of things when it is just on your own, I, I do run into trouble and thankfully, the debug tool bar come to my rescue. One of the things about working on your own, you don't know that tools like this are out there half the time. There is no one to tell you, oh great, if you are having trouble with knowing what template you are using, why not use the debug tool bar, you can just put it into your settings file and that is fine. I didn't know it was there, until I was Googling trying to find out how do I know what template I am on? The debug tool bar, when I first started to use it, it was difficult because I was new to Python and Django and it kept crashing. There was some incompatibilities with some other plug ins I was using. I almost put it to one side and then forgot about it almost. Then again I came across a bit of code, a few months later. I am stuck. I can't understand why all these my queries are mounting up and up and I need to see what is going on under the hood. I remembered the debug tool bar, there was a update and it stopped crashing, now I use it all the time. Couldn't be without it.

Python code, I really get enthusiastic about writing a clean efficient views and queries. I also like cleaning all the things in my forms.

Being a sole developer, only have to merge code with myself, if I am coming across code I wrote six months ago, it almost feels like I am a different person, so I write jokes to myself, so I can read them six months later.

Libraries I want, I don't have to worry, I can try it if I don't like it, I can get rid of it. One thing I really did like was the tumbler, I know that the Django tutorial has a Blog and I did not use a Blog app in my website, even though we have a Blog. I used the tumbler API because it was easier for our wines, our sales people, out in France tasting from the barrel, yes, this one tastes good, I will take a picture and put it on the Blog, easier for them to do that, rather than log into the Django site that I had created and mess around with the admin. With tumbler, they could log in and up load the photo. It worked really well for us.

So, as well as being enthusiastic, I can get carried away. There is steam coming off the keyboard there. That means sometimes I can reinvent the wheel, as a sole developer then I don't know sometimes that these libraries are out there. Same with the debug tool bar, I just, there could be, I could be spending hours and hours on writing a piece of code. Someone has already done it. I never know. Yes, I don't even realise the wheel exists, I have to say, I didn't even know that this conference existed until one direction had to make a move! (LAUGHTER).

One of my friends tweeted saying, friends tweeted saying, check out what one direction have done now, they are up to their own tricks. Django! I work with Django! maybe I should go? Say thank you to Daniele and all the organising team who encouraged people like me first time speakers to come and speak here, it has been great. I really enjoyed myself.

So, finally with the on the Python side of things, I don't get any technical feedback, I have no idea whether I am programming well. My code could be amazing, could be awesome but I have got no idea, I know one of the talks on Sunday was about imposter syndrome, overcoming the feeling that you don't know what you are doing, that everyone else knows what they are doing a lot better than you do. I really don't know, I mean, not only do I work on my own at the moment. I have always developed on my own, never worked in a team of developers. It has been really a real challenge to actually try and be confident in my code and make sure that the company is confident in me and I am happy to deploy this code and make sure that our website is going to run okay. Yes, I think we have done all right so it has been a learning process.

So, I am going the quickly run through my advantages and disadvantages with testing and bug fixing. If I don't like something, I can ditch it or better yet, convince my boss we didn't need the feature in the first place.

There is no need for a complex ticketing system, people can shout across the office to me and say it is not working or better send me an e-mail. Everyone knows to report the bug to me. Things don't usually get messed because I can keep them in one place, I can decide whether something is important and how to prioritise it.

I can actually deploy small fixes quickly, if I know it is a typo in a template, I know it is not dependent on other things, I can get in there and do it in half an hour, I don't have to worry about the complex methods you are doing, when you have to make sure that everyone on the team is okay.

But, testing. I don't do enough of it. I am sure no one does enough of it but, we work in a high pressure company. Things need to be done yesterday and when I have to turn around a fix in twenty minutes that means that testing is going to suffer and it always does. I need it more than ever though, if I am trying to deploy something quickly, then it is going to you know, I need a robust testing system to make sure that you know, I can be confident in that deploy.

I keep, definitely don't have any of these in my code at all. Comments saying to do, fix this later.

I am sure none of you do either? But yes, moving swiftly on.

Thinking out loud as well. When I am trying to solve a bug, then I try to think out loud at people and try and put the problem into a sentence and see if that makes anymore sense to me, my poor nontechnical colleagues have been on the receiving end of this and nodding and smiling. Yes, I lost you at Django, sorry, but it helps me to even say it out loud.

But, sometimes I just get really stuck. So stuck overflow it is.

Obviously it is very frustrating when I am sure again, most of you have this problem where you type in the exact question and only it is there but no one has answered it. Or you find it is actually you that answer that question, that ask the question six months ago and still no one answered it.

So I will move on to another leg now. That was my Django bit mostly done.

In terms of design and user experience that is where my background really is. I have been designed websites since the late 90's, lots of marquee text, terrible under construction gif's, you get the idea, having the front end experience helped me in terms of saving time. I know what works and doesn't work in terms of html and J query and, so I didn't have to learn new things, obviously I did learn new things but didn't have to start from scratch there.

And one aspect of working for a small company, we really have a good idea of who our client base is. We know that well, they are fine wine lovers, usually male, over 40 and a large proportion in China and Hong Kong, so we know they are quite tech savvy, we don't have to support IE seven I was pleased when I learned that. I do have to support mobile responsiveness. All the time I was saving not having to worry about IE seven I put into making sure that it works on tablet and mobile.

So all those, trying to save time, on the design side of things. It is completely scuppered if you are having to use photo shop. If you have got your development server, men cached, four different browsers Skype, Spotify and photo shop running on one poor little lap top things start grinding to a halt. I have wasted more time rebooting to try and get photo shop to work, than I have fixing bugs I think. So the solution I found to this is to make friends with your Mac using print designer and be able to be able to send files home and hopefully they can sort it out. I am sure some of you are thinking, why use photo shop? I don't know, never used gimp, been using it for the last ten years and I know I can use it quickly and well, hopefully quickly, I know where all the commands are, I know what I can and can't do. Whereas gimp I would have to start again and learn those things, I don't have time to do it. It will be great to use all these nifty bits of software, but this is what I have got and this is the tool I am using, and everyone has to deal with it.

Okay so my other leg is content. There is no point in having an empty website. So, one of the main challenges I had to deal with, importing our stock data, all of our wines from our CRM built in MS windows sequel server, I was running my sequel and I had to support ETFA characters, wine names have circumflexes and accents in them. Translated into Chinese as well for our Asian market.

How did I fix it? I can't remember.

I don't know, I can't remember, something to do with OCDV drivers, I don't know, it was two years, I know where the file is I can't keep all the things in my head, in doing the full stack, I don't know. I mean I spend most of my time concentrating on Django, that is the bit I love most.

I do not enjoy dealing with ODBC drivers so I just write myself a note, I know where the note is. If I need to change it I have to look it up so this will remain a mystery to you for ever I'm afraid.

As well as the actual typing of data BTFA and things it's the actual wine data itself causes massive problems.

OK burgundy, any burgundy fans? if you are a burgundy fan you will be able to see that 3 of these are actually referred to the same wine but one of them costs half the price of the others. So who thinks it is A? Hands up if you think A is the different one the cheap one? OK how about B? Yes, some hands of course. C? Yes and more people there and D? OK most people going for D.

It's actually B because it doesn't have the Chaumes vineyard there, my knowledge was nowhere near enough good enough to deal with this information and I basically had to get some help.

So these dogs do not have burgundy in their barrels but basically I didn't have time to learn a vast amount of wine knowledge so I got some help, I got an assistant who had some wine knowledge. While they weren't very technical, they could use the Django CMS. So I used, customised the admin as much as I could to try and help them out.

Also with the Chinese translations, my Chinese is worse than my Welsh so I could outsource those and I made use of Django's i18n ...

NEW SPEAKER: Internationalisation.

KAT STEVENS: OK. As well as the fun stuff I have to do boring things, reporting, analytics, invoices, where are the customers coming from, is the site making a profit, is it paying for itself. I have to do all this custom management as well as super tedium stuff renewing domain names, liaising with ad agencies, terms and conditions, being on hold to Barclays trying to explain to them no we're not going to sell wine futures, you don't have to Murray about these 6 week on-line selling regulations, I had to sign a bit of paper promising I would not sell {inaudible} premier on free web-sites and SYS admin my least favourite aspect terrifying for me I would have outsourced it if I could have done. It is something that fills me with horror because I just don't know enough to really be confident in what I'm doing here so I just have to do my best. In terms of security, well there is just me really so I don't have to worry about complex user permissions. And I could do a quick response when the heart bleed bug came up. I saw it come up on twitter and raced into fix it to try and figure out what was going wrong and then 24 hours later my hosting company sent me a nice email saying you should really fix heart bleed yeah. And a week later my boss comes racing through the door saying have you done anything about heart bleed? ! Yeah don't worry it's fine ...

Some problems are just beyond me and we are going to get some help try and outsource this and make sure everything is up-to-date and secure.

So our octopus sushi leg. Basically working in a small company, you know there is so much unpredictability that goes on, for example I've had to proof a 70 page cook book with matching luxury wines as a gift for our client base. I had a 6 month secondment where one of my colleagues left the company and I had to do email marketing for 6 month's that was fun. I am at the mercy of third parties especially when Barclays are hanging around for 6 months putting a block on our development time line. I also made an HTML 5 roulette wheel for my boss with wine names and things on for an event he wanted to do and unfortunately I don't think this is going to rotate very well - hang on is it going to go? No, that's it, it keeps landing on drunk so there we are {laughter}.

So to summarise, I have learned so much doing this project over the last 4 years or so especially how to research stuff how to look up things on stack overflow and make sure using the right name for everything and I discovered I really love Django and it's something I really want to do more of in the future. Definitely put jokes to yourself in your comment. I recommend that rather than song lyrics because it's what is this? I didn't remember at all. And if there is a problem I'm working on my own so I can reprioritise very quickly and easily.

However, things do get missed especially testing, it's definitely something I need to work on but it's always going to be difficult.

Holiday I haven't had a proper holiday in about 2 years. Generally being not on call but if the web-site does go down any time of day then I'm going to get a text message from somebody. And I've got no time to really get an in depth knowledge. I've got a broad overview but apart from Django I've been really trying hard to learn more about it, I don't consider myself an expert in anything.

Also I've got poor attention span, having to flit between all these aspects of my job so it's amazing I'm still here rather than wondered off out the door somewhere. Also the Barclays hold music is one of the worst things I've ever encountered! {Laughter}.

So I would like to give a big thanks to my friend Pippa who drew the octopus for me and she's at Pippa Alice Art. Anyone have any questions? {Applause}.

DANIELE PROCIDA: Thank you, that was absolutely brilliant and if I'm allowed first go at the questions I just want to say even down to the dates your career path seems to have mirrored mine yet you got away more lightly because I had to support 6 up until a few months ago at Cardiff university, that's not a question but thank you and I hope we see more of you in the community because we need people like you - any more questions?

NEW SPEAKER: That all sounded very familiar. Just one question, how do you make your boss less worried about you being run over by the proverbial bus.

KAT STEVENS: I have very good disaster recovery plan in place which involves hiring my identical twin. Yep ... {laughter}. No, it is a problem, and basically this is a problem I encounter right now because I'm going to leave the company and start a new job in a couple of weeks and they've been desperately trying to hire someone to replace me and instead they've actually decided to outsource the entire thing to an agency who's got a couple of members of the team who can take on the different roles.

NEW SPEAKER: 2 questions. One, you said you were working on a template tag that does dynamic pricing based on world markets and stuff and I'm wondering if you ever open source these kinds of fringe bits you write and could you put it on get up so the rest of us could check it out?

KAT STEVENS: I'd love to I'd have to run it by my legal team at the company to make sure it's not covered by -

NEW SPEAKER: Don't tell them just put it out!

KAT STEVENS: It's definitely something I'd look at because it was a lot of fun to do and would love to contribute something like that.

NEW SPEAKER: There was a couple of times in your talk where you explained something you had a tough time and I was like oh I know how she could work on that but you don't have your fitter up there so I can't tweet you.

KAT STEVENS: It's on the first slide, Kat Stevens.

NEW SPEAKER: Hi, thanks a lot for your talk a lot of that sounded familiar. I didn't have so much a question as a sort of a feedback on some of the things you touched on. Personally I'm in a very similar situation where it's very much a one person show and I'd really encourage you to invest interesting because that is sort of what's kept me sane for the past 2 years knowing I can roll-out something in production and it's not all going to explode in my face.

KAT STEVENS: Absolutely, I mean testing is something I've definitely done more and more as the years have gone, as the project has progressed just because the project has become more and more unruly and like I said I have a poor memory and I can't remember what affects everything else so having like a strong testing framework is essential.

NEW SPEAKER: Hi, that was a great talk. My question is your job sounds incredibly stressful and you said yourself that you are scared of the whole SYS admin thing. And I just wondered what was your background and what made you think when you applied for the job, oh, this is something I can actually do?

KAT STEVENS: Well I looked at the job spec and it said we're looking for a web designer for a wine company so I thought internet and booze those are my 2 favourite things {laughter} and I didn't really spec out what it was and I wasn't working at the time so I thought I'll go for it and see what it is and the company was really exciting and the people were really great so I thought well, it seems like a really good opportunity for me to explore this and be able to see, do the full stack and see what I enjoyed doing, find out what my strengths and weaknesses were. In terms of background I have a computer science degree but I stopped programming pretty much as soon after I finished it because I got disillusioned with the whole theoretical side of it and wanted to do something practical and doing web-sites was much more appealing to me than writing furrier transforms and things like that so as you are doing web design and have a chance to have a free rein to do whatever I want was a great opportunity. I want to say thanks to my boss for being so supportive for the last 4 years.

DANIELE PROCIDA: Thank you very much Kat. {Applause}.

NEW SPEAKER: Hello everybody, like to introduce Tom from Joinerysoft; Joinerysoft is the UK's biggest supplier of joinery software and talking about using it in a Django application so hello. {applause}.

THOMAS TURNER: I did it the wrong way round ... other way round, start again, that's better. Right so it's a bit unique this talk. It's using Django in a desktop application. About me, I'm Thomas Turner, I own 2 - am part of 2 family companies. One is somcom which does technical web-sites, done stuff to do with people like NICEIC, which does all the electrical stuff and has done technical searching web sites. The other company is joinerysoft which does joinery software for joinery manufacturing or carpentry. I've been using Django since - for about 7 years since 0.96 I think, there is my twitter account and git hub account.

Using Django as a desktop application. Is it a good idea or a bad idea? Let's find out.

So, as I said, the programme that I'm talking about is joinery programme which is called JMS so if I call it JMS, I refer to the programme. So JMS, it is a desktop application but is running Django so let's find a bit more information.

What technologies do we use? Python and Django obviously otherwise I wouldn't be here. It's an MFC application so it's Microsoft foundation classes. We use a library called DHTMLx for all our grids and all the sort of tables and all our sort of framework. We use reportlab for all our printing. And we use fire Bird SQL database and we use CEF as our web browser which is basically Google chrome. And we use openGL and sikuli for testing.

Why did we choose Django for a desktop application? It makes it future proof. So, in the end we want to make the programme go on to the internet to actually go on to as a cloud service. It's faster writing - well can be faster writing in Python than C++. HTML can look better than normal Microsoft dialogue boxes and becomes more network compatible and we did a similar thing before.

What problems did we have? Lots! One was securing Python and Django code. One was securing the HTML code. And one was finding a Django web server that could be embedded into C++. We also found there were too many programming languages. It's quite a nightmare that is. Finding a database engine that is simple to install. Our customers don't even know what a mouse is so they wouldn't know how to install postgres. It gets picked up by virus checkers because it's using port and it's harder to distribute tan the normal bog standard application and it's a nonstandard set up. And testing, we had a lot of problems with testing.

So, securing the Django and Python code. We put our own encryption into python. We took Python and found where the - so we took Python and put our own encryption into it. We had to build Python and so people could stop reverse engineering because we only distribute the PYC files not the PY files and of course you can still reverse engineer PYC files to get back to a PY file so we stopped that by putting our own encryption into Python.

We also had to modify Django to allow for the PYC because there is one bit that says asterix dot PY. It would be great if it said asterix dot PYC.

We also had problems securing the HTML code. The HTML code we didn't want our customers to be able to change it and do something wacky on the page so we actually put our own encryption I won't go through all the how we encrypted down the side. So we wrote our own template loader to do it first of all I didn't even know template loaders existed so I overrode all the template class but then we worked out that you could override the template loader.

So, finding a web server that could be embedded into C++. We tried Django development server - that's a really bad idea, at the time we were using Django 1.1 and that was a bad idea because it wasn't multi-threaded. It says not to do that.

We tried somebody's project, somebody's own C++ web server and that was too slow and too unreliable.

So, we settled on cherry py. Cherry py if you have used it, it is all written in Python, it is a very good web server, so we embed cherry py into our C++ app.

So finding a database that is easy to install. As I said earlier, our customers don't even know what a mouse is or some of them don't. So they wouldn't have known how to install postgres and postgres actually conflicts - because this is a windows application, it actually conflicts with other software. So, we found it's easier, so we couldn't really use postgres. And SQ light is not very good for networks. Our current version which is out there in the field, we use both of those products and we've had problems with both SQLite and postgres.

So the one we chose was firebird. Has anyone used firebird? Anyone know about it, put your hands up? A few of you but not many. So, it's easy to install. It's open source. It is not owned by oracle. And it works with Django using a third party plugin.

It actually came from a D base so there is the logo.

So, finding a - obviously we needed to put a web browser into the programme so we had to embed a whole web browser into our C++ application. We tried to embed Microsoft internet explorer and me we ran like a mile. It's a nightmare, we didn't want to go there. So we went for CEF. CEF is basically Google Chrome.

Why CEF? It's open source, it's cross platform, it's used in many big applications. I expect a lot of you have got these programmes installed or some of them installed: spotify, steamclient, github and Dreamweaver.

So what CEF is, it's a web browser, it's basically Google Chrome so yes it can be embedded thanks to somebody taking Google code and putting it into - making it so you can embed it into a C++ application.

So, another thing we had problems with was testing we couldn't get selenium to work because there was no web driver, we went for Sikuli, it works by image recognition, it is a bit of a nightmare, the problem is, if you do a star sheet change, you have to redo all the tests again. So that is basically why we had to use Sikuli.

Reports. We needed a way of generating our own pdf, we, a Django application will be running on a different thread to what the C plus plus is using, so we can't directly talk to a printer, so we had to go via a print preview. So we had to generate a pdf but settled on reportLab it wasn't up to the job, so we had to modify it. It is sort of written in Python, not well coded it is actually written like a C plus plus act, one letter variables, so, that was another problem we had.

So, yes. It was obviously hard to understand the code. So I am going the show you a little demo of the programme now.

So this is right. I can't see that side. Let me go over here, forgotten not duplicating. Hopefully. I can work out which side it is. Right. So, that is the database there obviously so it is a single file database and I am just about to call the programme up here. Yes, at the top is all the files, so it doesn't really look like a Django application.

So back to call it up. So to a user, they wouldn't even know it is written in, they wouldn't even know it is a web application. So back to log in. Obviously that is Django.

Obviously this is all written in, this is all the html web pages and obviously these are all in Django as well. About to get something more interesting, so don't fall asleep yet.

So about to call up a window. So. All of this is written in, apart from this bit here, this bit over on the right hand side is actually written in C plus plus, so we had some difficulties getting information from Python to the C plus plus engine.

So back to come out. Again that is all written in Python, the costing. Back to print a report.

Obviously this is reportLab so we had to modify this so it worked for our requirements, this is the sort of thing that our customer send out to their customers, so that is it anyway for the thing.

Go back to that. Slide show from current slide. Not from the beginning.

Be there in a second.

Right. So was it a good idea? You tell me.

We are currently recruiting, do a small sale pitch, just in Bristol, if you want to get more involved, join our team, any questions?

(APPLAUSE).

DANIELE PROCIDA: Thank you very much Tom that was absolutely amazing. I don't think anyone here has seen anybody try to do something like that. Was really Django the best way to do this? Was there, surely (LAUGHTER), I mean it is.

THOMAS TURNER: I wouldn't do it again. No! I proved it could be done, I proved its been done.

DANIELE PROCIDA: I think that shows another round of I (APPLAUSE). (APPLAUSE).

I am slightly lost for words because that is honestly one of the most radically, different ways of using Django that I have ever seen, congratulations.

THOMAS TURNER: I was expecting them ...

FROM THE FLOOR: Excuse me, I am here mostly to disprove what Daniele just said about 2009 I think I did something similar. We wrote an outlook plug in actually which was presenting html generated by Django for very similar concerns of making the thing, network transparent with intention to later make it available from servers.

We also, we had a lot of the same experiences and just wanted to say that I share a lot of your ...

THOMAS TURNER: Thank you very much.

FROM THE FLOOR: Thanks for the great talk, different application how to use Django, I thought about using Django or in particular for application for desk top application, basically as a data source for (NAME - INAUDIBLE) would you recommend me to do it or not do it?

THOMAS TURNER: Just don't do this! just stay away seriously, just don't do desk top, it is too complex! I just would stick with a desk top, stick with something like C sharp if you are going to do desk top work.

FROM THE FLOOR: Hello, thanks for the talk. NFC and Django, that is kind of a weird combination, is there a reason why you didn't try to use something like GDK or something like that?

THOMAS TURNER: The reason we did it we wanted to go on to the internet. My other company did web applications, I prefer doing web applications than desk top applications, so I probably pushed more for doing this than we should have done it.

FROM THE FLOOR: Yes but for the desk top part why not use GDK or another Python solution?

THOMAS TURNER: Our current version is written in MSC, so it is an extension, it evolved to this. We didn't get there by taking a browser, we only had half the window as a browser at the start. That is why, we got there in iterative steps rather than one big sweep.

FROM THE FLOOR: Okay.

FROM THE FLOOR: Hi there, thanks for your talk, it was interesting. I have got two questions, one you have embedded Django in a C plus plus application, did it ever cross your mind to try and, I don't know how it is easy to do, embed the C plus plus into the Django web page?

THOMAS TURNER: We thought of doing that, our customers wouldn't want to call it up in a web browser, they would want to call it up as an application, we would have still needed an exe for them to call up. So that is why we have done it this way. We use a lot of open GL which is also rendering to the page. So, some of the views we use a, one of the main view uses a canvas which actually does all the open GL part.

FROM THE FLOOR: Yes and my second question was, you mentioned at one point your users don't really know what a mouse is. At the same time, ... to hide it away from them, worried about the users fiddling? Is this copyright...

THOMAS TURNER: The programme is protected with a dongle okay, so one of these dongles so, people would hack it but it is, it ranges from the type of users. Some of them don't even know what a mouse is, but some are large organisations which would have technical people yourselves in there which could reverse engineer.

FROM THE FLOOR: Thanks very much.

THOMAS TURNER: No problem.

FROM THE FLOOR: The question I wanted to ask was just ask before but I wanted to congratulated you for making windows, inside windows, inside windows (APPLAUSE).

I wanted to say we did a similar thing and we experienced many of the same problems you have some others with some other stuff we have a lot of hardware communication going on, I may do a lightening talk.

THOMAS TURNER: Those were a brief amount of the problems, I could carry on and on, seven years’ worth of problems. So.

DANIELE PROCIDA: Thanks again very much Tom.

(APPLAUSE).

Go and visit Tom at his stand in the sponsor’s hall if you would like a chat with him.

DANIELE PROCIDA: ... I am very pleased to introduce Shai, Shai is one of my colleagues in the Django team and he will be talking about lispisms.

Thank you Shai.

SHAI BERGER: Okay. Hi there. I am going the talk about things that Python in general in Django in particular could take from lisp and how we could do this and also introducing here a library at a time I have been developing.

The first question that you could be asking is, why lisp? What is so special about lisp? What is interesting about it?

It is not a very new language. It is, it doesn't look very nice. It is hard to read if you are not used to weird things. Yet, it gets people very excited about it. You probably don't know Erik Raymond, like fifteen, twenty years ago, he was an important voice and an earlier adopter of Python.

What really is interesting. So, lisp actually pioneered many things that we take today for granted. Recursion, dynamic typing was first seen in lisp, garbage collection was first seen in lisp. So the question is, basically, are we done pulling things out of lisp or are there still things we could find interesting there?

Or, rather, did these things come first out of lisp just because it happened to be there first, or are there things in lisp that help foster innovation and allow us to and then give reason to their being more things that we should pick up.

People who know lisp usually when you say, let's take a feature from lisp to another language usually the thing that comes to mind is macros, lisp macros are powerful. People start adding similar features to Python, perhaps ... does that and it was rejected. There is libraries that does that, similar, sorry, does limited macros in Python.

But, actually that is not what I want to talk about here. I will have more about it if there is time later.

I want to talk about dynamically scoped variables.

So, what is the problem that dynamically scoped variables solve?

People everywhere for years have wanted access global access to request. Also to other of things it was just last weekend I think someone asked on the developers list for the current app to be available globally. Sorry. Whenever people make those requests on the developers list, it usually get the same answer: No. Globals are evil.

Why are globals evil?

Globals are evil because when you use globals in your code your piece of code is tied down to a specific system and you can't pull it out and reuse it. Because, when you are using globals and you call out to other code that other code can change it under your feet and it is hard to reason about the behaviour of your code.

All that is only, if you use thread locals, you can use those and things like thread safety to worry about.

Yet, evil is more than about being bad. You don't hear six hundred line functions being described as evil. Don't hear spaghetti code being defined as evil. Why are we saying globals are evil?

Because they are mighty convenient!

We like it that we can make ORM queries without passing the connections explicitly, we like to have access to settings whenever we like. In normal Python not related to Django, we like to have the STD out available by default for print and available not by default if we needed to press it.

So, globals are evil because it is hard to control them.

But what if you could have globals that do behave?

Globals were the values can only change down the call chain, so they can change from under your feet. Like a "with" statement where everybody change to the global would be like a "with" statement. Then each piece of code makes sense in isolation, you don't need to know what all your calls do to realise what your globals do.

You can set up local environments, so you can reuse codes that reuse the globals.

So, back to lisp. Lisp local change is the norm. Use let for assignments. What let does is not what assignment do in Python, but introduces a new binding, when ... disappears and then all bindings come back. After the inner let is finished, that is returned is the a from the external let and its values.

Equivalent in Python, you into deuce that shed all the old bindings, this is the equivalent Python to the previous lisp.

Let's keep this because it is not really important and we are short on time.

So, for Python, you still have the "with" statement in mind right? The thing is for Python it is not enough to introduce those bindings with "with," because Python likes objects and mutables with lisp with the examples we have seen before, all the values for atomic. With Python you can change parts of the variable without changing the variable. So, that would not be enough.

Then the obvious problem with the code, you give it a global, it changes its inner parts, you are back to where we were before. So, we need to make the objects immutable if we wanted to do this.

Slightly less obvious problem. Sometimes we do want it to change even in a restorable way we do want to change just part of the object and then changing bindings will again not be a good solution. So, we'll use something special for attribute change.

What we do and now I'm introducing the solution. Library called lispism which explores a special object called D and its attributes are the dynamically scoped variables and this way we have them marked specifically so you don't mistake them for normal variables. And for atomic values this looks exactly like the equivalent lisp. You do with D let and then you have the values for the variables. And then this prints first 1 when the value is 1, then 3 in the inner with and then goes back to 1. . And once you are out of the external with, you get a name error.

For containers, we do special frozen containers so after this, after the first let, you see that A has turned from a list to a topple which is immutable but the thing is they're deeply immutable, they're containers that also freeze their contents so if you access - so B is itself a topple but if you access its members in the originals they were lists.

And if I have a user defined type, with its own attributes, and its own - sorry some of them are just normal, some of them are containers themself, there are 2, one can only read and the other changes state of the object.

So, as long as I have tried to just read the object I get that I can access the attributes and that's OK.

If I try to modify an attribute I get a type error because it's a frozen object.

And attributes are controlled inside methods. I can call the reader and get the frozen attributes. If I call the rider I get a type error.

And I use ORM inspired syntax for changing specific attributes of objects.

Limitations of what I've done here. First of all, the whole freezing thing is a consenting adults API. That means that there are ways to go around it if you really want to. The intention behind it is to prevent values from being changed by mistake, not to force them to be frozen. That's one way you could just change attributes on a dynamically scoped variable without using let.

This a good thing because if anyone actually went to the trouble of writing something like this I suppose they know what they're doing.

And this is not as I said, not designed with security in mind, just with engineering.

I can't control extension models. If anything is written in seasonally it's not really controlled by the pattern library that I wrote.

I can and do try to make sure that values returned from extension methods even though they - sorry, if - dynamically scoped variable holds an object with extension methods and they return objects I make sure those are frozen but I can't enforce that they don't change the variable.

There is this annoying quality of generators that if you write a generator using a dynamically coped variable you change with let and under that you do yield and the variable is changed so that doesn't work yet. I have some things to do about it in mind but not done yet.

Dict access is currently a problem but again this will be handled.

And the future directions to go with this. The library is currently public, you can't pip install it yet. The repository is public. And the next places I'm going to go is, 1, handling threading, if you have a set of dynamically scoped variables and I open a thread they should be inherited but currently they just use thread local storage so they are not inherited by new threads.

Macro py the library I alluded to earlier, I could use that to make better sub object control. And it intend to do that. I could do things like here, to do things like, for example, for I can access not just attribute access.

And another thing that is interesting and dynamically scoped variables enable is something called restarts which is an error mechanism which is in lisp, has not been adapted by any other language I'm aware of and I want to talk about restarts a little.

The idea behind restarts is that when you try to handle errors you're basically in a conflict. The code protecting the error may be in the best position to do something about it, but it is not in the best position to make decisions. Code higher up knows the big picture, knows what needs to be done, but has no access to details to actually do it. So, when you try to catch an exception higher up most of the thing that caused the exceptions to be strong, to be raised are already gone because the stack was unwound.

So, the idea here is basically that the inner code instead of actually defining what to do defines options, it defines the different restart and each restart - the example is reading a set of a log file or a set of rows trying to pass each row, and recover when things go wrong. And there are 3 options for recovery. You could return a special record indicating error, you could return a default record saying this is OK, and you could drop in a debugger. But the decision of what is the right choice is not made my read rows by the function at the top, it's made by the function at the bottom which is the one calling things, and test parsers defines how to handle each of the errors that could come about, it maps exceptions to selection of restarts.

So, with the setting as it is here, if a record shows and the parser decides it's out of date then it returns an error record, but if it calls a parsing error it drops into the bug because it's still in development what we wanted at the moment.

And another important point about it is that the function in the middle, read file, knows nothing of this. It just goes over rows and return - and read each row and collects - sorry just the function just holds opening the file, it doesn't need to know anything about the specifics of - neither of policy nor of details of error correction. And that is only enabled by dynamically scope variables.

This is like research for the future project. This is like - of course not valid python code just pseudo code.

There is a great paper explaining this in detail how it works in lisp not in Python.

So, that's pretty much what I have to say today. Library will be released on PY PI soon but for now it's just a public repository. I think greenspun's 10th rule of programming does not apply as strongly to Python programmes but it does in some degree.

That's all I have. {Applause}.

DANIELE PROCIDA: Thank you very much. We've got time for 2 quick questions.

NEW SPEAKER: Thanks it looks awesome and clever and dangerous I don't know. I was wondering, is there any library that provides immutable that has structures for Python?

SHAI BERGER: Not that I'm aware of.

NEW SPEAKER: And have you heard about HY?

SHAI BERGER: Implementation in Python yes.

NEW SPEAKER: Thanks.

NEW SPEAKER: Thanks that was great so the question is yes have you used it for anything apart from like hey this will be cool? Is it in use for like a real production system for doing that and what is that and why was that a good idea?

SHAI BERGER: Well, this is not yet in production for anything. This is just something I thought would be cool at this point. But, thought it would be call cool is the applications I have in mind for it are in Django. I think doing this sort of making things globally available because most of the things that people want globally available they want usually for read only access -

NEW SPEAKER: Request object -

SHAI BERGER: Like the request object. People want mostly read only access so we could give it to them with acceptable guarantees that they're not going to make too much trouble. And I think that the thing with restarts would be very valuable for error ending an URL(?) for example.

DANIELE PROCIDA: Thank you very much. {Applause} I am really pleased to introduce Markus Holtermann who is another colleague in the Django core team. There are tens of people in this room with whom I've shared meals and drinks and quite a few with whom I've shared a hotel room or someone's spare room, but there is only one person with whom I've actually shared a bed. {Laughter}. And so that's what we do {laughter} - I was going to finish my sentence and say that's what we in the Django core team do for you when we go to sprints! Thank you Markus. {Applause}.

MARKUS HOLTERMANN: Hi, thank you for being here, thank you for having me here. Today I want to talk to you about how you probably use forms and how you could actually use them in a quite different way so might be different, might be similar way. And the way forms can, what forms can actually do beyond what most of you use them for.

But first let me introduce you to - let me introduce myself. I'm Markus Holtermann a master of science student at technical university of Berlin and I'm also a member of the Django core team since the beginning of this year and some of you might have seen me running around at euro Python 2014 fixing internet and Wi-Fi and being pretty busy the first days but conference was cool so I was relaxed the next days.

So, referring back to the title of my talk, forms are static, no they're not. I'm actually talking about the stuff in Django dot forms just to be clear. Before I go into any details I'm going to talk to you about what forms are and what basically you use them in general for and the most important use of forms is basically input validation then you have a user that goes to your web-site, enters some data and you want the data the user enters to be somehow validated against some rules and Django does a pretty fine job at that.

Some of you might not use it for end user direct user interaction but also to validate data that comes from API similar to what serialisers do in Django framework but yeah they are basically input validation.

The other thing is forms are static. But when I talk about static, what does it even mean?

Well, first and foremost forms do not change when you run your web application. More precisely, the {inaudible} layout does not really change. So what does change when I talk about forms? Well, actually the content user types into forms but, well, that's just per request and that doesn't really make any difference to the forms, it's just some data is validated.

So, the question arises why are forms static at all?

And static I define as you have to actually change code, you have to test and modify code, you have to probably run some testing staging environments and test your bundle release and deploy it and, well, this is pretty much the way how forms are used and how Django is a good idea how to use them, but sometimes you want forms to be more dynamic in terms of I have for example a user who whose locked in and I don't want him to have a catch up field he has to fill out when requesting data but on the other hand you want the page or form for anonymous to post data as well and there you have to have a mechanism for be a to prevent {inaudible} from spanning your site so there you have some kind of dynamic form, form feed allocation, the allocation you meant remove forms from fears from the form or you add them depending on what you actually do.

But what also changes in those kinds of situations is what Ola talked about in her keynote about - in the {inaudible} talk - you can have permissions on objects and you want to do filtering on objects you have in model choice spheres and depending on what's in your data basis you have different data for {inaudible} data in the fields which is pretty much obvious.

So I honestly have to disappoint you at this point because those forms are still static because you still have to modify code if you want to make them behave slightly different. You still need to test them on the {inaudible} and deploy them.

So, you might argue now that this is actually the right way to do and yes, I pretty much agree with you. Yes, I think almost all cases but, they are situations where it does not apply.

what are nonstatic forms then, they are forms that can change the lay out, the field that is are there. The fields that are not there, the order of the fields as well and practically, this, the dynamic forms are forms that don't need to be modified on a code level. Therefore, are without necessity to deploy them again.

Well, now you might ask, why do you need them? This is actually a brief story I have had a very, a company I have worked for, for a couple of years and we had a rather small client not that, not that much, not a large, they didn't have that large budget and the only thing they pretty much did for organising events throughout the year and the event organisation or event registrations was more or less the same each time but they all of those found forms had different additional fields they had to provide. Be it a birthday, e-mail address or whatever. So we would have ended up with coding every week, another form for the entire year which is not really code base you want to maintain after some time.

So, I came up with okay, let's do something else. Let's do, let's be smarter at that. What I later realised is that the application I came up with allows you to do even way more complex things. Before I actually talk about the application itself I want to talk about you, what I am doing there at all. In the end, I create classes dynamically. So, what of you used or are uses model forms?

Yes. Pretty much everybody. Which is what I actually expected. But, model forms are not that quite static. This is actually a pretty simplified version of what Django does when creating a model form. I hope you can read it. Can you see my laser pointer in the end? In the back? You Django first takes all the model fields from the model, creates the class matter, you design on the form, and then sets the model on this matter class and creates the actual class, we have dynamic class construction, which something that Python lets you do.

Then Django lets you create the model form putting the class on there, and you have the model form class you can use that is in somehow from the form and pretty much lets you save in model instance.

But, wait. They are not really static, they are not real, they are static, they are not really dynamic, why? Because the form that is constructed is constructed from the models from the models fields and for the sake of your health I hope nobody of you actually modifies models during the run time of the application. It is, might work but you probably don't want to do that. So, this forms this model forms are still static because if you want change, you actually have to deploy the code of the modification you do to the models and then release and deploy.

Which brings me basically back to the idea of the, to the function that is Django users in this code type. Which is a Python bid in that let's you create classes dynamically, so takes the name of the class, created the base classes it should inherit from the attributes so Django is a framework and pretty much what you do with web frameworks and Django, is always interact with the database, this is pretty much what I did. I put all the information I need for form into the database. So, I have a form instance that is an entry in a database table that has some attributes and I have couple of fields that belong to a specific form and they are at rows or entry in a different table but belong to the field and when you open a view, or when you open the actual view, this application Django takes the database and creates the form dynamically, you don't have any culture changed there.

This is the basic lay out of the application, which I explained. You have the form model which gives you the instance of the form or the form class, it has a name, it has actions and I am going to, to that in a second.

It has urls where the form should be able on and where it should redirect you.

The other thing our form fields which basically wrap the Django form fields into usable interface for this application. They are instances you can bind them to a specific form and then the, the app constructs the forms with those fields.

Talking about the actions, this is the layout of the function base idea of the, that is going the be in the next release, you are pretty much get the form model which is the instance of the form in the, from the database. You get the form that is the form, the instance of the form and you get to the current request. With that, you can actually go and create model, you can actually do and create basically any kind of interaction you want to do with the data the users submitted.

You can interact with the form instance, take the data, the users submitted and do whatever you want with it. In this case, I am sending the data to the administrators of your page, this admin settings.

There are two actions which let you send e-mails to the specifiable e-mail address, or list of e-mail addresses, the presentation to store the information in the database. Right now it is just text field that wraps, that is contains some Json and uses that. Then the next release, the Json field couple of days ago this is going to be Json field for 1.9 so you can later query on the data inside the Json field which is pretty cool. Well the action in the next release gets also the request, which allows you to access the current user in the form, so you can actually create a profile form with dynamic form fields and when and let the user modify their profile with this application. If you want to have a new field you just add it in the admin or the, yourself this field will show up in the user profile.

The other form are the form fields, these are the wrapper, you decorate them, to the application again, they have to inherit from this base dynamic form field and of course needs to read, imported at some point which basically do that in your ep config, it defines which form field it should bundle or relate to. The name how it should show up in select fields and some attributes you want to define. So the type, if ... the field that is used in the admin after all.

Well, and this is basically how it looks like. You have a form that has the name, it has two urls, the urls where you submit it to where it is shown on and submitted to. The success url define in the generic class based views. You have the actions which appear, you have which I added a few versions ago.

Different templates for different forms you will see in a second, one default template for a form, html without any (INAUDIBLE).

There is an option to allow the user to visit a previously entered data item and with that, they can well, review what they entered before. There is an option to for recipients, so if you want to send the form by mail it goes there.

These are pretty much the fields you can enter and just the admin inline, not that much magic there. The attributes you find here show up here and this is basically what it looks like, there is a form that is rendered when you access the url, there is the success page that is, that you get shown when the user, when the user gets shown when you successfully submitted something and this is actually the detail of you when you visit this enter again.

There. This is the application I have developed as part of those, this client project. The original code was I created during a train ride throughout Germany which was, which is, which was pretty much work running in production for a couple of months until I decided okay, this is something I want to release as a third party app. Might be helpful for others, yes, that is pretty much it thank you. (APPLAUSE).

NEW SPEAKER: Thank you, got time for maybe two questions or so, if anyone wants to come up? One question. One of the things if you have, when you are on dynamic forms much of the data series is common throughout all the different forms, the dynamic part is where you would like to have extra fields to add in. Would there be any way to keep your, the same tables for the static data but the common data and then use the dynamic storage to store it into extra tables somewhere. Otherwise if you store all your data in Json it is difficult to query. Particularly if it is common data.

MARKUS HOLTERMANN: I am not sure I understood. The forms inherited from a normal form class, could override the, class based view that is shown up here in the first screen shot? So you could hook up in there and define your custom form it should inherit it from.

The other thing of starting a database, since you have the instance of a user for example, in this particular case, you could actually update user item based on some investigation so you verify that the users, verify the idea and then update the, update the, in respect of the user profile item for example.

FROM THE FLOOR: Hi, how do you manage internationalisation of ... name.

MARKUS HOLTERMANN: I don't. This is pretty much mono content translation which is something I briefly mentioned in the and hash Django thing, you could probably come up with something like, with an idea of Django parlour or something?

It was never used for the cases where I used the application.

FROM THE FLOOR: Okay thank you.

RUSSELL KEITH-MAGEE: Okay, so that all hinges on some Meta programming and sort of self-reflective code, code that is reflective on itself to build the forms. Given that that is a feature that one of the reasons why we formalise the Meta interface in 1.8, how difficult that is to try and let everyone else know, how easy or difficult it is to build something that is reflective or self-reflective?

MARKUS HOLTERMANN: I don't use the Meta ...

RUSSELL KEITH-MAGEE: I know you are not using it in that case, but the same sort of tricks would be involved? So the Meta programming idea you were using?

MARKUS HOLTERMANN: So the actually the most creative stuff and I am not really proud of it honestly is the generation of those option fields here. It involves some really hacky Django admin things, if anyone else comes up with a better code, I am happy to ... the other one is straightforward I think.

NEW SPEAKER: Thank you very much.

(APPLAUSE).

DANIELE PROCIDA: We are going for a coffee break in a moment, so I will keep you briefly. We are running about 15 minutes behind our schedule, if you can come back from coffee 10 minutes earlier, not half an hour, but in 20 minutes that will help rescue some time for our lightening talks. Briefly the numbers are improving, keep them going. As soon as Thursday evening becomes equal or greater than the numbers for lunch on Friday then I will stop bothering you. If you are coming out to dinner tonight, if you have a ticket for the VFS that is at 8:00 o'clock, if you are going on the walk, you can do that if you are going the VFS, tickets for the VFS are still available from the website, need to be at the Clink by 7:30 if you are doing there, there are no more tickets available for that thanks.

(BREAK).

NEW SPEAKER: Can I introduce Aaron.

AARON BASSETT: This place feels a lot fuller when you are standing up here! I don't remember there being this many people in the hall when I was down that side. Yes I am Aaron, a freelance developer based in Glasgow been doing some sort of web development the last 15 years started in Django version 1.0 and been doing that more or less exclusively for the last 5 or 6 years, I run a small studio small is in me based out of Glasgow and as you might be able to tell I'm not actually Scottish myself I am Irish I have lived in Glasgow long enough however that my accent is somewhat muddled. I really don't envy the transcribers today sorry about this but I will try and make sure my slides give some context to what I'm trying to say so if you can't understand me you should be able to understand those. Throughout my talk I will be posting code, links, URLs, don't worry if you miss them I will tweet them after wards.

Before we get into real-time apps today I want to do a little bit of the history in real-time and how we started off and how we've progressed to what me now know is real-time applications.

Have to go back to 1996. 1996 was the introduction of internet explorer 3, it was the first web browser to have I frames, and I frames was really the first time that we could up-date a part of our HTML document without refreshing the entire page. In 1997 another Microsoft team this time the outlet web access team introduced what would go on to become HTML request object and that allowed us then to start doing real-time in a better way than what we were doing with I frames, it wasn't until 2004 with Gmail that we started to see these one page applications. But they were still using things like short polling. So short polling is where you issue an Ajax request to your server and issues lots of them so in this example we're issuing a request once every second so it's a lot of requests from our client to our server and it's a lot of wasted requests as well because we're issuing requests regardless whether the server has information or not, we've no way of knowing if there is anything on the server that's been updated since our last request, we just issued rue guest and see what's there. Then we brought in long polling and long polling was slightly different instead of having lots of very short requests we'd issue one long request with a very long time out. This example I think uses about a minute. And that request would stay open to the server until the server had some information for the browser, it would then send that information via that open connection, we'd close the connection and start a band new one over again and keep that connection open in the and that was slightly better because it meant then we weren't issuing lots and lots of requests whenever the server didn't have any new information to give us but both of these messages they're dirty they're hacks. We're trying to shoe horn in real-time functionality on top of a protocol that wasn't designed for it. In 2010 web sockets came along and web sockets it's what allows us to do effortless real-time apps. It's been specifically designed to solve this kind of problem and the major issue it overcomes if we use short polling as an example, every time we issue that request to the server we have to send our request headers and request headers are according to one of {inaudible} white papers are roughly 700 to 800 bytes. They're not huge but it's one standing on every request half second or whatever the frequency we set our short polling app that doesn't need to be sent, for the majority of these requests we don't really need to be sending these headers along with it web sockets it's slightly different, instead of full request header we have only 2 byte header with one byte that marks the beginning of the data and one byte that marks the end and that's it so it's quite a substantial reduction in the amount of header size.

I know 700 bytes doesn't seem like a lot any way but if we blow it up to web scale so say we have 100,000 users and these 100,000 users are using short polling so they're heading our server every second. And along with that request they send a header that's 871 bytes. So 871 bytes times 100,000 users per second 869 million 800 bytes per second a little over 665 megabits a second - I get bits and bytes mixed up - roughly 83 megabytes that is quite a lot of data we're sending really we have no use for, it's not containing any useful information, it's not new data pushed out to the client, it's just the same boring plate going back and for all the time. If we did the same with web sockets so had 100,000 users let's say the client has the information once a second we still have the same frequency of updates, still sending information every second, headers 2 bytes that's 1600(?) bytes but {inaudible} 0.2 meg. Immediately saving ourselves over 80 mega-bytes per second for 100,000 users so that's bandwidth not only receiving ourselves but in this day and age where we have mobile internets massively over taking {inaudible} internets and people have restrictions on their secular networks and it's costing a lot more for that band width we're saving our users and ourselves money. So to take nothing away from this talk polling is bad web sockets is good.

Imagine going to move on now to how we're going to use web sockets in Django and I'll take a couple of different approaches in this talk.

The first will be a way for you to host your own kind of real-time application of the structure and the next will be how you can use some of the same tools but hand off the actual hosting of it to a cloud based solution so you don't need to worry about the scaling and infrastructure behind it. In both instances we're going to use a thing called swamp dragon. Swamp dragon is a fairly new Django package I think the first commit was back in march last year so slightly over a year old. Don't let that put you off using it it's really everything you wanted from an open source package. It has great test suite's it has good documentation, it has easy to follow examples and the core developer and maintainer is quite ready to answer questions and feature requests to get help, yes it's slightly immature at this point but it really is a pretty complete package already.

So, what is swamp dragon? Swamp dragon is 3 things. It's Django. It's tornado. And it's Reddis. For those who know, Reddis is a key value cash or key value store. It is very similar to memcache in that it holds the data in memory so it's very, very fast but it also writes {inaudible} disk so can be used as a persistent store as well so we get the benefits of memcache but should we need to restart server or turn it off won't lose all our data and comes with pub sub publication subscription built in which is what we'll use for real-time stuff. Second part is tornado, Python based, a web server, a non-blocking - uses non-blocking U curls(?) so can use up to {inaudible} connections fairly simple and lastly Django, well if you don't know what Django is you'll be very confused the last couple of days. {Laughter}.

So how does this pull together what is our stack?

Well we've got our regular Django application and it still saves to a regular database in this case postgres. What we also have is in swamp dragon. Swamp dragon sits on top of our Django application and whenever we make modification to our data it sends the data to re dis. Whenever you develop this and you'll see examples shortly you are only working with Django, all the magic that does the real-time communication for you kind of bio directional communication via web sockets that's done via swamp dragon you don't need to worry about that, you write your Django applications more or less as you would any way and it takes care of the rest, so in front of Reddis we have our tornado server which is going to be creating our web sockets and they're going - we're going to create our subscriptions to these web sockets and browsers. We'll be talking about web browsers solely today. It doesn't mean it only works with web browsers, it can work with anything that understands web sockets, there is native libraries for IOS and androids, web sockets is just another protocol like http so really you can build apps that can subscribe to these services across a multitude of different platforms.

The easiest way to demonstrate how you build an application is to build an application so we're going to build another to do app because the world doesn't have enough of those.

To get started to do that we're going to install swamp dragon. Swamp dragon is a Python package a simple pip install. It will install most of your dependencies as well, you'll get tornado and things like that. What it won't install unfortunately is re dis. You need to install Reddis you self. You can do in apt gets brew install it's straight forward. The Reddis folks recommend you down load the {inaudible} off the Reddis site and do a stop to make sure you get an up-to-date version. The only place you'll find problems is on windows. Reddis doesn't support windows at all. If you are a windows developer then {laughter} I feel for you ... {laughter} if you are a windows developer you can run it in a virtual machine. There is also services such as Reddis to go, cloud based Reddis you can connect to instead but won't be able to install it natively on OIS.

Once we have these installed swamp dragon comes with its own dragon admin. Swamp dragon admin is very similar to Django admin and has a 38 project command. I personally don't like this approach. The 38 project will create a new project with D fault Django directory structure and I'm sure as many others here I don't use D fault Django directory structure I have a modified directory structure I have a different way of setting up my settings . I have multiple settings depending upon environments and lots of other things you'll probably - I've nicked from 2 scoops book so I don't actually use the dragon admin but looking at what dragon admin does is, well, it creates a new Django project, we can do that ourselves that's straight forward, adds a couple of settings in your settings file, well documented, and it creates this server dot py. Server dot py is a new file swamp dragon drops into the route of your project that controls the run server command for tornado, it's very similar to the managed up higher run server that comes with Django except instead of starting your Django server it will start tornado server so you need to run both. It's great for development same as the Django server. You're not going to want to put it in production. You probably want to manage your tornado server much the same may as you manage your Django application and want to put that under supervisor or use g unicorn(?) yes you can drop it in don't need to use swamp dragon to create it copy into side drop yourself but when you go into production you probably won't use it anyway.

So, we have our application all installed, we have updated our settings file. Then create models. I'm not going to put the full models up here. It's a to do app it's pretty basic. We have a to do list model which has a title and description and then we have a to do item model which has a text {inaudible} to hold is, {inaudible} and a foreign key to our list so we can group all our to do items together.

You notice it looks very much like a regular Django model, there is nothing really strange or exciting about it. We do have this swamp dragon code that we've imported here. Mixing called self-publish model. What that mixing is going to do is it's going to override the save method on our model. So whenever we create a new instance of a model, whenever we instance of our model it's going to call this new save method and what that save method will do is take the data we're interested from that model and it's going to send it to Reddis so then we can push that out to anyone who subscribes to browsers via tornado.

What we also have on our model is a serialiser. For anybody who has not come across a serialiser before they're a way to translate Python code into something that your client can understand. So, in this instance we are translating our Python objects into Json because we're sending it to a web browser, web browser can't understand Python objects so we convert it to Java script which it can understand. The serialiser is completely new it's not a modification of a Django model or anything, going to create a new py file for it saw in the see serialiser we have the model serialiser and they're kind of like our model forms in that you don't need to use a model serialiser, you tell it your function, it's there to make things easier because a lot of the time you're going to be dealing directly with models but you don't need to only serialise model - if you wanted to build an app let's say was moderate in your server so was looking at your CPE load or at the hard drive space you had left, could serialise that data and send it. So like the models form and forms class you use a model form if you just want to do the basics model or get into forms class - same as serialisers in swamp dragon you use model serialiser if you want to serialise a model or you can dive down into the kind of bare serialisers themselves and write one for any bespoke data you may have with the model serialiser we're going to tell it here is the model we want to serialise, tell them what fields we're interested in publishing so our done to mark whenever we've completed an item and text if there is in an item and we're going to find this up-date field. The up-date fields will see you later front end and that allows bio directional communication so we're telling it that in the client we want to be in up-date done status, want to be the tick to do item in our browser and have that saved back into Django as well.

For anybody who has used Django rest framework in the past they'll probably be very familiar with serialisers. DRF uses something very, very similar. It's a bit annoying how similar they are because they can't be used for the same purpose so you do end up with a bit of code duplication. There is however a branch currently on swamp dragon that actually adds support for DRF style serialisers so hopefully in the very near future you'll write one set of serialisers and that will {inaudible} rest PR I via rest framework and also {inaudible} via notifications via swamp dragon.

As well as serialisers we need to create roots. Roots are very similar to views in Django. So the fact we have our object which is going to return an instance single instance of an object and we have this get query set which allows us to override the get query set method and return like our own filtered query set. In this instance whenever I am looking at a list I obviously only want to do items that are part of that list so I am overriding query set and go OK here is the list ID only give me items that correspond to that. There is a bunch of built in verbs we have our get list, also have get list, delete, create, subscribe, unsubscribe, these are similar to the allowed http methods in our class issues. In much the same way you can define your own methods as well, you're not stuck using these verbs you can define your own and subscribe to them in the front end.

When we do subscriptions where you give it a new name so in this instance it's called to do I am. You'll hear me referring to roots also referring to channels depending on the software you are using almost interchangeable. Some will refer to them as roots some refer to them as channels. I'll use both in this talk so apologies if it gets confusing but in this instance they call them roots.

On the front ends a very basic kind of to do app front end. Use this thing called twitter boot strap, anyone heard of it? I don't know I never see it anywhere. On our template here we have our list title and list subscription then have some to do items 4 of which are marked in green as done one of which is in red still to do.

Now to make this actually function, we need to link it up to our Django code. I'm going to use angular. I know angular is no longer the hot new thing. I should have done it in react is that right is that the one that is cool this week? {Laughter} doesn't really matter which one you use to be honest or which one you prefer. Swamp dragon is not tied to any particular framework. It does come with an angular service that's why I use it because it's easier for me. If you wanted to use react or backbone you are free to do, also comes with Java script service if you are retro like that so don't worry if you don't write angular or you prefer something else or your front end team prefers something else. I'm using this for ease at the moment. It's not a deal breaker. You can use what you like with swamp dragon I'm not going to go {inaudible} that's really boring just the bit that matters and that's this small snippet that is going to control that template we saw.

So we have our to do list name, description then have this loop that's going to run through our list of to do items and it's going to put them on to the page as well.

We are wrapping it in these verbatim tags, the reason for that is angular, also uses the double braces syntax for variable names that Django uses, without them Django would try and replace our angular variables then it would disappear, we don't want that to happen, so we have wrapped it up.

Okay, I know it is way too small for people to see, we will go through it. Subscribe to your channel, subscribing to the channel is how we specify that we want to receive any updates on this route. So in this case, the to do. We are only really interested in to do items that belong to the first list, I have hard coded the list of one in here, if you are going the do this real, you wouldn't hard code the ID you would have some way for users to select which list you are interested in. Also worth noting that the query syntax for swamp dragon is similar to Djangos, pre-set filters syntax, the reason the double one to specify a property on a foreign key.

We have our get single, these are going to run when the page first loads, this is to make sure when you first arrive at the page it is not empty and blank. We get our initial to do list and items. Only interested in the first list, get single an ID of one., then get list, all the items in the database list and pop on the page for the first time. The list idea overflow clone one, but wouldn't do that in real life. Etc. etc.

So quick one back. So quick run back, subscribe

Get a list of our to do items, really thought better of naming these lists, pre-set lists and Python lists, when you call lists, the to do item lists, that will populate our templates.

Okay so. This is a bit of real-time where it comes in. We have the on channel message it is going the get called any time the server has new information for our clients, anymore new information for the browser, call this on channel message funs.

So we can check if it is the channel we are interest. ... The to do items, we apply the data mapper, look to see is this a new item, there is an existing item already with the ID, if there is, let's add that one, we don't want duplicate to do items.

Then I am nowhere brave enough to do a real-time demo on conference Wi-Fi, we will have a video, this is a real application, I do have it on my lap top. I can prove it does work, I am more than happy to do that away from the pressures that is the stage. Come and see me.

Here we have a chrome window, a fire fox window and a safari, then beneath the Django admin. As I go through, I am updating stuff in the Django admin, changing the status to done, immediately changes in all three browsers, normally it will change in the browsers before my Django admin page even finishes refreshing, obviously going the be running quick because it is running on my local machine, but when you are working on it remotely, because the connection is always open and ready to have information pushed down, almost as quick when you are working with remote servers as well. Notice I am able to update stuff in the actual browser and that is then sending that information back to Django, so it is bidirectional, not that we are pushing out from Django, but the clients can also send information back. We won't see the updates automatically in Django, in the same way, but it is saving it back, so if I refreshed that admin screen you will have seen all the changes I made were reflected in the admin screen.

All the code for this is on, have a look, play around, put it to react, backbone or whatever you want., if you do mess with your urls's, don't worry, I will tweak them.

So in the first example we were introducing additional infrastructure, what is your normal Django stack, I don't know about anybody else, real-time, I didn't have tornado running, I had install just for swamp dragon, so it is adding additional complexity and more things to scale. I have ... I don't want to have to manage anything else.

I have much better things to be doing.

So we look at the platforms of services. There is a couple of different ones out there, fire base, we have pub nub, we have pusher, some of the sponsors to have day. There is a few database as a service out there, there is not a huge difference between people doing pub nub and database as a service, most concerned with data sink, making sure you have lots of read reply cars and the ability to sync across platforms, the other ones pub nub and pusher and ones you mentioned earlier are more interested in the pub sub part of it.

There is an interesting aside, when researching the different services out there. One of the ones is called data Fly, I don't know how many work in corporates or enterprise -- good luck putting a purchase order through for data Fly.

Phil one of the pusher guys you may have seen earlier, done a really good Blog post, fairly non-biased considering he works for pusher on getting a. ... , don't take my word for which platform to use, read that, that should help you make your decision.

We will be using pusher today, it is pt1 I am most familiar with, it is kind of good for me up here because they do have this debug console, makes it better.

So how does pusher fit into our application stack that we had earlier?

So we are going the get rid of Reddis, going to keep pusher, we are going to keep swamp dragon but reduce the amount we use. The idea is we don't want to have to rewrite the application that we started with earlier, we wanted to keep as much of the swamp dragon stuff we can, swap out the data store, don't want to have to start off in swamp dragon and then hit scale and put in one of the other services and rewrite the application. Nobody wants to do that. Try to look at how we can keep much of the same but swap out what is it is actually doing the pub sub.

So now we are in a, ... how do we publish our information? Pusher has got a Python library, takes a pusher.trigger, send to channel. This is what I was talking of earlier, now we are switching to using channel to confuse everybody. We have a channel name and event, that is very much like the verbs we had earlier, so things like our updated created, deleted and then we have a payload, serialised model. Yes, that is my API key and secret key, yes, I did revoke them before I stepped on stage, better luck next time (LAUGHTER).

Okay, so how can we end up inviting pusher and swamp dragon, it sends our data to Reddis is our self-published model. That mix in at the start to be included in the models. We are going the rewrite the self-published model. So, we are going to keep our serialisers the same, they will still be the swamp dragon, we don't need to modify them in any way, but instead of sending the information to Reddis and then on to tornado, send it to pusher. Here you can see I have just used the same pusher code we saw a couple of slides back, sending it there rather than to Reddis, this is the way, I wrote both the applications, the swamp dragon first and then challenged to do the pusher application using few edits than I can.

This is the get dif. Most of it was deleting stuff. I didn't need the server.py. I no locker needed the, no longer needed the routers, that is by pusher, I added some code to do the self-published model that we saw a sect ago. Again using angular, I don't want to change, I am trying not to change as much code as I can., pusher has an angular library, it worked out well.

So this is our new angular code, I have had to make modifications here, because the pusher angular service doesn't map one to one with the swamp dragon one, we are now creating a pusher client, but we are still subscribing in much the same way. Subscribe to do item channel, then written a bit of code, updated to that same list we saw.

If I was to take this further, which is probably little bit out of scope of this, it would be java script heavy -- we would look at the angular service and ensures it has a one to one map with the, the swamp dragon and our pusher angular service and one to one map. The data we are saving through and the methods are the same. Then we can code the exact same.

Okay, another demo, so this time it actually isn't always going the be running on my local machine, sending information to the server and then going the be seeing that come back to the browser window, so I have the two browser windows on the left, then the machine, safari, I think fire fox, then the browser in the right is the debug console I mentioned briefly. The one on the right is running on pusher servers, the two on the left are my local machine, the admin is on my local machine, that is where I create the information and send it up.

As I create in the Django admin, you will see that on the pusher debug, you also roughly the same time see it in the two left hand browsers so you see it is being pushed back down the local machine.

That will work for all the same things as we had earlier, I will be able to create new to do items and edit, I will be able to delete to do items.

Code for that is also up on get hub, have a look, have a look at the dif, make sure I wasn't lying to you and give me any feedback you have got.

So the final bit I want to talk about, so far we have been talking about how you serialise model data, for a lot of the modification stuff, you might not want to send the whole model. This is a logo I am working with the council up in Glasgow, a data portal, a way for the council to open up the data they currently have and make it available to everybody, it is a pretty good product. Data from everything from bicycle rack locations through to school truancy rates and things like congestion levels and pollution and all the other stuff that should be able to us as, as like, members of that society or of that council area.

When I am working on the application, there is a lot of the data is automatically harvested from coup sill systems, a lot of the data can't be, the council systems are not exactly up to date. Not a lot of API's, available., it has to be manually entered a lot of the time by a team of data entry people. We found sometimes if you have got more than one person working on the same file or data resource, we can run into issues, we have our user one, two opens the source, user one saves the changes, user one changes vanish, that was the end goal to have some kind of collaborative editing, like Google docks style, but this is government work we don't always have the biggest budgets. Also it is not a problem that really occurred that often because we don't want to spend the developer time or the budget on it. A service called help Skype has a unique way of dealing with this. An icon that goes blue, red, it is telling you that somebody else accessed the thread and somebody is reading it and replying to it. That is all we needed. Needed to know when somebody else was working on the data set. Three desks over tell them, please don't do that. The easiest way to drop in the pusher code we saw earlier into our view, have it send an event r vent and the user name so we knew who was using it. Not really pythonic or dry, I have put a mix in, the response, this is published in a, about a week ago, some really good feedback on it. One of the things we are discussing, where is the best place to put the mix. If it is updating something, should it override? There is a big discussion I would love peoples feedback.

We are only really interested in updates about a particular instance of a model. We don't want to know when somebody is viewing any model. We want to know when they are viewing that instance. To achieve that per instances. So each channel gets a unique name to subscribe to in the front end, that is the model name plus the primary came. The data is the person's name, who else is editing this? Doesn't necessarily need to be something basic as that, we could use serialisers we saw previously, there is no reason not to. This could be any kind of data, using the same publishing methods that we did before.

On the front end, I haven't used angular, because it is small. Created a pusher client instance with an API key, subscribing the channel, the objects, this will only work on pages such as detailed views or update views that we have the object in the current context.

Yes, you can see I am kind of specified that there. I don't have a fancy video in action, it is still in development at work. I have wrapped this up in a Python package that is installable now from py.py. It is a few mix ins from your, creates update, delete, also the java script code we saw there, is all wrapped together in a nice template tags so really simple to use and also up on get hub. So this is a pull request going on, fortunate one of the kind of core developers of braces has been giving us some great feedback, I don't know anybody better to give feedback on mixings than anyone involved in Django braces, I am pleased to see that.

It is on get hub. I would love people to get involved in the conversation around that and the arguments where we should override the different types of events links there.

So, that is it really to be honest. Any questions?

(APPLAUSE).

FROM THE FLOOR: How do you handle log in authentication and access control to the data especially when you are using a service like pusher?

AARON BASSETT: Probably not the best, those guys, the swamp dragon stuff does have things for authentication as well, you can authenticate users before they subscribe to it. With the pusher I know they do have private channels which can also be encrypted as well, I don't know the ins and outs of that I am afraid.

FROM THE FLOOR: So when you push back the update from the client so when you click down on the to do item, it goes back query {inaudible} direction into the Reddis {inaudible} and from there reaches Django how?

AARON BASSETT: So although it's coming in via tornado then into Reddis we still have that swamp dragon layer and that's getting from tornado and that's back saving into your Django model.

NEW SPEAKER: So swamp dragon basically runs a listener thread where it listens for incoming connections from Reddis?

AARON BASSETT: Swamp dragon is integrated into tornado, that's why you see the run server for tornado in swamp dragon so it actually handles that side of it for you as well, as long as you are using the swamp dragon way of getting tornado up and running then it should handle that as well.

NEW SPEAKER: OK thanks.

NEW SPEAKER: That's great, can we thank Aaron again. {Applause}.

I think we're back on time now so here is Loek Van Gent from the one per cent club.

LOEK VAN GENT: We'll be talking about some of the things you heard earlier today just in Aaron's talk and David's talk but hopefully I can be of service to the ones that were thinking that's really cool stuff, but that's not for me. Hopefully I can convince you that part of things they were talking about can apply to any of you. I don't know, I reckon most of you would be back end developers. Oh, a first. Are you still awake? Who is asleep? Hi, quite some, who is awake, 2 hands yeah that's better!

The last talk but we should get pumped up for the lightening talks after right?

So, I'm going to be talking about and the title is true beauty is on the inside but your users are shallow. That should be a familiar - should sound familiar.

I want to tell you something about writing about front end code. So first about me this is me away from key board, I have done PHP for many, many years before as have others of you I think so too. 3 years I been riding ponies. Proper multi-Dutch Django association. If you are not a member yet and are from the Netherlands they just lowered the annual membership fee by 10 euros and it's free now so you should sign up. {Laughter}.

And I work for 1 per cent club. 1 per cent club has been remembered several times yesterday. We're not the richest people in America. We do run a platform for small projects that have a social impact. So far we're 800 - close to 900 successful projects, very small, couple of a thousand euros just to start for instance to buy solar lights for a village in Nigeria so kids can study when the sun goes down and very small, smart changes that have huge impact.

So, we're doing quite well. We are changing lots of lives, and doing projects with social impact that's something that also big corporates are interested in doing now and mainly as a corporate social responsibility programme, so now we're selling our web-site to them too and then we can earn some money because from do good crowd filling that we do you can't take money from that.

And so one day our product manager came and say, listen up guys, we've got a really cool idea, let's make a multi-tenant and a white label web-site out of the web-site we have. At that point, we were running our own web-site and not a web-site for a big company. And they said 2 web-sites should merge into one and make a multi-tenant. Interesting stuff of course. I won't tell too much about it but it was a big refactor we had to do and I only want to show the changes we did - oh no - it's too many changes to talk about it right now. But I can show you - you can imagine we went from multiple databases to one database with different schemas in it and it worked all very well. This is what overview with the project looked like. Then we changed it to look like this. Minor changes but the good thing is everything worked, people could still do their nations and do wall posts on the project site and even the project manager was really impressed and in big swirling letters he said wow this amazing guys we really love that new design! Yeah,, {laughter} and that was only 2 per cent of the time we spent on that. So true beauty is on the inside but your product manager is shallow {laughter} sounds familiar?

So what's all this fuzz about front end?

We have to go a little back in time. Second reference to that movie I think. OK.

Web-sites something like this. This is the first web-site. You all should recognise that right?

Anyone could put a year on this?

NEW SPEAKER: 1996, 7 ... {applause}.

LOEK VAN GENT: That's almost good. This is when the screen shot was taken. Before that they didn't take a screen shot it's from 1989 but good enough.

So this is what web-site used to look like and this is how mobile phones what mobile phones looked like.

And of course, mobile phones have taken a rise and we're not texting like this anymore we're swiping away and playing games and using the apps and in fact phone cells the interface to mobile apps are so - come so naturally, that they are more easy to use than most web-sites, they're intuitive, they know what the user wants.

And web-sites are starting to lag behind, the traditional web-sites. Of course you have things like Gmail. Anyone has this mail in their in box by Baptiste that you really should start filling in your dates for the menus that you want to have? ... Of course you can't call this a web-site any more. It's an application. So, that is clear.

Web-sites, the front end for a web-site is the user facing part of your web-site is very important, but how can we write better front end codes?

Now it's not a live demo but it's a console screen.

Let's create a Django project. We can use Django admin for that. Probably everyone did at once when they did the manual or the tutorial and after that narrow down but let's try it. We're going to create a project named true beauty. We're going to add an app called heart and a app called spirit.

Let's look at the project layout.

You'll have the main project and your 2 apps down; that's all good. The templates and the front end doesn't have those bits yet but normally you would place those inside the heart and in the spirit.

Because they're portable and you can take them out and copy and paste them somewhere else, that's traditional may how we Django go noughts think but let's ask Danny and Audrey what they think about it and they say you have to really group your front end code, at least I have to look it up in their newest 2 groups book, 2 scoops has been mentioned before and it has been said you should own that book so that's all clear. So they say group your front end code, group it in a typical project Django project layout. You should put your templates and your static files, your Java scripts inside your main project folder.

This is where you are. This bit is for the front end guys. That's it right. But I don't know, how many do full stack or - full stack? Most of you right? So, if you have a bigger company then you would hand that over to the front end guys and when we first did this in our company to start a grouping, our front end guys were really happy because they turned crazy every time we did changes in our template and had to {inaudible} all their app folders.

Let's look at how we can further improve that little bit that's really the main front end.

I'm going to just shout out some tools. Hopefully most of you are already using them or similar one we can discuss about it later if these are the best ones or the other.

So, take Bower, it's very useful to do your requirements instead of having just a base HML where you specify all the Java script you want to include like J query and J query UI, you have dependencies like you have in your Django folder with your requirements file.

If you've ever tried to do CSS coming from the back end you would, yeah, it would still be spaghetti code, you switch to Django from PHP and really happy, but with something like sas or las you can really structure your code that would in the end generate CSS and you would use something like Grunt to run those tasks to create your sas file or minify some templates and you can use NPM tool to install all that so if you do deploy NPM will ensure everyone is in power and run Bower and suddenly everything is magically done.

When I first started with projects, running Django I would just include my J query in my repository which is a bad thing, I hope you all agree. Yes so you have all these tools to get your front end in shape but it's starting to look like an entire app in your Django project right?

Now you're getting it!

So you have this back end and this front end. The back end is probably has to deal with some file storage and the database and there is probably a user talking to your front end hopefully. You need some back and forth communication.

Django server can talk, can perfectly have that data, talk to your front end using Django Rest Framework in Json, that has been - it was in error sock as well of course. In front end you'll use full tack framework like Ember or Angular or React.

The good thing about splitting those is if you have an API in your server, if your server is just talking API, then you can easily test that and have an integration test. Of course you can have your unit test on a low level, but you can start writing your back end and you don't have to wait for all those front end as they are tweaking their pixels and can just run off if the functional design is clear, you know what the API should be able to spew out and accept back then you can easily write your test and your back end is done.

And the other way round is the same. You can start creating your front end, you just use a mock API, and you don't have to wait for all those back enders over complicating things. You can easily run a demo if they suggest new designs, it's very easy to just change the styles, and you can even deploy the front end separately from the back end unless you have API schema changes of course.

So, what we use it's not our hit thing Aaron was trying to think of. We use Ember CLI, for our front end, it's just an example how you can use it. And I'll leave enough time so we can argue if Ember or Angular is better.

Back to your console. All you do is ember - no we want to create a new application. It's called good looks because it's on the outside. Let's go into good looks. We generate the route that will - Ember starts with the route and then your MPC(?) pattern backing that up and create a smile, that's the 2 apps to talk in Django terms and it would look something like this. You have your app JS that will spin up your ember server and then all the components that will give you the eyes and the smile and will make your user happy.

If you want to really go wild, you can also use ember serve that will spin up a web server and you can proxy your own API after that and then you - then the API calls will be proxy to your Django server. And then you have a fully independent front end just like a web app, just like a mobile app, but backed up by your API.

So, summing it up in a few easy steps to hopefully give you a hand to changing your full stack Django web-site into 2 things back in the front end, first thing is love your front end. It's the only thing you use they get to see. You don't see the marvellous thing you've done on the back end and the marvellous ways you manage your database and stuff like that. Group your front end, if you are the only container, Kat where are you? Then still it is good to have it in clustered in one place.

Use tools to help structure your front end code, if you are not using barrow or NPN or something like that, try it, the same goes for, for instance Ember start using it, try a demo organisers use Angular or React, but forget about Backbone, it used to be popular among the Django developers but that is not the way to go to build it all yourselves.

So that is all from me.

If there is any questions or suggestions please, be welcome.

(APPLAUSE).

FROM THE FLOOR: Thank you for your talk, that was a really good talk. I did look into Bower and Grunt and all those lovely looking tools, all my colleagues, I was on a didn't have the internet and the project I am on, the client didn't have the internet on the network we are going to deliver on. Any advice for trying to use the tools without access to the repositories they will pull from?

LOEK VAN GENT: So you are talking about an intranet.

FROM THE FLOOR: Essentially.

LOEK VAN GENT: I would reckon you could do a stage deploy, so first, make sure step one, build it as you would in a normal life in the world website and then deploy that. But yes, I am not sure.

FROM THE FLOOR: Okay thanks.

FROM THE FLOOR: Hi there, you mentioned about multi-tenanting bit, what package did you use and what problems did you have with it?

LOEK VAN GENT: Were you we were, we were using Django tenant schema.

FROM THE FLOOR: I patched that,.

LOEK VAN GENT: We did the same thing, extended it with useful stuff. I did send one pull request that I didn't get on.

FROM THE FLOOR: Maybe I will have a chat with you, thanks.

FROM THE FLOOR: Thank you very much. I would like to know if you did try to implement some life with load, so every time you changed your SASS files or java script files automatically the ...

LOEK VAN GENT: Ember does ... I don't know if you mean that.

FROM THE FLOOR: You don't use a grant for that?

LOEK VAN GENT: No, so if you use embassy live, use that as the web server for the front end, that fires up live reloads if you are coding it will reload like the Django server run server can do right? Is that what you were looking for?

FROM THE FLOOR: Yes, the way to implement it, Django server with this life will load methodology.

LOEK VAN GENT: Ember will do the same and I think Angular has started shipping a CLI as well I think.

FROM THE FLOOR: Thanks.

FROM THE FLOOR: Hello. What would you recommend if I would, were to, if I would like to recycle one of the apps in another project? With front end and all?

LOEK VAN GENT: Sorry?

FROM THE FLOOR: If I have Django app and the java and I would like to use it in another project, how would you do that if you can't group it all into one project?

LOEK VAN GENT: Yes I think it shouldn't be in your projects from the start. It should be separated out but I think the question you want to ask is, you have this lovely app that is we all use, all these libraries and they do also have templates in them right? And if you split it yes, then you have this, that Django template stuff going on and you have to rewrite that in your Ember or Angular application right? Yes. I don't know.

I understand the problem. But, yes, the main thing is, how, how can we continue using Django in, now I think we all sense this is happening that we, that web servers will be mainly to pride data and that representation will happen in the browsers. But, yes, I am, I don't know. So you can really use your app up and until the API so the API framework. I think.

FROM THE FLOOR: Thank you.

FROM THE FLOOR: It is going to be really quick, I just want to answer the question that has just been asked.

If Ember is your plus Django is your regular stack, ember is component if you want to do an app in Django, you can usually map it to your component in Ember, when you want to have your template with your reusable app, you can pack it as a component and put it in there.

LOEK VAN GENT: True.

DANIELE PROCIDA: Thank you very much Loek.

(APPLAUSE).

Okay. We are in the very last session of the day. Our lightening talks so if you signed up yesterday and weren't on the list, make your way to the front. If you signed up today, form a queue behind them.

Briefly, Djangocon Europe 2016 will be held in Europe, but we don't know (LAUGHTER) (APPLAUSE).

Other than that, we don't yet know where! so we are inviting proposals and suggestions. If you think that you might be able to put together a team in a suitable venue, in a suitable town to host this conference next year, come and talk to us. We will announce this very soon. We have already got some expressions of interest but, we would like to know more. What's more, if it is, if you feel it is something you would like to do but perhaps aren't ready to do it in a year's time, now is also the time to come forward. Because we would like those people who are putting forward serious proposals for a Djangocon Europe in 2017 to be involved with the organising committee of Djangocon committee 2016, so there can be an exchange of knowledge and information and so on. If you would like to be involved in organising it. The time to speak is now, even if you are not going the do it next year.

Peter Finch is in the house, ready to go on a Cardiff walk if you haven't been. Put your hand up if you went yesterday? Do you recommend it? Yes. Even though it was absolutely horrible with rain, everyone who went on it thoroughly recommended it. If you are booked on, you will need to meet Peter on the steps of the museum at 6:30 sharply. We have got some of Peter’s books for sale on the registration desk, if you have bought one, I am sure you can get him to sign it if you ask him nicely.

Vegetarian studio, be there by, if you have got a ticket for the clink, please be there by 7:30. Let's go on to lightening talks. (APPLAUSE).

We have Daniel. Did you do a lightening talk yesterday? Yes, you did. So who is next?

Django and robotic telescopes, Edward thank you.

Plug yourself in and off you go. Got the timer?

So, in case you weren't aware, there is only rule one of lightening talk club, your talks last no longer than five minutes when we get towards the end I might give you a signal and you can start clapping like so, just to let people know and when I give you another signal. Go into a round of applause when their time is up.

NEW SPEAKER: Well normally when I stand in front of a group of strangers, either astronomers or school kids, I am daunted by standing in front of an impressive group of brains, oh what has happened there? I keynote quit. I am not impressed with that, slightly more daunted. My computer is doing weird stuff. Hang on a sec. What I was going the talk to you about, something fascinating about astronomy. Everybody loves doing this sort of thing on the fly, I will try and get what was my presentation back.

I would sing to you, I think that would be more terrifying than what I am about to say.

These Macs are rubbish!

(APPLAUSE) (LAUGHTER).

Okay now got a minute less than when I started so, this is a beautiful picture of a galaxy, who likes space? Oh wow, I thought IE seven going to have to walk off the stage, that is a galaxy, a hundred billion stars, the greedy star on the right is eating the one on the left. This is something in our own galaxy, this is stars being born. This is an asteroid that I observed. This is zooming through space, this is a galaxy with an exploding star in it that was brighter than the whole galaxy, so these hopefully have wet your appetite. If you were one of the few people that didn't like astronomy and space before, hopefully that has made you like it now.

I work for an observatory, we have a robotic telescope that spans the globe, unlike the British Empire we are not evil (LAUGHTER) and, the sun never rises on our network, see what I did there? We have 12 currently operating telescopes, we have a big one like this, the size of a double decker bus or if you are not from Britain, the size of two buses on top of each other.

Telescope, 1 meter, the size of a family car. We have telescopes that are 40 centimetre class ones, about the size of R two D two maybe you are too clever for that one.

We cluster them together at the sitings, we have more than a thousand active users about 250 of those professional astronomers and about 750 school kids. Now, how do we accept all the requests and package them together, we don't let people look at individual telescopes or let alone to fly to them. They go to a website and they make requests. There is a cyber-robot, it is a bit like a giant Tetris programme in multi-dimensions, there is the most optimised observing strategy, we use Django from almost everything, from the submission interface, to the request handling, the observation request handling and storage, a jerobi solver to package them tonight. The schedule that produces is stored in a database and we can inspect it with Django and the telescope is in java, boo.

Once your observation has been done, the image data pipeline is done in weird stuff. The archiving is done with Python and then we can inspect that with an open access archive through Django and an observation request interface which is also Django, so the reason for my talk is, we need you guys. I am a little bit daunted because I sort of had a crazy idea the other day, you could do more than just support us through amazing and awesome things with Django's and Django's core at the third party apps. Maybe try it out for yourself, normally we only let professionals and school kids use our network, but you could also try it out or, if you want to try it out and take it a bit further also work for us.

After Russell's talk said he is a doctor, there is me in the Tardis saying I am the "doctor."

DANIELE PROCIDA: ... following from, Craig following by Carlos please.

NEW SPEAKER: Craig: How do I get this to?

DANIELE PROCIDA: After Carlos we have, Thomas. Then Matthew.

CRAIG: Does this work? Hello?

[changing resolution].

I can't get the slides working, I spent an hour, if you want to see them, find me later on.

My name is Craig stone, you can find me on twitter. Since December I have been trying to mash Django with angular JS together to try and create a real-time web thing. So, you can get, so I am guessing how many people here know about angular Jess, everyone knows Django? An excellent back end from what I found and most of this mashing together the two frameworks is because my team leader said we should be doing this, I have been doing that.

I have been using Django fresh ... an app called Django jangular, I had to rename them annoyingly.

So if you can use Django rest framework to serialise your objects, set that at the front end as a Json, then been using Django angular to allow remote, you can decorate a method on your view with allow, remote underscore, invocation, what that will let you do, that will let you call your Python from the angular JS so if you have some Python code which you don't want to rewrite in angular or java script, or have a long going process which you can fire off, you can access it that way. Simple if you can, if I can show you the docks later on, talks you through it.

So that is what I use Django angular for.

The other package I have been using jangular, that lets me access the settings from java script, so I can point at the static urls, when I ... I don't have to rewrite stuff.

There is some other cool things that I have not developed yet with the packages but there is something called I think swamp dragon does slightly better but not looked at that, three way binding with Reddis. Input something on the page and type out for you as well. But also, there is way of using Django angular, to pipe it to Reddis and to pipe it to your database directly.

Also some edits to manage.py which one of the packages does, rather than have star app, star angular app, that gives you a ..., based on angular seed. I also found using Django angular and jangular as a good way of learning.

There are some issues, angular Jess uses curly braces, use the verbatim type you saw earlier, I let Django have its time with my index view, and passing it to jangular JS to fill everything else in. Both do urls routing, seems a waste to have two doing differently and you can't use both.

As nice as it is, I had to learn a whole new language, so, so I suppose that is a plus and a minus. Am I in time? Do I have time for questions?

Oh. I can see a time limit. One minute of questions then.

No? Okay. (APPLAUSE).

Okay, Carlos, API by example?

CARLOS: ...

All right. So, this is API by example. It was, it used to be!

It is no longer. Here it is again. So API by example this is an approach we took to testing across different services. So, this is what we had at when we started we had classic Django application it was a monolithic thing, what we wanted to get to, to more split up into smaller services in the back end we wanted to use Django and a API server, the ... this is the kind of simplified view of part of the system and so the, we had two teams a front end team and a back end team and we basically wanted both teams to be able to work independently, we wanted the UI team not to be blocked by the back end team. We wanted to reduce the dependencies between both teams and wanted both teams to be able to test their part separately and we wanted to avoid the fact that the, in the past, the back end team was dictating and the front end had to adapt to it after.

So we developed a way to specify the API we wanted to make it a bit formalised. We defined API by example. Specified with a sample question and responses, structured into the files, ABE files. This is what the files look like, you have the description bit at the beginning, more, more humans, so it is meant for a machine human consumption, there is a part that describes exchanges of requests and responses, you can have as many examples of this as you want. Structured repository, we had .. and mocks and back end application.

So the, the important thing to establish, began around the discussion around API so this discussion was formalised as a pull request for the repository, this pull request had to be signed up at least one person in back end and front end, we wanted to know it was usable by the people.

In the end, the PR discussion became a documented discussion over the API process and ABE file became documented not just a documented API but with some tooling around we could use it for testing. Whenever a this was, PR was merged it could be labelled as a API ... (INAUDIBLE) easy to install it as a Python dependency... after the new API release, took two PR's to produce the, -- this on the front end and back end team could work on implementing the stuff.

We built tooling, the whole idea around this was not just (INAUDIBLE) of the API but to make it easy to ensure that when we are testing separately, then we were testing the same thing, so we built something around this. The tooling for angular testing was at the AV protector tool, generates a back end with these responses and requesting responses and also built a small ... worked in Python units or integration tests.

If you think this idea to apply to you and it is something you could do with, it is still much a work in progress, we would be happy to hear of others problems, to come up with something that we would love to be like widespread around the, the community and we would love for the tooling system to grow and to be much more useful for the documentation to be better, so you can come, find me, I will try and make a shirt that is more identifiable, so you can easily remember me. Contact me over channels as well. We have an organisation and we have example and we have several there with other tools we have. Thank you. (APPLAUSE)