Speaker 1: Confirm this.

Speaker 2: Later or to say it now.

Speaker 1: No I need to confirm it before I record.

Speaker 2: Yes, you can record.

Speaker 1: Alright okay so let's start so maybe in your own words could you please describe what Spring Framework does?

Speaker 2: Spring Framework is an enterprise Java framework so it facilitates basically… it historically started out as being an alternative to enterprise Java Beans more than ten years ago so two thousand threeish two thousand fourish and then became basically a framework to make Java Enterprise [Indiscernible: 00:43] making that easier the traditional Java [Indiscernible: 00:43] APIs leave something to be desired in some respect there and we tried to make it easier.

Speaker 1: And so can you maybe tell me what the original design process was so I know you guys have a lot of modules [Indiscernible: 1:02]

Speaker 2: The original design was very simple I mean the project started as [Indiscernible: 1:07] published in a book right I'm sure you've heard it but for the sake of [Indiscernible: 1:11] so that book I can't even remember the title Rob Johnson wrote it and in it he described the framework the interface 21 framework and people said that's very good why don't you publish that as open source and that's basically what happened and that's turned into Spring and so there was I mean [Indiscernible: 1:34] we don't have a real design so I mean we don't think of new models we just see if there's something needed if we see something [Indiscernible: 1:43] we see that coming up on mobile we see that can help and then well we better should do that or something we should do something there otherwise well people will ask us too right maybe we are not I think we are a bit reluctant in that case because we don't we are a bit reluctant [Indiscernible: 2:04]

Speaker 1: Okay so how is this decision really taken so I know you are one of probably the more original developers in Spring Framework but was there like a core team involved did you guys actually involve community did you involve companies you said Java enterprise but I see it as more open source framework.

Speaker 2: Initially it started very open source [Indiscernible: 2:27] because there was no company so that's when the book came out and everything so that was people working and then quite quickly these people figured out that there was something here and a bunch of them about 8 of them I believe said okay there's something here we can make money out of this so let's start a company around it and this company was called Interface 21 [Indiscernible: 2:53] so this was around two thousand fivish and gradually what happened is that we started hiring [Indiscernible: 3:09] so in the terms of the company the community became the company in a way. Then all things happened right along the road and in thousand and nine we were bought by VMware and as a consequence of that a lot of those people… original founders left because at that point they were probably rich so they don't really need to work anymore and but a lot of people [Indiscernible: 3:39] I mean I think we are still… I am still there I've been there for ten years now and other people from my time [Indiscernible: 3:49] original founders if you see they are

quite a lot when it was being bought by a bigger company you know there was entrepreneurs [Indiscernible: 4:01] I hope that answers your question what was your question again?

Speaker 1: No, more about the original designs. [Indiscernible: 4:08]

Speaker 2: No there wasn't an original design we just…

Speaker 1: [Indiscernible: 4:11]

Speaker 2: Well so I mean there was no big plan we are going to do this [Indiscernible: 4:18] with JDBC even with JDBC [Indiscernible: 4:20] let's see if we can [Indiscernible: 4:22] better let's maybe if you want to call that a design [Indiscernible: 4:26] there was no very pragmatic way of looking at it saying I want to do this doing this currently sucks let's leave it and do better that's a very traditional open source [Indiscernible: 4:38] so it's basically…

Speaker 1: And if I can ask you I know the answer to this but how well is the API documented?

Speaker 2: Quite well I mean we do Java on all public methods basically and even [Inaudible: 4:56] sometimes we even have internal design documents I mean I am still wondering what you mean by design.

Speaker 1: So it's more the design, the architecture.

Speaker 2: Right.

Speaker 1: So how did you decide you know that you want to go this route.

Speaker 2: Right yeah [Indiscernible: 5:22] in most cases just trying to be a layer on top of existing stuff just making the existing stuff more convenient to use so [Inaudible: 5:36] very limited design.

Speaker 1: Okay alright now let's get to the more interesting part perhaps which is how do you guys really decide to change the API I mean I am sure the API changes all the time the interface that is exposed I mean does it change first off because [Indiscernible: 5:54] change quite a bit and… Spring four as well.

Speaker 2: It doesn't really. [Indiscernible: 5:58]

Speaker 1: No problem.

Speaker 2: So changing is [Indiscernible: 6:09] always add stuff because that's [Indiscernible: 6:14] in terms of backwards compatibility you are probably talking about stuff like moving stuff or changing stuff.

Speaker 1: Yeah.

Speaker 2: We do remove stuff we deprecate stuff that we no longer think is useful.

Speaker 1: So what's your deprecation policy so do you like deprecate it first.

Speaker 2: We deprecate it when we have a point where we see it's not useful and then we [Indiscernible: 6:30] major version.

Speaker 1: How do you see it's not useful?

Speaker 2: Because we have [Indiscernible: 6:37] better which [Indiscernible: 6:39] both for instance.

Speaker 1: Is it only for both instances or could it also be better for [Indiscernible: 6:44]

Speaker 1: Okay so can you help me with an example.

Speaker 2: I can talk about an example that involves [Indiscernible: 6:52] and then I found a better way to do that involving a building pattern than a factory pattern so I added the builder version and I deprecated the factory version pointing to the builder version so it's just a better way of doing it for maintenance [Indiscernible: 7:16]

Speaker 1: Did this change the signature.

Speaker 2: It was mostly about things that people couldn't do in a [Inaudible: 7:22] and they wanted to do and the only answer [Indiscernible: 7:25] override this but there wasn't really yeah people asking stuff that you couldn't accomplish with these [Indiscernible: 7:34] so that's why it needed to change versions.

Speaker 1: But did it facilitate in changing the interface or [Indiscernible: 7:42]

Speaker 2: It was pretty much the same but it was different results [Indiscernible: 7:47] so what I did was basically deprecate the old one pointing [Indiscernible: 7:52] and then the next major version I think it's even still there but they haven't had a major version released since maybe they did I don't know so yeah whenever version 4 or 5 comes out then we [Indiscernible: 8:06]

Speaker 1: Okay so but there are still plans to remove deprecated features.

Speaker 2: Yes.

Speaker 1: Do you guys [Indiscernible: 8:12] cleanups.

Speaker 2: [Indiscernible: 8:14]

Speaker 1: But oracle doesn't I mean as you know maybe the Java date API [Indiscernible: 8:21] today actually.

Speaker 2: Okay.

Speaker 1: Well the beta… I have looked at the beta and the beta still has it so…

Speaker 2: [Indiscernible: 8:29] you could argue I mean I am not saying it's a good argument but…

Speaker 1: But they changed the versioning system as well so they've have changed it to…

Speaker 2: Yeah that's what's coming isn't it.

Speaker 1: Yeah it's coming in 9 so the versioning system…

Speaker 2: [Indiscernible: 8:41] yeah I mean that's I mean we are very reluctant but not as reluctant as the others [Indiscernible: 8:49] because I want to say more.

Speaker 1: Yeah sure absolutely please be as verbose as you are.

Speaker 2: So that's just removing stuff right adding stuff removing it and then there's changing stuff for change sake basically they have an existing method but they want to change it we don't do that [Inaudible: 9:07] or we do that very little so because we really bend over backwards to maintain backwards compatibility.

Speaker 1: Why?

Speaker 2: Why because we think that's the key to our market share.

Speaker 1: Okay but who are your major competitors.

Speaker 2: If you are… let me put it this way if you are we had a very interesting maybe it doesn't fit in your format but it's an interesting story anyway so we are working [Inaudible: 9:37] which is the new version so that's going to be all reactive it doesn't really matter [Inaudible: 9:42]

Speaker 1: Please go into that as well.

Speaker 2: And [Inaudible: 9:46] so you can look at those one of the things we are also going to do is move away from a server architecture that can run in Spring 5 [Inaudible: 10:00] going into the lower levels to be more efficient right [Inaudible: 10:14] however the API that [Inaudible: 10:24] is going to use is going to be exactly the same and that's my design to go there… not… I mean I am talking about the annotations [Inaudible: 10:34] we have sort of we don't have a real layer you could look at our APIs having multiple layers [Inaudible: 10:40] or you know those annotations those are very visible and then there's like abstract [Inaudible: 10:50] that's also public I mean it's a public class but we do feel a bit more allowed to change that than the other one right there's difference I mean it's not [Inaudible: 11:04] never change anything right or it is going to… but sometimes you have to [Inaudible: 11:10] go into that later to secure the other [Inaudible: 11:14] you might have to change that and if it's one of those underlying deep classes that's just [Inaudible: 11:23] but these higher level annotations they never change you might add to them but [Inaudible: 11:34] if you have an old application you can just upgrade to the new version but [Inaudible: 11:41]

Speaker 1: That's not something that everyone does so…

Speaker 2: But that's why we are so popular I mean people look at this especially engineers [Inaudible: 11:53] but the moment that you introduce it [Inaudible: 11:59] that they would have changed the user API in new Spring 5 the [Inaudible: 12:04] so no more App controller no more… and it would have been sense because controller is very [Inaudible: 12:09] and you want to get away from [Inaudible; 12:12] but you know or [Inaudible: 12:16].

Speaker 1: I personally don't think it is but…

Speaker 2: It never was but you could argue that if there is [Inaudible: 12:27] in the web then at least it has moved to the client side [Inaudible: 12:30] in JavaScript typically while it used to be done on the server side and then even there you could argue it wasn't ever [Inaudible: 12:38] it's nice to have a label you know to talk about stuff even though it is not the right label so we are using the [Inaudible: 12:49] goal there is to keep our user base [Inaudible: 12:53] if we would have changed that for something that might have been technically superior you are [Inaudible: 12:59] a choice for the users they are basically saying they are going to change Spring is going to change so we've got to learn new stuff for [Inaudible: 13:09] right so [Inaudible: 13:14] so we are introducing a choice basically.

Speaker 1: Yeah but couldn't you maybe still introduce this choice while keeping your [Inaudible: 13:25] customers on site for instance you could add new features or replace older features by deprecating them and still keeping them [Inaudible: 13:31]

Speaker 2: That's what we do, that's what we do I mean but the goal is basically to if there is… sometimes we see [Inaudible: 13:38] right people especially in open source people are working in their spare time on stuff and it's never it's maintaining scrap which you probably [Inaudible: 13:49] and so everybody wants to go in a new version of stuff and especially replacing everything right so you start from scratch [Inaudible: 14:00] and that's the stupidest thing you can do.

Speaker 1: So in your experience do you see people actually moving to the latest version of Spring?

Speaker 2: Oh yeah.

Speaker 1: Well how do you guys track this?

Speaker 2: Downloads we get a lot of feedback on [Inaudible: 14:12] we now have a reputation for people that they can actually use the latest version and still when they are [Inaudible: 14:22] maybe with other frameworks they wouldn't do that but we have a lot of people out there [Inaudible: 14:27] especially on platforms that we don't have I mean we don't use like we don't have older Mainframe Unix variations running you can't do them on VMware so we have people testing that for us [Inaudible: 14:42] but if there is an issue they will let us know.

Speaker 1: Are these clients who actually test all this stuff are they more open source clients, close source industry [Inaudible: 14:53] because let share some from my personal experience because I have [Inaudible: 14:59] a lot of Spring usage I have mined over 15000 clients who use Spring [Inaudible: 15:02] and so we know the exact popularity of the different modules you know core [Inaudible: 15:09] number one [Inaudible: 15:11] number two.

Speaker 2: [Inaudible: 15:13]

Speaker 1: Yeah I found that strange too but I've mined all this data so I have mined and we've seen that a lot of the developers so a lot of the guys who are using Spring are not affected by bugs in Spring are not affected by deprecation in Spring so I'm wondering if there is a separate user base somewhere that uses different features of spring because I looked at your [Inaudible: 15:36] I connected to your Spring repository itself to see what the bugs or what the buggy methods were or you know where the bug fixes really were and it appears that none of the bug fixes that were being done in Spring or now the bug reports were being registered at least from the Open source domain at least from the GitHub open source domain the user [Inaudible: 15:56] not using the same data or not using the same features that were reported as buggy and this is despite mining their entire history so maybe they did it and [Inaudible: 16:05]

Speaker 2: I don't really understand maybe you can there are bug reports out there.

Speaker 1: So you guys have bug reports.

Speaker 2: Yes.

Speaker 1: And you look at them [Inaudible: 16:14]

Speaker 2: Yeah okay so you basically look [Inaudible: 16:19] and then as a consequence these files change.

Speaker 1: Yeah you look at the comments because in the comments tag you guys add the [Inaudible: 16:24]

Speaker 2: Yeah, yeah.

Speaker 1: So I looked at that [Inaudible: 16:28] where the bugs were okay at the same time you see whether these same features were in your bug fixes so the ones that were reported as buggy or [Inaudible: 16:35] whether they were being used in the open source domain in Github.

Speaker 2: Okay usually…

Speaker 1: And there was this huge separation…

Speaker 2: Yeah okay so that's not strange because…

Speaker 1: Yeah I mean for me because for me because the thing is in research we mine [Inaudible: 16:54]

Speaker 2: [Inaudible: 16:56]

Speaker 1: Okay so I want your opinion on this actually.

Speaker 2: Well I mean I am not sure if [Inaudible: 17:06] but I mean I can explain it [Inaudible: 17:08]

Speaker 1: Exactly so that's what I want to know so your clients…

Speaker 2: Yeah and they are I mean basically yeah I would argue that our customer is probably more on the conservative side of things using this stuff for maybe [Inaudible: 17:27] and these people appreciate our conservatism I guess we have I mean we don't have a lot of… we have some paying customers we don't have a lot of paying customers who pay for support on [Inaudible: 17:46] most people think you know open source support is good enough some people banks out there quite a lot of big banks actually they do pay us for that because they want to have a [Inaudible: 17:58] the whole infrastructure is based on Spring so you know they can't really have somebody [Inaudible: 18:05] when stuff goes wrong and so that's there yeah we do we have those we have a lot [Inaudible: 18:13] government agencies all kinds of [Inaudible: 18:17] who use our stuff you can also sometimes [Inaudible: 18:19] for your research to look at the [Inaudible: 18:23] that are in the reports and see what factors they use and you will find that it's usually [Inaudible: 18:29] it's not a lot of open source stuff right it is more like EU or MIL or GOV or BIZ or you know [Inaudible: 18:36]

Speaker 1: Thank you for that thank you very much actually because we did the study and we found it very strange because you know the Hibernate API it also showed similar behavior and Guava which is Google didn't show the same behavior because I guess it's more used by open source developers these days…

Speaker 2: Guava is yeah… [Inaudible: 18:59] new library [Inaudible: 19:01]

Speaker 1: But it's number two in popularity now.

Speaker 2: Yeah it's on GitHub.

Speaker 1: On GitHub yes so…

Speaker 2: That doesn't really…

Speaker 1: Because all my data can only be sourced from GitHub right now and we would love if…

Speaker 2: As long as you have some sort of… as long as you know that's not the real world.

Speaker 1: Yeah, no but that's the argument I am also trying to make actually [Inaudible: 19:18] also generalizable I reckon then this is probably not the case.

Speaker 2: I mean it depends on project I think [Inaudible: 19:31] probably yes but with Spring you write Enterprise applications which are not necessarily [Inaudible: 19:40] on GitHub also right I mean some people do it I guess [Inaudible: 19:46] but that's about it I mean nobody I think people [Inaudible: 19:53] if I look at my own [Inaudible: 19:55] personal experience and spring is more like the bricks that you build your house with it's not really cool but it doesn't have to be cool really it doesn't have to be cool in the sense that you know a you know like some car has to be decent or a truck has to be good at driving but it doesn't have to be cool so I am not saying that Spring isn't cool Spring is cool.

Speaker 1: Yeah it's a pretty cool API I agree I use it myself so…

Speaker 2: [Inaudible: 20:33]

Speaker 1: That's my idea as well.

Speaker 2: Yeah I think you are right.

Speaker 1: And we had a sitting with March Phillip last week from J Unit who said the exact same thing he didn't think that the GitHub data would be representative enough to give you any kind of insight into the actual usage patterns of Big APIs such as J Unit and I don't think this is… I think it's the same thing with Spring as well that probably you know your clients are actually enterprise clients or closed source at the very least I am not really…

Speaker 2: They wish they could use GitHub my clients.

Speaker 1: Exactly I wish they could use GitHub because then I would get data you know.

Speaker 2: Maybe they won't use GitHub you know the public data [Inaudible: 21:10] these companies maybe there's another way [Inaudible: 21:16] of people who are [Inaudible: 21:19] or CVS observation or whatever and they are all their own infrastructure and for good reasons right I mean if you are a bank you don't want to [Inaudible: 21:27]

Speaker 1: [Inaudible: 21;28]

Speaker 2: Even they have a private account if you just don't want to have the risk of [Inaudible: 21:36]

Speaker 1: Absolutely so no but thank you for that insight because it's not really related to this study actually but this is a question I have had actually for the last year [Inaudible: 21:47] last year so because we had this theory that we would see that the most popular parts of an API would have more bug reports associated with it [Inaudible: 21:56] the public parts of the API that have been used on GitHub are incredibly stable.

Speaker 2: In fact, I think the popular parts are much more used they are stable I think it's more the obscure parts [Inaudible: 22:09] depends on the bugs.

Speaker 1: Yeah but then the question of course is who is using these obscure parts right and when we found zero usage then we said okay it must be closed source [Inaudible: 22:17] confirm that so that's …

Speaker 2: I don't think it even… I mean even if you would look at the Applications that have GitHub right they also are self-selecting in the sense they are going to be probably Spring [Inaudible: 22:37] applications you know but what I am saying is you are not [Inaudible: 22:43] a mainframe [Inaudible: 22:46] uses a mainframe on GitHub because it's not cool right and people especially private projects they want to do cool stuff.

Speaker 1: I think they all use Spring Grid because Spring Grid is probably one of the coolest things to expect to tell you the truth because I myself use it [Inaudible: 22:58] so that's…

Speaker 2: I don't I don't use Spring at all so [Inaudible: 23:03]

Speaker 1: Alright so let's talk a little bit about changes where you guys said you did backwards compatibility it was very important to you and you do change stuff can you tell me what decisions really prompt you to change stuff [Inaudible: 23:21]

Speaker 2: [Inaudible: 23:22]

Speaker 1: That's it but I am talking more about new features here what prompts you to say hey I really need this new feature is it a client who comes to you and says hey guys I need this…

Speaker 2: Well yeah it really depends on [Inaudible: 23:35] I mean we have [Inaudible: 23:37] for every release [*Inaudible: 23:38*] but then typically never come out [Inaudible: 23:41] right because we don't get a if you bring in a really big new feature you want to do it, it's unchartered territory right it's new territory and in that sense you want to lead you want to say as a product [Inaudible: 23:59] and this is the way we are going to go and that's not debatable I mean it's debatable but in our group but it's not debatable for the user, user can't say oh hang on you were looking [Inaudible:24:10] but I prefer you would use [Inaudible: 24:14] Java or something like that and right now you are entitled to your opinion.

Speaker 1: Can you tell me maybe why in RS Java.

Speaker 2: That's I don't think it's relevant but there are reasons [Inaudible: 24:27]

Speaker 1: No it's my personal curiosity.

Speaker 2: Because we think [Inaudible: 24:31] is advancing as the [Inaudible: 24:33] which is not built into RS Java and reactive streams especially with reactor which is our own… even RS Java is going to rebuild itself version 2 in order to [Inaudible: 24:45] and it's just it's a better industry standard I think RS Java has a… it's big [Inaudible: 24:53] is very small for that purpose I think it's better as a sort of [Inaudible: 24:59] reactive library I mean it would be nice if [Inaudible: 25:03] use but that's not happening [Inaudible: 25:06] maybe in Java 9 but…

Speaker 1: No, it's not.

Speaker 2: [Inaudible: 25:09] is not coming.

Speaker 1: No.

Speaker 2: Oh it isn't.

Speaker 1: I think it's related to Java 10.

Speaker 2: Oh there you go. Even better, better choice to make see you've got to make some choice.

Speaker 1: So…

Speaker 2: [Inaudible: 25:21] I am not saying that it's a good example because you can use both it's just what we use internally I mean what I am trying to say is that you have to put out some posts in order to show you the direction and that's a new feature if it's a major new feature if you are doing we did rest right [Inaudible: 25:40] I am talking about stuff that I was involved in because that's the one I have most experience with so Spring [Inaudible: 25:45] was all about Rest right and there's a lot of people [Inaudible: 25:50] more of a… in a web search framework or in a [Inaudible: 25:57] framework just put annotations out there and everything will be exposed automatically and I really pushed for it being [Inaudible: 26:06] in our existing MVC framework at the time and that was a choice right I mean if you would have asked our users they probably would have said [Inaudible: 26:17] this other thing where you just put some annotations on it seems much simpler [Inaudible: 26:21] because I think that's right you know.

Speaker 1: But do you ever get feature requests say…

Speaker 2: Yeah we have feature requests so there is a famous saying about [Inaudible: 26:34] right I would ask a customer whether [Inaudible: 26:36] and that's true that is really true I mean sometimes you have users who will go out and request something big but I mean the chances of our… it's not that hard to have a new idea right it's the way you do it everybody knows it is the simple execution that matters so everybody can come in and say I want rest support in Spring like we had a famous issue [Inaudible 27:04] I think five years already at that time very old [Inaudible: 27:10] and of course we had feature requests there and we take into consideration all the things that they ask for but after all we have… we are responsible we have to maintain it we have to keep the backwards compatibility so in the end we decide what's goes in and what doesn't that's basically our work.

Speaker 1: Okay does [Inaudible: 27:32] have any input?

Speaker 2: No.

Speaker 1: So it's just…

Speaker 2: I mean obviously we are not going to make it hard for [Inaudible: 27:38] because at that point I am eating out of my own wallet basically but in the framework itself [Inaudible: 27:47] people have tried in the past not difficult but other companies have tried say well can't we put this in there I said no it's not going to work that way.

Speaker 1: So alright do you guys as developers have any kind of…

Speaker 2: [Inaudible: 28:01]

Speaker 1: Oh sure [Inaudible: 28:03]

Speaker 2: Oh sure. [Inaudible: 28:07]

Speaker 1: Okay so another question I have so you said that you perceive that your clients always go to the latest version.

Speaker 2: No, no, no we want to give them no reason whatsoever not to do that.

Speaker 1: But…

Speaker 2: We see a lot of people staying on the same version.

Speaker 1: Yeah but does that annoy you.

Speaker 2: No.

Speaker 1: No.

Speaker 2: I mean it's their risk basically I mean if something works they can't change it if they have a running application and it's running and they don't have any issues with possible security issues that might have been fixed I mean what can I do there's nothing I can do.

Speaker 1: That's what I want to ask you so do you put out advertisements… do you put out posts saying hey guys Spring 4.1.1 is just the best thing ever and…

Speaker 2: I personally never did that.

Speaker 1: Okay.

Speaker 2: I mean…

Speaker 1: I mean I have seen your blogs on Spring…

Speaker 2: [Inaudible: 29:12] if you have seen my blogs you probably have also seen that I never write positively about [Inaudible: 29:16] I am not a evangelist or anything so but we want people to upgrade yes and we have a [Inaudible: 29:35] I mean obviously I am not exactly sure which version we support but we don't support version 2 and 1 we support version 3, 4.2 something I don't know yeah we got 4.2, 4.3 and now in a couple of weeks next probably we'll start the main [Inaudible: 29:59] Spring 5 so there's always like… [Inaudible: 30:05] I am not sure of this but you can also get support on version 3.

Speaker 1: Okay so but what happens if there was a bug… what happens if you get a bug report from version 2 do you guys ignore it in Jar?

Speaker 2: Yeah well if it's [Inaudible: 30:33] version 3 and… we currently support version… [Inaudible: 30:31] I mean you have to pay for the support it's not free so we are also supporting older versions but you have to pay for it that's basically [Inaudible: 30:42] how we make our money because [Inaudible: 30:44] people using this but I mean there are like I said conservative clients are there banks etc. who do pay for this.

Speaker 1: [Inaudible: 30:53] from the open source [Inaudible: 30:54] is that we've seen a lot of people preferring to hang back so a lot of people seem to be I found usage of Spring 2.5.1 out there.

Speaker 2: Yeah that's fine.

Speaker 1: [Inaudible: 31:07] so it's still there so my point is that you as an API developer I want your personal opinion here what is your preferred behavior what is your preferred…

Speaker 2: I don't care.

Speaker 1: You don't care.

Speaker 2: I really don't care at all.

Speaker 1: You really don't care okay.

Speaker 2: As soon as I mean it's not completely true but as soon as I [Inaudible: 31:28] over the fence then I am done I mean I like… I try not to think about it too much about how many people and how they are using my stuff because that scares me I think about the not millions but yeah maybe or indirectly millions of people are using my software also for mission critical stuff and that just freaks me out…

Speaker 1: But you don't want any insight into how people are using it so like exactly the usage patterns of Spring.

Speaker 2: [Inaudible: 32:03] but I am always more of a teacher rather than yeah I don't know it's very hard to get that data right I mean [Inaudible: 32:20] you could say well people are still using 2.5 [Inaudible: 32:23] it is good enough for them fine if they don't need the new features they don't need the security [Inaudible: 31:31] issues that might have occurred in that version fine I mean there's nothing I can do about it the problem would be in open source development this is a generic thing you never… you get very little interaction with the user anyway the only interaction you get is when stuff breaks.

Speaker 1: Of course.

Speaker 2: If stuff is running fine then you don't hear from them right.

Speaker 1: So let's get to the breaking thing has it happened often. That you get like a billion complaints that hey guys you really, really screwed this up I mean maybe you have one example of a breaking change.

Speaker 2: I can't think of any I mean intentional [Inaudible: 33:19]

Speaker 1: I mean any either…

Speaker 2: If it's unintentional I probably would have heard it if it's basically a regression right [Inaudible: 33:28]

Speaker 1: And if it's an essential thing.

Speaker 2: [Inaudible: 33:34]

Speaker 1: My question was what your client reaction. So I am sure you say security or bug or whatever what do they say do they come on to GitHub and say [Inaudible: 33:48]

Speaker 2: You know we say well there's a reason it doesn't happen that often I try to think of the other thing that I can think of are [Inaudible: 34:00] the only one I can think of is security related issues where we did something which was [Inaudible: 34:07] so you have to explicitly allow for something [Inaudible: 34:18] allowed something by default and then the [Inaudible: 34:22] so people have to change to get the old behavior people have to change the flag explicitly enable that's not a big thing.

Speaker 1: But you guys [Inaudible: 34:30] I think because I've read a lot of your Spring docs I think you guys have deprecated the method and told them that this is [Inaudible: 34:37] I don't think it was immediately fixed or removed so…

Speaker 2: Okay I don't know…

Speaker 1: I mean this was based on like [Inaudible: 34:46] huge analysis on deprecation and so I have read a lot of your Spring Docs and Java Docs [Inaudible: 34:52] these methods when you do deprecate you give a very good reason normally that it's security issues you also give a date I've seen this that it says it will be removed by this date which I found was pretty cool because I think you guys were the only ones that did this in the open source world.

Speaker 2: In the Spring Framework.

Speaker 1: In the Spring Framework and I [Inaudible: 35:11] but I have seen that this is more the level of documentation I found that actually pretty [Inaudible: 35:18] it was the most verbose and was [Inaudible: 35:20] documentation.

Speaker 2: I can't say that's a policy because I don't do it I mean we are very [Inaudible: 35:34] I mean I am to be honest I am not that much involved with the day to day [Inaudible: 35:39] I am more like I said working on the next version and I am especially mentoring the guys who are writing that so they [Inaudible: 35:47] like I said I have been doing this for ten years and my role is slowly becoming more… I am becoming an advisor in the background so I am not dealing that much with the day to day issues anymore I am not constantly tracking [Inaudible: 36:02] to see what happens.

Speaker 1: I still see your name a lot in the [Inaudible: 36:05]

Speaker 2: Yeah I still do I mean as people come in and say what do you think about this and sometimes you can [Inaudible: 36:08] most of the times you can't even see it because it's hidden for developer's people obviously they [Inaudible: 36:16] internal discussion which is always fair I don't think of any we don't shout at anyone alright so there's no offensive stuff going on it's just a technical decision because sometimes you don't want [Inaudible: 36:34] you can do that on the issue [Inaudible: 36:38] so yeah this particular style of referring to a new method and saying we will change it by this time maybe that's [Inaudible: 36:46] we had a lot of security issues recently last year mostly related to XML parsers doing stupid stuff by default basically [Inaudible: 36:59] Spring problem but so we have these XML marshalling models which [Inaudible: 37:07] by default [Inaudible: 37:14] can do all kinds of nasty stuff but the problem is that this module marshalling module is not just used for web [Inaudible: 37:30] but depending on the use case you want to disable or enable some more [Inaudible: 37:39] so that was a bit of an issue there yeah I mean if that's what we do then I am sure you are right and I've personally not done this. As far as I know there's no hard rules in this.

Speaker 1: Okay there's no hard rules because I have seen it quite a lot [Inaudible: 37:56]

Speaker 2: Well maybe people do that now [Inaudible: 38:00] if that's so that's a good thing.

Speaker 1: It is.

Speaker 2: Yeah, yeah, yeah.

Speaker 1: So I want to get to deprecation.

Speaker 2: Sure.

Speaker 1: And well you said that you deprecate because of security bugs or something like that do you think it's a good language feature as an API developer.

Speaker 2: Yeah I think it's very useful.

Speaker 1: Do you think… so what is the role that it really plays in Spring other than just like [Inaudible: 38:24]

Speaker 2: It's a no entry sign basically [Inaudible: 38:27] going the wrong way go back use this other lane.

Speaker 1: Okay but in your experience now as a developer an API developer do you react to a deprecated method so say you were using a Java date API and it suddenly magically got deprecated [Inaudible: 38:43] let's say you were using that in Java 3 you went to Java 5 or Java 6 so then your code breaks what would you do?

Speaker 2: You mean I am still using it.

Speaker 1: Yeah.

Speaker 2: The program is still in active development [Inaudible: 38:59]

Speaker 1: So you went from one version where it was being used and now deprecated you went to another version [Inaudible: 39:04]

Speaker 2: Yeah, yeah, yeah I am still working on this deprecation.

Speaker 1: Yeah.

Speaker 2: I am still… okay then I would get rid of it.

Speaker 1: Definitely.

Speaker 2: Yeah.

Speaker 1: And is this something which you do regularly in your [Inaudible: 39:13]

Speaker 2: [Inaudible: 39:14] then yes I am mostly using… we are using third party modules especially in Spring [Inaudible: 39:23] so yeah they say that you shouldn't use this anymore [Inaudible: 39:36] because the next version is probably going to be gone not everybody has this nice policy that we have of keeping stuff around for [Inaudible: 39:43] major or minor versions most likely it's just saying there's one minor version that is deprecated and next minor version it's gone so that's sort of the last chance you got to change this now otherwise.

Speaker 1: And in Spring does this happen often where…

Speaker 2: Yeah we [Inaudible: 39:57] yeah we change a lot especially like I said with the JSON libraries itself and the problem is that not all these libraries like you said have this nice [Inaudible: 40:08] major minor version and we have to [Inaudible: 40:11] we cannot say to all of our users well all of you update to JQSON 3 [Inaudible: 40:16] because that's not going to happen realistically that's not going to happen so [Inaudible: 40:21] that we do in order to support another version is this method available in this class if so invoke it otherwise invoke this other method that's what we do [Inaudible: 40:38] I mean you can look at this as another… that's more or less required if you are building a framework which has to deal with multiple versions I mean the alternative will be to create another module might be [Inaudible: 40:48] three, four, five but you can't do that for every [Inaudible: 40:55] you can't have a Spring [Inaudible: 40:58]

and then a Spring Web JSON [Inaudible: 41:04] four module you know it doesn't work because you end up with [Inaudible: 41:09] so we have to use [Inaudible: 41:14]

Speaker 1: So I want to get back to more of deprecation as a language feature do you think there's enough incentive for a developer to change his invocation [Inaudible: 41:27] doesn't break anything okay you guys do it personally but do you mark something as deprecated [Inaudible: 41:33] security bug it could be just anything because there's a better way to do something.

Speaker 2: I mean I am not sure how much more incentives you could [Inaudible: 41:40] I mean the only problem with… the problem with deprecation is that the time between the first notice let's call it that and the eviction can be very short [Inaudible: 41;51] like I said a lot of frameworks libraries they one version they deprecate something the next version it's gone so if you missed one main minor version somewhere then all of a sudden it's gone I think you should be conservative not as much as Java is trying to do.

Speaker 1: Java is a bit too conservative.

Speaker 2: Yeah but you know I think we are doing a pretty good job [Inaudible: 42:11] Spring 5 is going to come everything is going to be marked as deprecated it's going to be removed in a week really for sure.

Speaker 1: Okay.

Speaker 2: Because well depends on what it says but typically version 3… no version 4 has been around for 2 years now something I guess three years [Inaudible: 42:32] a lot of stuff has been collected deprecation has been [Inaudible: 42:36] so let's get rid of all that.

Speaker 1: Okay so again [Inaudible: 42:43]

Speaker 2: That's the idea anyway I am not sure if that's the practice that's the theory.

Speaker 1: So [Inaudible: 42:49] this is the most remarkable statistic I have ever seen because I think [Inaudible: 42:56] only 41 were ever affected by deprecated [Inaudible: 43:00]

Speaker 2: From us.

Speaker 1: From you guys.

Speaker 2: Right.

Speaker 1: So from Guava there were like 750 out of 3000.

Speaker 2: Yeah.

Speaker 1: And there were….

Speaker 2: Yeah so you want to know why… how we can do this.

Speaker 1: Yeah how on earth do you guys affect only 41 people.

Speaker 2: We think about our API a long time a very long time.

Speaker 1: No it's phenomenal because this is you guys were the [Inaudible: 43:23]

Speaker 2: Yeah.

Speaker 1: I mean I would show you the paper but…

Speaker 2: Oh I want to see that I do want to step outside to have a smoke.

Speaker 1: Oh sure let me just stop this and…

Speaker 2: [Inaudible: 43:36] I will be right back.

Speaker 1: [Inaudible: 00:01] alright so let's go back into the deprecation you guys said… you said [Inaudible: 00:06] we do a great job here.

Speaker 2: Oh we think I mean it's as much as not doing something as it is about doing something so if you are not a hundred percent sure as to how you can add a certain feature we just don't do it because if you would add it you know you would have to change it the next release because you did it wrong then we might as well not [Inaudible: 00:32] so we experiment a lot but not everything makes it into the framework.

Speaker 1: But do you have any more Beta version maybe…

Speaker 2: Yeah we have [Inaudible: 00:41] but those are [Inaudible: 00:49] should be I mean by that time there's already sort of that stage is already done we've picked what we can add [Inaudible: 00:59] I mean I am just talking about the way I work right I mean obviously different people work in different ways but this is the way I think that we should do things and we effectively also do it so yeah like I said it's as [Inaudible: 1:21] it is about doing it… it's about adding something I mean and it comes into [Inaudible: 1:34] because a typical feature of [Inaudible: 1:37] is going to be somebody [Inaudible: 1:41] so here's a patch [Inaudible: 1:49] where I add the fix that I need with a Boolean flag or something like that which.. the Boolean flag could be as simple as do something for customer [Inaudible: 2:01] it solves that particular problem while our job is to see I mean especially when I was beginning [Inaudible: 2:11] open source development I was really happy about this [Inaudible: 2:14] let's show how much I appreciate the help [Inaudible: 2:20] nowadays that's probably the last think I will do because I take ownership of that code right typically I am not saying that there isn't a problem [Inaudible: 2:41] definitely is a problem otherwise [Inaudible: 2:44] but it might be that the solution [Inaudible: 2:47] while the problem might be a single instance of a whole range of different problems right so rather than having a Boolean flag or something we might say lets' introduce a strategy here [Inaudible: 3:03] and you can write your own interface or [Inaudible: 3:09] and the case that the user described the issue is just one of those cases so you might [Inaudible: 3:13] I am trying to think of an example I hope you can understand so some user wants something… feature we think well that's you want this but by next week another guy is coming but he [Inaudible: 3:29] slightly different right it happens, it happens quite a lot and that's why we yeah we try to think well let's [Inaudible: 3:38] can keep backwards compatible right away rather than introducing a Boolean first then introducing the… then [Inaudible: 3:45] and deprecate the Boolean so you effectively know what you are doing [Inaudible: 3:54] we always know what we doing but sometimes [Inaudible: 3:58] you can sort of foresee the deprecation already if it's sort of [Inaudible: 4:01] if I think about it a little more and look at it as being a class one instance of a class a bigger problem then yeah and I mean the fact I guess [Inaudible: 4:19] that it still works.

Speaker 1: Well it's… as I said it's phenomenal because everyone else had above ten to fifteen percent of the clients affected [Inaudible: 4:32] even more so about twenty to twenty-five you guys were miniscule.

Speaker 2: [Inaudible: 4:36] Java is a library, we are a framework so we call [Inaudible: 4:45] the classes and we also especially the clients have made a very strong case for not needing to depend yourself on Spring right Spring obviously has a big class library in it as well but we typically say I mean we have our own Spring [Inaudible: 4:58] but we don't say use this [Inaudible: 5:02] in the Java docs don't use this right it says there this is for internal use if you are looking for a [Inaudible: 5:09] you probably want [Inaudible: 5:11]

Speaker 1: [Inaudible: 5:13]

Speaker 2: And that has to do with the fact that we take it requires your dependencies you cannot include even though there is something wrong with [Inaudible: 5:21] we still couldn't depend on it because we don't we only require one thing and that's [Inaudible: 5:32] I mean obviously if you want to do web development you need to have a server API [Inaudible: 5:38] we don't require [Inaudible: 5:45] yeah I mean dealing with deprecation for me really has to do with upfront learning [Inaudible: 5:53] thinking about simplifying so you cannot wait and that takes a long time and there is no trick or anything I mean if there would be a simple trick I could tell you [Inaudible: 6:06] I think maybe there is but I didn't learn it yet but it is about constantly trying to simplify you have an API reducing it making it smaller getting [Inaudible: 6:16] trying to [Inaudible: 6:21] entry points as possible and the more you expose the more you need to keep backwards compatible if you only expose a single method there is very little that can go wrong with that right if you have to… so yeah reducing the customer facing service [Inaudible: 6:42]

Speaker 1: I must say you [Inaudible: 6:44] my final question really here is your opinion again do you think by deprecating features but not really removing them just deprecating them would you disincentivize developers from transitioning from an older version of the API to a new version.

Speaker 2: No [Inaudible: 7:04]

Speaker 1: If they don't need new features.

Speaker 2: Yeah I don't think so I don't think people need to [Inaudible: 7:09]

Speaker 1: You don't think okay yeah.

Speaker 2: [Inaudible: 7:12] it doesn't work I mean obviously if you are still actively working on it if you are still loading the Application in your IDE and stuff I mean that's by far the most Spring Apps are running [Inaudible: 7:24] most Spring Applications out there are running and they are probably running on Spring 1.2 we made a very big point like I said [Inaudible: 7:35] in 2005 yeah 5 we made [Inaudible: 7:45] we were working on Spring 1.2 was out there and 2.0 was coming out, 1.3 was coming out but we thought there's so much stuff in there we wanted to call it 2.0 basically we started the trend there of making our major versions always also backwards compatible with the minor versions so 1.3 wasn't 2.0 [Inaudible: 8:07] replacement for 1.2 and it still should be like that I mean for 95 percent of the cases it is a [Inaudible: 8:14] replacement you could just take it [Inaudible: 8:16] no need to recompile it even [Inaudible: 8:18]

Speaker 1: So [Inaudible: 8:21] knows that.

Speaker 2: Yes.

Speaker 1: Because I was looking into the [Inaudible: 8:24] of your library and for some reason they still compile with 1.4 Java I am not sure why [Inaudible: 8:33] version is 1.4.

Speaker 2: Recent version [Inaudible: 8:36]

Speaker 1: No I mean [Inaudible: 8:39] the older ones and it was…

Speaker 2: [Inaudible: 8:42] I mean we support Java versions older than [Inaudible: 8:48] typically because of IBM, IBM has their own [Inaudible: 8:50]

Speaker 1: Yeah, yeah [Inaudible: 8:53]

Speaker 2: [Inaudible: 8:54] then you have to install [Inaudible: 8:56] right.

Speaker 1: [Inaudible: 8:57] again you explained another thing [Inaudible: 8:58] because you guys were the only ones that were sitting with 1.5 I was like why [Inaudible: 9:03] on 1.4 when everyone else has gone to 1.6 or 1.7 what do you guys actually compile with.

Speaker 2: Which version right now.

Speaker 1: Yeah.

Speaker 2: Well next version I can tell you that because that's what I am working that's going to be Java 8 [Inaudible: 9:17] that's because the Java 8 [Inaudible: 8:22] quite quickly and it's actually a life changer especially with [Inaudible: 9:27]

Speaker 1: Yeah people are loving it.

Speaker 2: [Inaudible: 9:29] we don't want to do that with [Inaudible: 9:31] classes even though [Inaudible: 9:33] it's very nice [Inaudible: 9:35] it's quite sweet.

Speaker 1: Alright so that was really the end of my interview.

Speaker 2: Well we can talk on I think there's a lot more to say.

Speaker 1: We can talk on but yeah sure [Inaudible: 9:46] recording.

Speaker 2: [Inaudible: 9:47]

Speaker 1: No but I must say it really surprised me when I found 1.4 in there and then 1.5 a little later and you were looking at whether API developers like you guys yourselves started using the latest features deprecation like [Inaudible: 10:05] because Java had the [Inaudible: 10:08] annotation which was a Java doc annotation.

Speaker 2: Yeah [Inaudible: 10:11] in 1.0.

Speaker 1: Yeah so that was in there and then 1.5 introduced the [Inaudible: 10:16] annotation.

Speaker 2: Yeah annotations were introduced in 1.5.

Speaker 1: Because [Inaudible: 10:22] if you noticed in how you guys mark deprecation in your Spring framework because the reason being obviously you could only mark it using the Java Doc annotation and you guys kept that in there despite moving on in the [Inaudible: 10:37] versions.

Speaker 2: No because I think the annotations cannot do comments cannot say see use this instead.

Speaker 1: No so the [Inaudible: 10:44] annotation which came in 1.5 which is the one that is supposed to give a compiler warning because according to the Java standards or the actual Java library standards the Java doc annotation is not supposed to throw any kind of compiler warning and so the Sun JDK does do this as a feature.

Speaker 2: Yeah, yeah.

Speaker 1: Okay and that's why [Inaudible; 11:01] can pick it up so because I used Spring a lot [Inaudible: 11:04] your source code says you use the Java doc annotation but not the source code annotation the source code annotation according to Java standards is supposed to throw a warning.

Speaker 2: Yeah.

Speaker 1: But you guys use it inconsistently in the sense you would sometimes add only the Java doc annotation [Inaudible: 11:18] the source code annotation.

Speaker 2: I see.

Speaker 1: But you really screw up on [Inaudible: 11:21]

Speaker 2: Right okay well [Inaudible: 11:23] because the temptation did exist.

Speaker 1: Exactly and that's why I looked deeper and I said oh the [Inaudible: 11:31] version is here and in 1.4 obviously the annotation didn't exist so…

Speaker 2: Yeah that's right we actually have a… we care about deprecation so much that we also have a Spring post processor I wrote it I am not sure if it's still in there but at least I wrote it the problem with what we had especially when we were still using [Inaudible: 11:49] so I actually wrote a post processor because [Inaudible: 12:11] warning that's now I think included by default if you do that most people now use Java [Inaudible: 12:20] so then [Inaudible: 12:22]

Speaker 1: No it's definitely so we've noticed at least not with Spring that much but with other libraries where the behavior or the reaction to deprecation if you just delete the invocation [Inaudible: 12:38] so they don't seem to care so the guys using Guava and which is why I keep questioning the…

Speaker 2: So calling the libraries [Inaudible: 12:49] gets deprecated and [Inaudible: 12:52]

Speaker 1: No they could only [Inaudible: 12:55] but they don't seem to think it's worth their time to really go to the…

Speaker 2: [Inaudible: 13:02]

Speaker 1: Yes, they do I mean these are big APIs Guava, Spring, Hibernate [Inaudible: 13:09] and there's one more…

Speaker 2: That's interesting.

Speaker 1: Exactly so…

Speaker 2: I think they just want to get rid of the warning [Inaudible: 13:18] I guess that makes sense.

Speaker 1: Yes.

Speaker 2: The quickest way to get rid of the warning is to remove it.

Speaker 1: Well the other way is also to say suppress warning.

Speaker 2: Yeah.

Speaker 1: Because you can also [Inaudible: 13:26] but the thing is the warning is not really… it's a warning [Inaudible: 13:31]

Speaker 2: It is irritating as [Inaudible: 13:33]

Speaker 1: [Inaudible: 13:36]

Speaker 2: [Inaudible: 13:38] saying that people don't read popups either.

Speaker 1: Exactly so…

Speaker 2: If you are talking about IDE [Inaudible: 13:42] computer [Inaudible: 13:44] research has shown it is the worst place to convey information.

Speaker 1: So because you know Small Talk the language there if you mark something as deprecated what happens is it actually breaks [Inaudible: 13:57] in the sense it raises the debugger and [Inaudible: 14:02] tells you that you are using a deprecated feature you know watch out [Inaudible: 14:05]

Speaker 2: Because it is a very [Inaudible: 14:07]

Speaker 1: No but it is around the original [Inaudible: 14:13]

Speaker 2: [Inaudible: 14:16]

Speaker 1: So I was at [Inaudible: 14:26] in February so there was the Java [Inaudible: 14:29] who is now the head of the Java [Inaudible: 14:32] Project he was there and he was talking about the same stuff where [Inaudible: 14:36] he said this deprecated annotation in its current form [Inaudible: 14:40] it doesn't give a developer any incentive to transition you mark something as deprecated wonderful it's a compiler warning big deal who cares about it right and the orange like that [Inaudible: 14:50]

Speaker 2: Yeah, yeah, yeah.

Speaker 1: That's it [Inaudible: 14:52] so he said they are actually changing it now in Java 9 they are actually changing the annotation where they will give you more information [Inaudible: 15:01] adding messages to this.

Speaker 2: Obviously yeah.

Speaker 1: So they want to make it more in your face.

Speaker 2: Not sure [Inaudible: 15:09]

Speaker 1: Exactly so this is what we are trying to research is with this help or do we need to go more into like [Inaudible: 15:18] where we say hey broke your code man so fix it especially because API developers give amazing documentation [Inaudible: 15:27] it's a wonderful documentation out there.

Speaker 2: Yeah oh yeah I can certainly see the reason why people ignore it I mean deprecation really should be a last the last effort I think sometimes it is applied differently like I said it's a lot easier a lot more fun to work on something new [Inaudible: 15:51] maintain something to bend over backwards to keep the backwards compatibility I mean [Inaudible: 15:58] is much the same as I can [Inaudible: 16:13] in Spring 5 so that's the fourth time I guess and you can really… I mean obviously [Inaudible: 16:25] easier to say well you know new framework new rules let's start from scratch let's thing about it again but like I said [Inaudible: 16:35] and you are… and the same applies I think to deprecation I mean it's a… it should be a last ditch effort I am not saying oh well I wrote this or some other guy wrote this three years ago [Inaudible: 16:53] perhaps I mean I don't use Guava I don't use [Inaudible: 16:59] library I use J unit obviously I know how if I see any deprecation that I do it apparently seems to be very different than what other people do.

Speaker 1: Well at least in your [Inaudible: 17:10]

Speaker 2: Right, yeah, yeah.

Speaker 1: No so I think Guava the other thing that's happened with them is that they have nineteen major releases [Inaudible: 17:21] five years old, six years old [Inaudible: 17:24] so they've been releasing very frequently and they've been changing their API every at a rapid rate.

Speaker 2: And for me it suggests that they didn't think about it enough beforehand I mean it's [Inaudible: 17:36]

Speaker 1: No so we can't talk anyone from them because it's Google it's all internal they claim open source development it's not open source development because they have internal [Inaudible: 17:43] to GitHub

Speaker 2: [Inaudible: 17:47] you would argue that we don't do open source development either.

Speaker 1: True but [Inaudible: 17:54]

Speaker 2: Yeah I mean you can contribute but I mean at the end of the day it's our product it's how we make [Inaudible: 18:03] it's also our responsibility so I think people tend to think about open source it doesn't [Inaudible: 18:14] everything you see so many cases where people just want to throw stuff your way they wrote something they don't want to maintain it they just dump it on [Inaudible: 18:22]

Speaker 1: Which is a Microsoft [Inaudible: 18:26] now.

Speaker 2: Also from [Inaudible: 18:29] perspective saying they are pushing [Inaudible: 18:31] to us because it is not related to the business [Inaudible: 18:36] they want us to maintain it or [Inaudible: 18:38] it happens but [Inaudible: 18:44] I mean obviously the smaller the PR is the more bigger chances are that you will resolve it [Inaudible: 18:52] people introduce a lot of new types, new classes and new interfaces and it doesn't happen very often that people get it right the first time we might I mean depending on the input we might say this is good enough [Inaudible: 19:10] we sometimes try to push [Inaudible: 19:15] because it's always the question of how much the other party is interested in [Inaudible: 19:20] sometimes people just dump something [Inaudible: 19:22] okay I want to [Inaudible: 19:25] this feature [Inaudible: 19:27] if you want to do this then this is [Inaudible: 19:30] current design it could be the stupidest formatting it could be the stupidest copyright, notices all those kind of it could be as stupid as how many empty lines you have between [Inaudible: 19:40] and input statements all the kind of that's what we care

about and If you want to go this way I mean [Inaudible: 19:46] not easy but it's open [Inaudible: 19:49] a lot of people just don't have that time.

Speaker 1: Can I ask you something I [Inaudible: 19:54] that you process.

Speaker 2: [Inaudible: 19:58] that's it. You want to eat something here or…

Speaker 1: Sure [Inaudible: 20:05]

Speaker 2: Well yeah I am getting hungry I have a little baby who wakes me up at seven so [Inaudible: 20:16]

Speaker 1: I mean in our research group [Inaudible: 20:20]

Speaker 2: You probably take lunch [Inaudible: 20:27]

Speaker 1: [Inaudible: 20:30] terrible.

Speaker 2: But have something [Inaudible: 20:34]

Speaker 1: [Inaudible: 20:35]

Speaker 2: [Inaudible: 20:39] and it depends I mean we have an internal [Inaudible: 21:03] you've been reorganized or if your code has been reorganized basically means [Inaudible: 21:06] empty lines [Inaudible: 21:14] rearranging stuff and sometimes also reflecting stuff but I mean a more senior somebody becomes the less organizing he needs if I [Inaudible: 21:23] some code chances are [Inaudible: 21:25] I look at a lot of code too my goal is obviously [Inaudible: 21:44] my goal is always to get rid of stuff and people say that I do that quite well because like I said the less you have the less you have to worry about.

Speaker 1: Do you guys use [Inaudible: 21:59]

Speaker 2: Yeah we have [Inaudible: 22:04] which we use and [Inaudible: 22:10]

Speaker 1: But these are just [Inaudible: 22:13]

Speaker 2: [Inaudible: 22:16] stuff like [Inaudible: 22:19] so yeah we don't I mean if you [Inaudible: 22:37] you can just comment a line saying what you are doing [Inaudible: 22:43]

Speaker 1: So my question is because we do a lot of research in [Inaudible: 22:45] and we've noticed [Inaudible: 22:48] the thing is getting a proper [Inaudible: 22:53] of the actual changeover that a change is made.

Speaker 2: [Inaudible: 22:59]

Speaker 1: [Inaudible: 23:08]

Speaker 2: You don't have to learn Dutch if you are living in the Netherlands.

Speaker 1: [Inaudible: 23:15]

Speaker 2: I know; I know but [Inaudible: 23:20] I am going to go for just a sandwich [Inaudible: 23:30]

Speaker 1: [Inaudible: 23:39]

Speaker 2: [Inaudible: 23:43]

Speaker 1: I think I'll have the hamburger [Inaudible: 23:48]

Speaker 2: Oh there you go that's way too much for me I need to do some thinking [Inaudible: 22:52]

Speaker 1: if I eat a [Inaudible: 23:57] meal I will not be [Inaudible: 23:59]

Speaker 2: That's interesting.

Speaker 1: I think it's a cultural difference right.

Speaker 2: So yeah what was I saying.

Speaker 1: [Inaudible: 24:06] so you don't really have to change order you don't know exactly which file to change in what order [Inaudible: 24:16] alphabetical order.

Speaker 2: Right well what I do when I introduce a big [Inaudible: 24:24] and I want to show people what [Inaudible: 24:26] that's what I do so the first comment will be introducing something new then the second comment will be using the new stuff and then the third comment might be getting rid of the old stuff there's a logical order.

Speaker 1: Do you think it would be useful to have some of these [Inaudible: 24:46] for instance giving you like all the line spacing or [Inaudible: 24:51] checking automatically so would it be.

Speaker 2: Yeah, yeah, yeah it would be very easy but…

Speaker 1: Because we are currently doing research on this so there are some [Inaudible: 24:59]

Speaker 2: Also we have some of our internal tools we have tools I mean we I don't know how much you know about the Java compiler but the fact that you use [Inaudible: 25:16] version only means that the compiler [Inaudible: 25:20] it doesn't have anything to say about the library [Inaudible: 25:24] you still can use 1.8 library while compiling with 1.4 obviously we don't want this right we want if we say we want to at least know where we used our [Inaudible: 25:36] where we used our seven classes [Inaudible: 25:40] we have written our own annotations for this basically and we have [Inaudible: 25:45] right now we have like [Inaudible: 25:51] uses Java 8 and then we can just say if a class uses Java 7 class and it doesn't have the annotation [Inaudible: 26:03] stuff like that but those tools are very specific to us and another thing is that unfortunately a lot of our code style is pragmatic and cannot really be fit in rules I mean we have sort of a [Inaudible: 26:24] it's all about context right I mean you cannot say [Inaudible: 26:36] and then it doesn't really work that way because sometimes we have to break these rules and we have to make something more [Inaudible: 26:46]

Speaker 1: [Inaudible: 27:01] so the thing is we found a lot of these [Inaudible: 27:12] out there so [Inaudible: 27:15] issue such as [Inaudible: 27:17] I don't know but say there were [Inaudible: 27:27] does he really go to every one of them or do you actually…

Speaker 2: No he doesn't I mean like I said [Inaudible: 27:36] upgrading of a library or [Inaudible: 27:46] like I said [Inaudible: 27:49] introduce new types, new classes [Inaudible: 27:53] introduced as it is right that might be back and forth that might be a complete rework [Inaudible: 28:03] we just take it and then we go with it yeah I mean the idea is that you I mean it's all about phases really the initial work is being done [Inaudible: 28:26] general captain lieutenant sort of hierarchy right you have like the big general and

then you have all kinds of people under him doing smaller bits but at some point they are going to show [Inaudible: 28:46] they already made sure that it uses [Inaudible: 28:49] I mean I would personally prefer the spaces but [Inaudible: 29:01] so we stick with that [Inaudible: 29:06]

Speaker 1: How do you check this stuff so do you actually like…

Speaker 2: [Inaudible: 29:14] I just reformat stuff and then do some additional stuff manually which cannot be done [Inaudible: 29:20] I mean that's the problem I think we do have is that the code style we have cannot be effectively I haven't found a single clue that can do everything automatically unfortunately I personally would say and I have had this discussion that we should change our code style [Inaudible: 29:40] because otherwise it's not just [Inaudible: 29:46] I have lost that fight so I am not going to fight it anymore.

Speaker 1: So would it be useful so maybe in the [Inaudible: 29:50] when someone actually makes a [Inaudible: 29:52]

Speaker 2: Yeah.

Speaker 1: That there was an [Inaudible: 29:54] that gave them all the information about coverage and how they may or may not have [Inaudible: 29:59] the new changes and it also gave them the [Inaudible: 30:01]

Speaker 2: Possibly I mean that might also scare off people right if you are coming across the first thing you have to do is to people I think you are trying to do two things right I mean obviously [Inaudible: 30:18] on the code but it's also about building a relationship with the user the first thing you do is [Inaudible: 30:24] right.

Speaker 1: But do you guys have a contributor's guideline right.

Speaker 2: Yes.

Speaker 1: But then do you enforce it?

Speaker 2: No I mean the code style and everything like that [Inaudible: 30:43] we don't like I said it depends on the interaction that you have with the contributor I mean I am going to suggest it but I mean you can really see like I said you can really see some issues people just [Inaudible: 30:57] the code and think that's my [Inaudible: 30:59] I don't want to do anything anymore that's fine [Inaudible: 31:02] that's for sure because it's not yours anymore right and that's one extreme the other extreme is that people might want to show off their skills because they want to join [Inaudible: 31:22] and there's everything in between and if you have like the latter that I mentioned [Inaudible: 31:29] obviously he's going to be very interested in conforming to our contribution guidelines because he wants to get a job [Inaudible: 31:38] the other guy he works at Morgan Stanley he doesn't care he doesn't want his name to show up because it's he doesn't want to show… he doesn't want anybody to know that Morgan Stanley uses Spring so that's [Inaudible: 31:52] the tool that you described will be very useful for this guy for that guy it might be offensive you know what I mean [Inaudible: 32:01]

Speaker 1: Yeah, yeah. No it's a question that we [Inaudible: 32:05]

Speaker 2: [Inaudible: 32:08] that's what I am saying I think the social aspect of PR is much more important than conformation to these rules.

Speaker 1: Okay so you personally think Spring wouldn't have any [Inaudible: 32:24] would be reluctant to use [Inaudible: 32:29]

Speaker 2: I think all projects do Linux does for instance right I mean I am not following Linux development that closely but when I've read about it they have tools which show if the patch is conforming to their guidelines I think that's one I think you are creating a barrier that not everybody is willing to climb and you might say well [Inaudible: 32:55]

Speaker 1: So I don't think it's more on the lines of where we want to say that we don't want your stuff more on the lines of saying that you know that should be maybe a little better.

Speaker 2: Yeah but it's still [Inaudible: 33:14]

Speaker 1: I am saying without human interaction because you would still have to do this yourself.

Speaker 2: Yeah, yeah of course.

Speaker 1: So my point is [Inaudible: 33:22]

Speaker 2: No.

Speaker 1: So it's a platform basically that does [Inaudible: 33:33] and it will say…

Speaker 2: If the PR reduces [Inaudible: 33:41]

Speaker 1: And it just says you [Inaudible: 33:43] you want to look into it. I am not sure if there is a negative impact on the [Inaudible: 33:49]

Speaker 2: I think [Inaudible: 33:53] are useful but I think maybe that's my history of twenty years [Inaudible: 34:02] I think if you are not careful then you tend to optimize for those tools [Inaudible: 34:09] that shouldn't be the goal the goal should be to make something that we [Inaudible: 34:12] and that can be I mean I've seen people in the past completely focusing on test [Inaudible: 34:21]

Speaker 1: [Inaudible: 34:23]

Speaker 2: It's rubbish the only thing is like you said if a particular [Inaudible: 34:27] you will have people in a company who will focus on that who will say okay test coverage is [Inaudible: 34:42] seventy percent I don't even know what kind of coverage [Inaudible: 34:46] I am assuming it's …

Speaker 1: Eighty-nine last time I checked.

Speaker 2: Alright very good. Okay [Inaudible: 34:54] anything higher than 75 I am fine with because like I said [Inaudible: 34:57]

Speaker 1: Which is what people do.

Speaker 2: Yeah exactly. And that's rubbish that's the [Inaudible: 35:10] consequence of [Inaudible: 35:11]

Speaker 1: No so my question again is I am not saying [Inaudible: 35:18]

Speaker 2: No, no I think [Inaudible: 35:20]

Speaker 1: What I am saying is it just gives the reviewer an overview of hey these are the things that you might have to look at.

Speaker 2: [Inaudible: 35:27] if there was a tool I mean yes obviously if there was a tool that could like [Inaudible: 35:33] make it conform yes but the problem I mean obviously it's not our job to do that and we have had to make a choice to… we are always spending a lot of time on building our own infrastructure and this seems like an awful lot of work.

Speaker 1: No I am talking about [Inaudible: 35:53] I am not saying you guys [Inaudible: 35:56] I am actually saying [Inaudible: 36:00] not me personally I know people [Inaudible: 36:02] and there are a lot of other universities which are working on [Inaudible: 36:06] where you have tools like [Inaudible: 36:09] and all that which help you maybe find a reviewer so Microsoft itself internally has its own [Inaudible: 36:14] all it does is review [Inaudible: 36:17] but it doesn't really enable the reviewing process if you know what I mean so it doesn't like so when you are reviewing a piece of code it doesn't tell you that this is the line you need to focus on or this is the change [Inaudible: 36:29]

Speaker 2: Like I said I find GitHub enough [Inaudible: 36:33]

Speaker 1: [Inaudible: 36:35] is that you can get more statistics about just along the lines of [Inaudible: 36:43] or like you know this… there's a new warning that might be in the code [Inaudible: 36:50] so just these basic…

Speaker 2: [Inaudible: 36:51] I think for me once again I focus on simplicity a lot so if there's a big change in GitHub I mean its's [Inaudible: 37:02] this is probably not something we want to do if the change is too much then there should really be a good reason behind it and so I… I mean the tool [Inaudible; 37:24] discussion I think but there is always a risk [Inaudible: 37:28] right now if I look at PR I look at it like I don't even look at the tests really I mean I am sort of [Inaudible: 37:40] maybe writing code for I don't know ten, fifteen, twenty years they know how to write things right the only thing that they don't know [Inaudible: 37:56] I focus on stuff that I [Inaudible: 38:03] assuming a lot right I am assuming [Inaudible: 38:06] so it could be that they took a feature from somebody else right a PR reworked that into something and then [Inaudible: 38:18] like I said most of the time when I do stuff he seems to be happy with it. So I look at stuff like what are you using [Inaudible: 38:37] what's this all about [Inaudible: 38:40] why are you introducing that stuff because we have a lot of stuff in Spring that some people working on Spring don't even know it exists [Inaudible: 38:50] built five years ago but it works it does a similar thing [Inaudible: 39:04] different order so why not introduce that order [Inaudible: 39;08] right that's why we have all these templates [Inaudible: 39:11] they don't have the same API but they have a very similar role [Inaudible: 39:19] used one of those templates before you know that they are… you can allocate them once [Inaudible: 39:23] initialization you have these callback methods but there's a similar pattern there and that's my job consistency, simplicity in terms of [Inaudible: 39:40] when I look at [Inaudible: 39:42] and they introduced you know [Inaudible: 39:54] at all.

Speaker 1: No.

Speaker 2: You know [Inaudible: 39:56]

Speaker 1: Yeah.

Speaker 2: ATOS stands for [Inaudible: 40:00] of application state [Inaudible: 40:13] you know about [Inaudible: 40:19] links right it just means [Inaudible: 40:21]

Speaker 1: Okay.

Speaker 2: So we have a project for that and we have introduced the concept of a link class link entity because you use that a lot when you are creating JSON you want to link to something you might [Inaudible: 40:32] so that was very useful so we said why don't we move that into the [Inaudible: 40:37] we use a lot of portfolio projects [Inaudible: 40:43] for instance we had a [Inaudible: 40:45] module in there [Inaudible: 40:47] part of Spring 2.5 same with this stuff we had this [Inaudible: 40:54] facility [Inaudible: 40:56] you don't have to maintain the links [Inaudible: 41:05] you can just link to other class [Inaudible: 41:06] however this particular library [Inaudible: 41:23]

Speaker 1: I don't eat that healthy.

Speaker 2: So we have this [Inaudible: 41:40] link entity but if you look at the Spring framework [Inaudible: 41:43] framework there's not a single entity there so we [Inaudible: 41:50] yeah it's going to be the first entity class in Spring [Inaudible: 42:05] let's move the link creation logic into Spring 4 but not the entity because that breaks the pattern if you have a particular pattern it requires that you want to be consistent [Inaudible: 42:15] a very simple solution to the ending result was very simple right but I like those kind of solutions where [Inaudible: 42:30] you make a particular choice and all of your troubles all of a sudden disappear then that's effectively a good choice [Inaudible: 42:40]

Speaker 1: The thing is you have all these new features and all of this why was a decision made not to standardize this [Inaudible: 42:53] I mean I know you said you lost the fight on this but.

Speaker 2: [Inaudible: 42:59]

Speaker 1: Do you know why he [Inaudible: 43:03]

Speaker 2: [Inaudible: 43:07] if I look at source code which is not informative in Spring [Inaudible: 43:15] the point about a coding standard is that it's going to be stupid anyway right there is no proper code standard every one of them is going to be arbitrary [Inaudible: 43:34] in some way that's the whole point [Inaudible: 43:36] so I've learned a long time ago that it's really, really pointless to talk about coding standards and back 13 years ago whatever it was there were no tools, hardly any tools [Inaudible: 44:00] so we just picked something [Inaudible: 44:11] I wasn't even working on the framework [Inaudible: 44:15] they just picked something [Inaudible: 44:17] I mean you could say well now we have to change it lets change two spaces but why I mean yeah obviously the tools can give you [Inaudible: 44:33] makes a lot of sense and we have a [Inaudible: 44:41] get ninety nine percent of it right but there are still things like we use two lines between a field and a constructor [Inaudible: 44:54] lines and I haven't found a way to do that in [Inaudible: 44:58]

Speaker 1: [Inaudible: 45:01]

Speaker 2: Sorry.

Speaker 1: You don't use [Inaudible: 45:03]

Speaker 2: We don't use that yet no but I don't think you can to be honest I don't really keep in touch with this stuff I have done but it's there's a lot of people in the [Inaudible: 45:22] these kind of tools than I do

so it could be that we are using stuff that I am typically not aware of because I focus on different things right like I said I focus on the stuff that makes us not having to deprecate things in five years.

Speaker 1: But do you guys use [Inaudible: 45:44] any of these.

Speaker 2: Yeah, we use like I said [Inaudible: 45:50]

Speaker 1: Do you find [Inaudible: 45:59] useful.

Speaker 2: We don't use [Inaudible: 46:01] I find I mean…

Speaker 1: Because I found it to be quite useless.

Speaker 2: Me too. I mean the only… there are always exceptions where you do want to [Inaudible: 46:14] and that's where [Inaudible: 46:18] and yeah I think humans always are more capable of [Inaudible: 46:34] exceptions to the rule than [Inaudible: 46:35]

Speaker 1: [Inaudible: 46:36] So do you know anything about testing [Inaudible: 46:40]

Speaker 2: [Inaudible: 46:42] major feature [Inaudible: 46:52]

Speaker 1: But do you know the [Inaudible: 46:53] process or…

Speaker 2: YOU mean QA testing or…

Speaker 1: [Inaudible: 46:56]

Speaker 2: [Inaudible: 46:58]

Speaker 1: You personally.

Speaker 2: Yeah, yeah. [Inaudible: 47:06] we don't have a separate testing team [Inaudible: 47:09] and I am old fashioned in the sense that I still use [Inaudible: 47:16] J Unit and [Inaudible: 47;17] I don't even know what [Inaudible: 47;21]

Speaker 1: [Inaudible: 47:23]

Speaker 2: Yeah I am personally not a fan of these [Inaudible: 47:28] because I think that's [Inaudible: 47:38] maybe not.

Speaker 1: Would you be interested in [Inaudible: 47:44] told me you were very interested in coming [Inaudible: 47:49] and talking to our students so we have a course going on [Inaudible: 47:55] right now there are lots of open spots for your lectures if you were interested in talking about testing.

Speaker 2: I don't know if I can say too much on testing.

Speaker 1: What would you like to talk about?

Speaker 2: API design.

Speaker 1: Okay so we have a course on software engineering methods [Inaudible: 48:12] September would you be interested in [Inaudible: 48:18] that.

Speaker 2: Yes.

Speaker 1: So you are around [Inaudible: 48:20] right.

Speaker 2: Sorry.

Speaker 1: You are around in the Blinded.

Speaker 2: Yes, I live in Blinded. [Inaudible: 48:26]

Speaker 1: We'd be very happy to have you [Inaudible: 48:30] I think the course is full.

Speaker 2: Yeah that's what he said I mean he asked me to [Inaudible: 48:38] sure what he had in mind it could be the same thing that you have in mind but yeah he said it was full this year so next year yeah hopefully.

Speaker 1: So this is a pretty big course we have about 300 students.

Speaker 2: What year are they in?

Speaker 1: Second year.

Speaker 2: Okay.

Speaker 1: But we will invite [Inaudible: 48:56] I mean it's not every day you get the developer for Spring out there so it would be an open invitation.

Speaker 2: But I would assume that you don't know anything about what we do here [Inaudible: 49:06] what we do is of very little interest for you guys because it's not about computer science I don't do computer science.

Speaker 1: No so we are in the software engineering department.

Speaker 2: That's true yeah.

Speaker 1: So we [Inaudible: 49:32] and there is a huge [Inaudible: 49:35] research in industry there is [Inaudible: 49:38] you know [Inaudible: 49:39] so our aim at least as a group nowadays and I only joined about 5 or 6 months ago [Inaudible: 49:45] PhD is really to get closer to the industry we need to talk to developers what do you guys want you know and what are the pressure points how do you guys do stuff we need to understand that better which is one of the major reasons I am conducting these interviews and which is why it is cool to talk to guys from J unit or you guys from Spring.

Speaker 2: Yeah.

Speaker 1: And it would definitely be a very interesting lecture if you will talk about API design [Inaudible: 50:17] you just told me about backwards compatibility not everyone does this not everyone thinks about the design.

Speaker 2: No especially nowadays with [Inaudible: 50:32] people tend to think that the [Inaudible: 50:40] and people seem to misunderstand the simplest thing as [Inaudible: 50:47] the first thing they'll think of [Inaudible: 50:50] I would even argue the first thing they think of is [Inaudible: 50:55] at all.

Speaker 1: [Inaudible: 50:59] Facebook model right which is [Inaudible: 51:01] as they call it [Inaudible: 51:04] when you start coding stuff [Inaudible: 51:06] to work and after [Inaudible: 51:08]

Speaker 2: But I mean it's really about the product [Inaudible: 51:12] this approach fits our model if you are building a framework you want to keep your user around [Inaudible: 51:23] you have to think about other [Inaudible: 51:25] because that's very important for you we don't want to give anybody any reason to leave us right as a user they might switch to a different technology altogether but we don't want to give them any reason right.

Speaker 1: But that's the reason I asked you about [Inaudible: 51:45]

Speaker 2: Yes, Eric Meyer yeah I know him.

Speaker 1: [Inaudible: 51:50] he is a professor at [Inaudible: 51:53]

Speaker 2: [Inaudible: 51:55] conferences with him and I have spoken to him yeah.

Speaker 1: He is a very nice guy.

Speaker 2: Oh yes.

Speaker 1: And that's the reason I asked you because I know him personally he is in my department [Inaudible: 52:05] so but they have the same philosophy at [Inaudible: 52:16] which is move fast break things [Inaudible: 52:20]

Speaker 2: Well I mean they built…

Speaker 1: And it is a cool framework but…

Speaker 2: Yeah but I mean RS Java I will argue would be very much focused on what's [Inaudible: 52:36]

Speaker 1: [Inaudible: 52:39]

Speaker 2: [Inaudible: 52:41] than what we do I think there is a different sort of [Inaudible: 52:57] then any change you make [Inaudible: 53:10] if you are a library like Guava you have to think about changing something a bit more right if you are Spring [Inaudible: 53:34] we are not going to change anything [Inaudible: 53:37] security need or whatever not in the sense that we don't like it anymore [Inaudible: 53:44] this doesn't work anymore or it's a security issue or something like that that's when we typically introduce breaking changes [Inaudible: 53:55] programming languages right [Inaudible: 53:57] and programming changes they don't… they make a change [Inaudible: 54:12] programming languages [Inaudible: 54:14]

Speaker 1: [Inaudible: 54:22]

Speaker 2: Yeah and I find those I have an extremely large respect for people like [Inaudible: 54:32] you know who can within the confines… from a computer science perspective it's completely amazing to me because [Inaudible: 54:42] from an engineering perspective but he has done that not just [Inaudible: 54:47] as a team but they managed to put this into Java [Inaudible: 54:55]

Speaker 1: [Inaudible: 55:01] the standard.

Speaker 2: Exactly it's I don't know I think that's a very [Inaudible: 55:07] we did an interesting talk about that you might have a look at that [Inaudible: 55:14] so you might want to look that up I found that very

interesting [Inaudible: 55:25] I actually tweeted about it so you can also look at my Twitter [Inaudible: 55:31] and then see but it's a long time ago it could be a year ago even.

Speaker 1: Java [Inaudible: 55:36] because at nine they were presenting a lot of the features [Inaudible: 55:41] new modularity feature coming in and…

Speaker 2: Yeah, yeah but it's not very [Inaudible: 55:51] right so yeah I mean I never really I never learnt computer science anyway so [Inaudible: 56:01] and a master in AI.

Speaker 1: From?

Speaker 2: From university of [56:09] and I actually wanted to get a PhD in AI as well my specialization was machine translation so translating natural languages well I, I don't know I couldn't do both I had a job which I enjoyed I wanted to do a PhD and I had to make a choice and choose for the money. Back then actually [Inaudible: 56:41] back then the funding [Inaudible: 56:43] now it's still terrible but it's better I had a simple job for one day a week and if I would switch to a full time [Inaudible: 56:57] job I would actually get less so this was in 2006.

Speaker 1: Because I had the same choice PhD or [Inaudible: 57:05] I chose to do the PhD because I was like [Inaudible: 57:08] get the money now I will never come back right it's… and I love research to tell you the truth it's very interesting [Inaudible: 57:20] questions.

Speaker 2: I love it too. Maybe one day when I have my pension…

Speaker 1: [Inaudible: 57:36]

Speaker 2: I think summarizing deprecation [Inaudible: 57:52] is really what your goal is like I said we don't want to give [Inaudible: 58:01] any reason to switch frameworks.

Speaker 1: I would assume that that's the case I think with most API developers it's just that not everyone [Inaudible: 53:12] because you have like a patchy [Inaudible: 58:12] collection if you know which has been completely killed because of Guava absolutely killed there's been murder out there [Inaudible: 58:20]

Speaker 2: I think people saying that you want something and actually acting upon it these are two different things and so se say it and we also do it right we put our money where our mouth is and that basically means mostly not doing stuff… mostly [Inaudible: 58:46] our industry tends to have a very… we as developers we tend to be interested in technology for technology's sake and that also shows sometimes [Inaudible: 59:04] suggesting about new features right there is a new framework out there [Inaudible: 59:12] and we are very reluctant to do so because you know if we do so we are depending ourselves on that and so we also find that we are not the first [Inaudible: 59:28] you don't have to be the first to be reactive or to be [Inaudible: 59:33] we don't really care about [Inaudible] the people who bring in our money who pay my paychecks are not the ones who [Inaudible: 59:56] so I don't really care I mean obviously Spring is not really a cool framework yet I know every day [Inaudible: 1:00:05] JavaScript framework so there's that but…

Speaker 1: It is a market leader.

Speaker 2: Yeah, yeah.

Speaker 1: It's not… let's not really push it down in the popularity stats we think it's right up there it's just behind I think [Inaudible: 1:00:23] Guava and J Unit.

Speaker 2: Yeah.

Speaker 1: J Unit is a bit of an anomaly because it's kind of like they captured the market of testing anyway.

Speaker 2: Yeah.

Speaker 1: There is no framework that does it better in Java at least. [Inaudible: 1:00:35] by the way in J Unit 5 it's almost as if [1:00:39]

Speaker 2: Yeah I mean we actually sponsored it one of our team was involved in that J Unit 5 [Inaudible: 1:00:45] it was actually one of the few people [Inaudible: 1:00:50] we still have in the outside working on Spring so we have external people working on Spring [Inaudible: 1:00:57]

Speaker 1: Yeah because I have seen from most of your [Inaudible: 1:01:05] email id.

Speaker 2: Yeah there is one fellow who worked on Spring 5 called Sam Brenan he is actually he used to work for us but [Inaudible: 1:01:16] testing stuff.

Speaker 1: [Inaudible: 1:01:28] where you talk about I think more about how APIs are designed and developed.

Speaker 2: If we do that I want to talk a bit more with [Inaudible: 1:01:37] about the target audience before we do that.

Speaker 1: Sure.

Speaker 2: I want to do it but I also want to make sure that I… if I talk about stuff [Inaudible: 1:01:46] is that going to be [Inaudible: 1:01:49]

Speaker 1: Yes.

Speaker 2: Alright, I just want to know what level they are.

Speaker 1: They are [Inaudible: 1:01:54] programming language course they have done a lot of these courses the other alternative of course is another course [Inaudible: 1:02:05] course but that's a much smaller audience it's about 30 students, 40 students max but we would I think combine both and get Masters, undergrad and our PhD [Inaudible; 1:02:16] and sit through your lecture I think everyone would be interested in hearing what you have to say because everyone knows Spring yeah because as I said it's not a small framework.

Speaker 2: No, no, no.

Speaker 1: Which is for me why it was really cool to have this interview.

Speaker 2: Yeah and surprisingly close to [Inaudible: 1:02:38]

Speaker 1: I didn't… I honestly didn't know that you were [Inaudible: 1:02:42] okay then I can do this interview right because the thing is people from Google and all that are not very open to sitting down to these kind of things or talking to researchers in depth because I did actually [Inaudible: 1:02:56] Guava

they were even willing to make their internal bug tracker public or not public or at least give me access to [Inaudible: 1:03:02] is rubbish it's not the real thing.

Speaker 2: [Inaudible: 1:03:08]

Speaker 1: No exactly Spring is completely open source in that sense at least.

Speaker 2: [Inaudible: 1:03:14] internal problems.

Speaker 1: Sure but [Inaudible: 1:03:17]

Speaker 2: WE have internal issues I mean when a security issue comes up we have to [Inaudible: 1:03:20] public when it releases out right [Inaudible: 1:03:27]

Speaker 1: Yeah but you get generally the tag from your [Inaudible: 1:03:36]

Speaker 2: [Inaudible: 1:03:38] that we are also [Inaudible: 1:03:40] so there is no short channel I cannot walk into your office [Inaudible: 1:03:44] I have to communicate in the same way that everybody else does obviously I can use Skype or video conferencing we are using the same channels as [Inaudible: 1:03:57] I think that really helps the fact that we [Inaudible: 1:04:00] really helps in the openness of the project.

Speaker 1: No it helps us as researchers as well in gaining more insight into Spring and which is why as I said there are a lot of these baffling statistics out there of Spring users and now I actually got a good answer to them because I only had the hypothesis saying that okay it's probably because we don't have representative clients and we don't have and that you guys probably do a great job of designing but now I know after talking to you.

Speaker 2: [Inaudible: 1:04:27] I don't know

Speaker 1: No but I mean there's a more concrete answer from you guys.

Speaker 2: No, no, no it's true I mean it's still an assumption though [Inaudible: 1:04:37]

Speaker 1: [Inaudible: 1:04:38] versus your assumption would be something very different.

Speaker 2: No, no that's true.

Speaker 1: I think your assumption is more someone who is [Inaudible: 1:04:44]

Speaker 2: [Inaudible: 1:04:46] based on the fact that if I go to the conference and people come up to me and say well it's really great and stuff and I [Inaudible: 1:04:52] and like I said it amazes me but it also scares me.

Speaker 1: I don't think it should scare you to tell you the truth when I was talking to Marc Philip of J unit he is like I really want to know how people are using my stuff because they really want to know what features [Inaudible: 1:05:22] they really want to know [Inaudible: 1:05:27]

Speaker 2: I think that really depends on the person.

Speaker 1: I guess so.

Speaker 2: For me [Inaudible: 1:05:40] feature with the most capabilities you know like I said it all falls into place and yeah I have a couple of principles which I try to adhere to [Inaudible: 1:06:04] Spring has a

very different philosophy [Inaudible: 1:06:29] makes choices for a user which I could never [Inaudible: 1:06:32] start thinking about the fact that if you are building a Chinese or Indian or any kind of non-western website you are effectively increasing the textual bandwidth by two at least or three times so that's' a bad default [Inaudible: 1:07:02] increases your bandwidth by two it's not a very good default [Inaudible: 1:07:07] I mentioned this to them and then they said well what should be the default and I said well I don't think [Inaudible: 1:07:12] that's why we don't have a default in Spring that's why I don't think.. and I said well we [Inaudible: 1:07:20] and I think it makes sense I guess it makes sense too but it just shows you that even within Spring we don't agree on these things and that's why I work on the framework and they work on [Inaudible: 1:07:32] different perspective, different way of [Inaudible: 1:07:42] is to make stuff convenient as possible.

Speaker 1: Which it is by the way.

Speaker 2: Yeah it is. I think that I want to expose… I want to confront the user with choices that they have to make and I want to hide the non-choices I think that's a good API where you can focus on the thing that's important and I don't think anybody is going to disagree with me [Inaudible: 1:08:08] going to say well the choice between [Inaudible: 1:08:11] or Chinese or whatever that's not an interesting choice I don't think it is an interesting choice so maybe that's I mean what I am saying is that there's [Inaudible: 1:08:36] different people [Inaudible: 1:08:37] you cannot say my perspective on these kind of things is the only perspective that we have in Spring versus my perspective is [Inaudible: 1:08:44] all the people look at it differently and together that's what makes [Inaudible: 1:08:50] I guess.

Speaker 1: It's very interesting how this open source development really happens you know so you guys in J Unit also [Inaudible: 1:08:57] make these decisions about backward compatibility which I didn't really think that you guys would care about that much because honestly when you look at other frameworks like Guava as I said they don't seem to care about backwards compatibility and whereas you guys really, really seem to and I really like that and appreciate it from an open source community [Inaudible: 1:09:18]

Speaker 2: [Inaudible: 1:09:19] they are a library [Inaudible: 1:09:20]

Speaker 1: [Inaudible: 1:09:23] backwards compatibility is not something that everyone really honestly thinks about it's… so I mean I came in here of course as an interviewer so I had to keep all my preconceived notions out of the door right I just wanted to learn as much as I can from you but to me backwards compatibility is one of the most interesting things because I would say that that's what you should have that's a killer feature of the library in my opinion but not everyone agrees with this.

Speaker 2: It's [Inaudible] feature too.

Speaker 1: Its' a very [Inaudible: 1:09:53] feature.

Speaker 2: [Inaudible: 1:09:58] while we would perhaps not do something there is a lot of things that we work on [Inaudible: 1:10:09] a couple of weeks ago [Inaudible: 1:10:25] reactive string of some sort in our current server based [Inaudible: 1:11:29] it's basically it has to do with marketing [Inaudible: 1:11:45] I can pay for my food.

Speaker 1: No I [Inaudible: 1:11:52]

Speaker 2: Ah there you go I am not so yeah basically let's say that you introduce this feature [Inaudible: 1:12:01] that's going to be a fine argument for other [Inaudible: 1:12:10] so we decided not to do that for

that particular reason so just saying well you cannot do it right you can do it but it's not going to be the way to do it so we are not going to do it yet and that's we make a lot of those choices.

Speaker 1: I mean these are the hard choices that you guys make [Inaudible: 1:12:41] learnt from you today is that there is a lot of [Inaudible: 1:12:46] that goes into your API design and that you actually try to introduce things for the clients benefit at most times without actually taking [Inaudible: 1:12:54]

Speaker 2: [Inaudible: 1:13:00]

Speaker 1: Yeah exactly. So which is I think a very different approach it's not the approach taken as I said by everyone.

Speaker 2: No, no, no.

Speaker 1: And I think this will make for a very interesting lecture [Inaudible: 1:13:19] because I think students need to learn at a young age.

Speaker 2: Yeah right. There's nothing wrong with saying no [Inaudible: 1:13:30]