EPISODE 1138

[INTRODUCTION]

[00:00:00] JM: Salesforce is a platform with a large number of developers, ISVs and companies built on top of it. There's a thriving ecosystem of applications built and managed around Salesforce leading to an important set of relationships and integration points between Salesforce and the other entities involved with the company.

Kevin Poorman works at Salesforce as a developer evangelist helping to strengthen the relationships in the Salesforce ecosystem. Kevin joins the show to talk about Salesforce and the applications that connect to it.

Full disclosure; Salesforce is a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:00:40] JM: Kevin Poorman, welcome to the show.

[00:00:42] KP: Hey, thanks for having me.

[00:00:44] JM: Most people who interact with Salesforce don't think of it as a developer ecosystem. They think of it as a CRM tool, but there is a thriving ecosystem of developers who build things around Salesforce. Tell me a little bit about the developer ecosystem.

[00:01:00] KP: Developer ecosystem for Salesforce is pretty fast. I've seen everything from purely business apps being created to – I just wrote a blog post a few days ago about a guy and a team that created a game on the Salesforce platform. So you can literally go in in just about any direction you want. We, the developer evangelist team at Salesforce, have a group of sample apps that we try to use to demonstrate all the different things you can do. But the ecosystem is so broad that even those sample apps don't fully cover everything.

[00:01:31] JM: Well, tell me a little bit more about what kinds of people are building

applications.

[00:01:36] KP: So it's mostly businesses, because Salesforce is a platform for enterprise level software development. You don't find a lot of people who are, “Oh, I’m going to sit down on Saturday and write an app just for the fun of it.” It’s not like a more open source platform like Node, or Ruby, or something like that where you have an itch, you can scratch it. You can still scratch itches with the Salesforce ecosystem, but it's to say that because of the upfront cost to get it set up, then you're kind of looking at mostly businesses who are doing this. I call it the upfront setup cost, but you can go and sign up for free ort and start developing however you'd like. But again, if you want to get that into production, as it were, then you're going to need to be a Salesforce licensed subscriber.

[00:02:28] JM: What kind of SDKs are therefore plugging to Salesforce?

[00:02:31] KP: Oh, that's a great question. We have a number of SDKs, most of them revolve around dealing with third-party systems. So we have SDKs that allow you to, for instance, build mobile apps. Those SDKs wrap our REST APIs, and authentication help you write mobile apps that are a joy to create, much less painful than making raw REST callouts. And we also have APIs for ingesting all sorts of data. So if you want to drop a record into Salesforce, we can do that with the REST APIs. If you want to drop in 50,000 records, we can do that with the bulk API.

We’ve got all sorts of variations on a theme there. We got something called the composite API that allows you to create multiple object records all related to one another in a single call. So for instance, if you had an address book application, you could create an office location record, and the record contact information for the three people who work in that office location all one in one call. We also have a number of APIs. I guess SDKs is probably the better word for this, where there are toolkits for you to integrate into your own, whatever your application's built-in.

So for instance, my first start with Salesforce way back over 10 years ago was I was working at a company and we had an email application that we were building. And we created an integration in the Salesforce so that one people signed up for free trial, they got created as a lead. When they converted and started becoming a paying customer, they became an account, and support calls were tied in there, that's sort of work.

[00:04:12] JM: Tell me about the most complex application you've seen built upon Salesforce.

[00:04:17] KP: So as it turns out, selling billboards is incredibly complicated. And I never would've imagined this, but I helped write an app that is specifically for a company that marketed and sold billboards and all the variations thereon. So you've got billboards, and bus wraps, and bus station signs, and basically outdoor advertising in general. We also have this thing called a spectacle. And a spectacle was like you could buy time on every surface in Times Square for spectacle. But the process of selling that was way more complicated than I ever realized, because every bit of outdoor advertising space has demographics that get with it.

So there is an industry firm that goes out and says, “Okay, here is an outdoor advertising space, and we’re going to write down everybody who walks by it so we can say, “We are looking for women who are between the height of 5 foot and 6 foot who see this billboard every day.” And then they turn around and say, “Okay, if you want to target women who were 5 foot 5, you want to talk to these – You want to buy these three billboards.”

And so the demographics and the number of billboards just becomes a vast sea of information. So one of the most complex applications that I ever had the joy of working on was an application to help people figure out which billboards to target when building a marketing plan for a client. So they’d get a client in, and the client would say, “Well, I'm about to release our new product, and my target market is teenage drivers who are going off to college with their car to come up with an example.” And we would turn around and would help the sales agents figure out, “Okay, the demographics that apply here are age-related.” So they went looking for 17 and 18-year-olds, maybe 19-year-olds, who have cars. So probably billboards along interstates. And then we would have –They would come up and it would do a very complex search to find the billboards that fit those demographics, and we would turn around and then they could graphically drag and drop those billboards into a marketing plan.

So that system relied on external connections to that industry company that talked about that did the demographics. It relied on vast amounts of data within Salesforce, and it relied on a very complex user interface to make it all usable.

[00:06:58] JM: In that application, what would be the process for building it?

[00:07:04] KP: So the first thing we did is we looked at the data model. And some of the data models we had control over, and some of the data models we had no control over. We couldn't, for instance, change the data that we could get from the industry analyst firm. We could only access it. We didn't truly control it. But at the same time, we controlled all of the marketing plan information. We controlled all the bits about – The metadata about the individual outdoor advertising spaces. And we controlled the information about the actual clients and marketing plans and how they chose the demographics, all that sort of information. So we started off looking at our data model, and we built out as best a data model as we could to start building.

Once we started doing that, we worked on the bits of our code that would be required to aggregate that data and pull together and then look at what we needed to do to make sure that it was fast and efficient, because we were dealing with so much information. We started off with fairly naïve queries in on the order of showing everything from Arizona. And then we had to start narrowing that down and becoming more and more specific. So we got to the point where we had a query set that we could ask when so many plugged in demographic data, we could turn around and say, “Okay, this is how we look for that information in this vast sea of things.”

So a lot of it, we started off on the data side of things. How do we do this? How do we interact with this data in an efficient manner? And once we built that, then we spent a lot of time with the clients understanding how they wanted to use it. How they were currently doing this process of selling outdoor advertising. And we worked with them to come up with a user interface, or at least the idea of a user interface that they could understand and intuitively use. And then the bulk of it was building that user interface.

[00:09:02] JM: Was that application built in-house or was it built by some consultancy?

[00:09:09] KP: A little bit of both. There were definitely in-house developers. I was, for lack of a better term, the architect on the project at the time. And I did work for a consultancy at the time.

[00:09:20] JM: Interesting. Tell me more about the ecosystem of software vendors and software developers around the Salesforce world.

[00:09:29] KP: So in the Salesforce world, we have ISVs, independent software vendors. And they build products, and they put them up for sale in the Salesforce app exchange. And in order to get into the app exchange, there are a lot of work they have to do in terms of passing a security audit and all that sort of stuff. If you can imagine what it's like to get an iOS app approved in the app store, it's like that, but under much more detailed microscope. There is a lot of work that goes into the security review.

And once Salesforce signs off on that, then it goes into the app exchange. And the app exchange is a wonderful little app store for lack of better term that allows you to go and say, "Okay, I'm looking for an app that will help me do X, Y, and Z," and say you wouldn't have to do scheduling. You want to be able to set up a way for your customers to directly schedule appointments with some of your workers. You can look in the app exchange for an app like that. And then you click a button and install it.

Now, those how they ISVs work. Now, you also have in-house developers. So if you are a Salesforce client and our customer and you want to build on top of the platform, and whether that's because you want to build in some automation. Say you want to automatically update contact information whenever the accounts that they belong to has a new address, right? You can automate that kind of work. You can build that out in a couple of different ways. We have what we call low-code tools. And those tools are graphical in nature. So you're picking fields. You're dragging and dropping things into place. You are generally interacting with. It's not training wheels per se, but it is a way of helping you guide your logic and decision-making in a way that makes it much easier to build out automations and things like that with Salesforce.

And then we also have the code interface. So Salesforce as a platform has a domain specific language as it were called Apex, and it's very Java-esque. It's not Java, but it's Java-esque in its syntax. If you write C or one of the C family of languages, you'll probably be familiar with the syntax of Apex. And then also, for our UI layer, we have a JavaScript framework called lightning web components. Lightning web components help you build reusable components, as the name implies, to build out your UI that interacts with that code in the backend and Apex.

So as an internal developer, you might get a story or a request for you to automate some data.

And you could say, “Okay, I’m going to automate that, but I don't really need to write code for it. I can do that in Process Builder or Flow, which are low-code tools. And you could turn around and say, “Okay, now I'm going to –” Our company standard, because we have so much of this automation going on, is to use Apex. I can write that in Apex as well.

And then you have the sort of the third group of people, so not in-house developers, not ISVs. These are the consultancy groups. And consultancies have their own developers who interact and work with end customers, orgs as well.

[00:12:47] JM: You mentioned a variety of different ways of interfacing with the Salesforce data. So you’ve got these low-code tools. You’ve got APIs. You've got SDKs. What's the process of the feedback loop between the developer ecosystem and Salesforce? How does Salesforce render new ways of interfacing with Salesforce data?

[00:13:06] KP: It’s an interesting question. Let me take a stab at answering it. It's actually really surprising to me. When I first started doing Salesforce work, I was surprised that the sort of democratic way in which I could talk to the developers at Salesforce who were building the products. And there are a number of ways that that happens. There are certain people at Salesforce who are sort of well-known in the ecosystem and have very clearly defined developer related roles within Salesforce. And whether that's a developer evangelist or a product manager, almost all of them that I can think of are freely available on Twitter.

So I take Twitter questions, comments and remarks all the time. And I know a lot of other Salesforce employees do as well. So I think that's partly, I think, unique thing about our ecosystem. The other one that I want to call out there is we have a number of places where people can go and ask questions and/or leave feedback. The two big ones are Trailblazer forums, which are forms, obviously. But then we have a developer section there and people can ask questions, post code, ask, “Why did I do this?” Or “Why isn’t this working?” That sort of thing.

But the other one is something we call – Now that I go to talk about, the name is currently escaping me. It's a system we have where we can put up and give our end users the option to help prioritize things. They don't get the final say in prioritizing things, because there might be

like cross team dependencies that required X before Y. But they get the ability to go in and say, “Okay, this feature is really important to me,” and vote on that feature. And that plays a big part in our product roadmaps.

Now, we also have other third-party ecosystem players like Stack Exchange that have a really vibrant community built around them for Salesforce. And even inside those third-party systems, we also have our product managers, our developer evangelists. We’re always in their answering questions, talking about things, raising up issues. We just had one the other day where somebody discovered something that they were trying to do that they thought they could do really easily, only to find out the enums didn’t work the way they thought they did.

And so we've been having a discussion with that developer. What’s your use case? How are you trying to do this? Is this a one in a million use case? Or is this something that would be truly? And then bringing that up to the product owners and the product managers.

[00:15:38] JM: Tell me more about how regularly you are interacting with the Salesforce developer team.

[00:15:45] KP: When you say the Salesforce development team, are you talking about our customers who are Salesforce developers?

[00:15:51] JM: Internal engineering teams that are building the SDKs and the interfaces.

[00:15:55] KP: Yeah. I'd say pretty much every day. Maybe not the same teams every day, but pretty much every day I am in contact with one or more of those teams. Today I’ve had conversations with the mobile SDK team. And I've had conversations with the apex development team. So as a developer evangelists, I have this sort of twofold role here. I'm talking and interfacing with our customers who are using and writing software. And so for that part of my job, I get to build cool things and then demo that to them and then educate people on what we do and what you can do on the platform.

And also, I am integrating and talking to the various product teams. So if the product team launched something or have something in beta, and I'll be working with a client on how the beta

works. And then I'll be able to go back to the product team and say, “It works as you've documented it, but it's not terribly intuitive or user-friendly. If we made this slight change, it might be easier or better.”

Unfortunately, I don't want to have any say in that. I can only pass on that feedback. I wish I had more of a say sometimes. I could say, “No. We have to do this one.” But I definitely get to pass on that feedback on a daily basis.

[00:17:07] JM: Are there any ways in which the Salesforce platform has changed significantly in the last few years where the downstream effects of Salesforce changing has led to downstream effects in the APIs or the SDKs, the developer experience? How have things changed in the last few years?

[00:17:29] KP: Oh, that's a great question. We've got a number of things that have changed recently. I see recently in the past 3, 4 years, maybe even 5 years back. The two biggest ones come to mind is we have an entire Salesforce developer experience tooling change collectively referred to as DX. And the tooling change, what that allows you to do – I mean, it used to be that when you did Salesforce development, everything was based around the Salesforce org in which you were developing.

And so you would have your production org for your company, but you would also have a couple sandboxes, or maybe some developer additional orgs that were connected. And you do the work in those developer additional orgs, the sandboxes, and then you would promote your changes from the developer addition or the sandbox to production.

And there are number of reasons why that is not a great experience. The biggest one being that if you have multiple developers on your team, it's hard to work on the same codebase if you are having – In the situation where he could step on each other and overwrite other people's changes. And so having larger teams work on the same codebase becomes difficult, because you sort of silo the work based on who’s modifying what files, etc.

And so that can lead to several issues. And now with DX, we have the ability to do call source tracking. And so source tracking is your more traditional source base development. So you can

throw your code into Git, or Perforce, or SVN, whatever version control system you are familiar with. And your data and your metadata is all described in files. It can be source tracked in that version control system. And we have the command line that goes along with that. And it's sort of the frontend, this whole DX idea. And that command line gives you the ability to take data that you've checked out from Git. Take that metadata and push it to an org.

And along with DX comes scratch orgs. And scratch orgs are ephemeral. It lasts between 1 and 30 days and they automatically get deleted. And the idea here is you can take a branch of your codebase, say you want to create a new feature. You create a feature branch. You check that out. You create a scratch org. You push your existing code to that scratch org and then you can start working on the data as it is.

Because everything is sourced track, when you merge your feature branch back in, other people can pull that feature branch back into their feature branches, etc. And you can manage it and do CI and bigger teams much more effectively. So I think that's one of the ways in which it's really, really improved. I think we are still working on tweaking it and making it better. But I think that DX has made a huge improvement in that sort of quality of life of the Salesforce developer. The ability to do source tracked rather than an org tracked is big.

The other one that's come about even more recently is something called lighting web components. I mentioned them a little bit earlier. They are the idea that you have the ability to create a UI element in a web component and then reuse that UI throughout your org. And lighting web components come with a group of other things, for lack of a better way of putting it. They come in a bundle. So you have the idea that lighting web component also has access to the UI API, which is a newer API for doing UI-based interactions. And you have the UI for – Excuse me. The API for grabbing record data. And you have a whole suite of what we call base components that make it so much faster and easier to spin it up. You don't have to go and hardcode all the elements in logic of a button. You can just call lighting:button, or lightning-button, I'm sorry.

And so those two things here in the past five-ish years have really taken root and changed really fundamentally the way Salesforce developers can work. Now, that not to say everyone's doing them. I know a lot of Salesforce developers who still do order-based development, but they

have the ability to do that source tracked development.

[00:21:36] JM: Could you tell me more about the lightning web components?

[00:21:41] KP: Yeah. Lightning web components are an extension to the web components API. It used to be back a few years ago before web components were really a thing. That any sort of UI framework had to handle not only all of the UI bits and pieces, but also had to figure out how to talk to the host and get data from the backend services, etc. And when you talk about doing that in sort of a context of Salesforce over an enterprise software platform, you got to worry about and handle security and who's got access to what, and all that sort of jazz.

And now with lighting web components, what we've been able to do is to take proposed extensions to the actual web components standard and use them for taking care of those like communicating with the backend, etc. So on top of web components, we have a few what we call decorators. And those decorators handle things like disk talking to the backend. So you can turn around and do an at wire call to a backend method. And that's how you can extract or save or insert data in the backend from within your lighting web component.

We also have one that another decorator we have that exposes a property within the JavaScript class that is the controller to the HTML DOM. So we of the API decorator, and that allows you to say when you pass in a DOM parameter, you can give it a value there inside the HTML. And that'll be available to the JavaScript controller. All this is based on open standards. So we've got the ES6 and above standards going on. So you've got classes and imports. And you can import backend functions, which is how you can wire to them. You can also import other components, CSS, that sort of thing. And the lightning web components have a really pleasant – I was really surprised as I was being onboard with lighting web components. I was really surprised at how pleasant they were to work with. And now I've got history in Angular, and I know enough React to hurt myself. But when I was working with lighting web components for the first time, I was really, really enjoying it. It's really quite a breeze and a joy to build out. And they have a fairly intuitive syntax. There are a couple places where they didn't quite make sense to me immediately, especially around inter-component communication. But once I worked out those kinks, I really enjoy them.

[00:24:21] JM: Are there integrations between Salesforce and Herroku that are worth discussing? Like, Heroku, obviously for those who don't know is this acquisition, Salesforce-made, one of my favorite hosting platforms. Herroku is amazing. At the time of the acquisition, it didn't make a whole lot of sense. Over time – I mean, it made a lot of sense in the sense that it was great asset, but the synergies between Heroku and Salesforce were sort of undetermined. I'm wondering if those synergies have matured over time.

[00:24:52] KP: Yeah. There're a lot going on there. So I want to expand a little bit about what Heroku is. Heroku is a platform as a service where you can write your code in pretty much any language. I say pretty much. I don't know that I've seen a COBOL build pack for Heroku. But if you are working with PHP, Ruby, Python, I think there's even a C# and an Elixir build pack. And your code, you write your code for your application and you push it using Git to Heroku. And Heroku will turn around and look at your build pack instructions and then build out a fully functional – They call it a slug. And that slug then provide you the ability to execute your code in the cloud there. So you end up having – It's great for web applications like Ruby on Rails You can turn around and fire up a Ruby on Rails app in a minute and a half from the time you push it to the time it's up and running. Heroku is one of my favorite web properties anywhere, I think.

But thing that makes Heroku interesting, especially now, is that we have a way for you to take your Salesforce data and expose that to your Heroku app and vice versa. And so we have this thing called Heroku Connect. And when you are there in Salesforce, you can say, "Okay, I want to share these objects." And then Heroku Connect takes over, and those are shared out to a Heroku PostgreS database.

So we give you the ability to say, "I want to sync these objects, these fields on these objects and go." And it's not quite instantaneous. But it's pretty quick. And what happens is that allows you to write code in any language and that you can run on Heroku, and then have that interact with your Salesforce data.

So if you are just getting started with Salesforce andyour team doesn't know anything about Apex and they've never written anything but, let's say, Python. They could turn around and go in there and set up a Heroku Connect and then interact with the PostgreS database through Python. And the manipulations you make to the data inside that PostgreS database will be

mirrored back into Salesforce and vice versa.

So that sort of little logical bridge between the Salesforce data and your Heroku instance really means that you can write all sorts of very powerful integrations without having to worry about how to handle record syncing and that sort of work. So there's so much you can do there. It's almost hard to come up with a good example, because there's literally no bad examples. But when you're talking about using Heroku Connect, you can think about doing it in a number of ways. Let's say you've got a client-facing application on a mobile device and you want to be able to set it up so that your mobile device users can create questions, can create cases, for instance, support cases. One way to do that would be to set up a community. And then you could build the Salesforce community into your mobile app, etc. But if you don't want to deal without that UI, or you don't want to deal with that SDK, what you can do is expose the ability for your users to write that PostgreS database in a way that then gets translated back securely and safely into the Salesforce database.

[00:28:19] JM: Do you spend a lot of time building toy applications yourself just so you can familiarize yourself more and more with the platform?

[00:28:27] KP: Toy applications. Yeah. I tend to think of them as more like proofs of concept. I just finished one a little while ago that exports data from my phone's Health Kit application and then drops it into Salesforce so I can do some machine learning and some sort of visualizations of that data. I am a type II diabetic and I have a system that will grab my blood glucose every five minutes. And that get stored in Health Kit. And so I wrote a little iOS app that then turns around and exports that into Salesforce as a way for me to go in there and start correlating what I eat with what effect it has on my blood sugar.

Now, I don't know that that's ever going to be a great use case for the Salesforce platform at an enterprise level, but I do know that I can demonstrate to developers how to do that remote integration with our SDK from an iOS app into a Salesforce org and then turn around and demonstrate what it looks like to do some lightweight machine learning. I say lightweight, because machine learning is a net new area of interest for me. So I'm experimenting with it. But my hope is to be able to say if I go into my phone app and I say, "I'm going to go eat pizza for lunch." That it could come back and tell me, "Well, your blood glucose is currently X. And if you

eat two slices of pizza, it will probably be Y." It's given me a way to sort of budget, for lack of better way of putting it, my blood glucose.

[00:30:03] JM: Tell me more about the opportunities for machine learning in the Salesforce platform.

[00:30:09] KP: Yeah. So they fall into the – I guess of Einstein. We've got a few Einstein offerings. We've got some OCR ability. So you can upload an image and have it processed using Einstein to come back and give you the text or to identify like – I thinks this is probably a birthday, for instance. The classic example that we've used is like if you'd scanned a driver's license. It would be able to show you, "We think this is the birthdate." "We think this is the picture of the person." "We think this is her address," and that sort of thing. And the use case there is actually if you're letting people or visitors into your building and it's a secure facility, being able to take a picture of their driver's license in ordered to figure out who's there is actually pretty useful. So we don't have to worry about typos with names and that sort of thing.

We also have the ability to – There's an Einstein set up for sort of identifying trends in your data. And I know that's kind of the definition of machine learning. But I have stated that generically, because when you pass something into the Einstein prediction builder, you're actually saying, "Here's a massive data, and I want to look at it and know in this dataset what it looks like for someone to – For an opportunity to be more likely to be closed one or to be closed lost. And it can highlight and can learn some of those differences and illustrate them to you.

We used to do Einstein object detection. That was a fun one. I built one of those toy apps to try and determine given a series of webcam images I took my back door. I demonstrated that I could identify fairly concretely which of my dogs was leaving the back door. Just as a way of like doing object detection and sort of color detection, I could determine which dog was which.

Yeah. So machine learning, you can do an awful lot with it, but some of it you're going to have to train yourself. But are sort of our low-code offerings for Einstein, our Einstein prediction builder, and doing things with Einstein OCR.

[00:32:24] JM: how do you see the Salesforce platform evolving in the near future?

[00:32:28] KP: Wow! Okay, evolving in the future. I think the things that people count on are going away. We're not getting rid of lightning web components. We're not getting rid of anything like that. I think more and more people are going to be doing work with DX. And part of that is we've announced that we've got this thing called code builder, which is sort of a next-generation IDE running in your web browser. So you can log into your org, hit a button and launch your IDE.

And I think that's really going to change the way people interact and develop with Salesforce, because that is built on top of the presumption of DX. So I think that's a big one. I think lightning web components are going to continue to evolve. They've only been out for a couple years now. But I think they're going to continue to evolve in positive ways, especially in the sense of more and more base components, what we call base components, which are those, "Here's a button. Here's a table that sort of work."

Yeah. And I think the other big one is we announced at Dream Force last year Salesforce Functions. It's been compared to a lot of things, but it is a runtime for writing functions as a service style applications. So you can write a function and then have it go do something and then return the data to you or not. One of the classic use cases I've seen is you've got a sale going on in your Salesforce org, and that sale is to consumer. A consumer then turns around and receives a ticket to go to an event, for instance. And you can have the actual ticket generation process be written as a function that runs on demand and scales so that from 9:05 to 9:15 when you just open up the sales for a new rock concert, you don't have to worry about scaling up and provisioning a bunch of VM's. This will scale for you.

So I thank functions is also going to be pretty important. I think it will bring about a new way of thinking about developments. And we've heard a lot about microservices and sort of the steps along the path to functions. But I don't know that a lot of Salesforce developers of really thought through the differences between writing it as a function versus writing it as something that happens on, say, Heroku, or in a third-party call out to a custom-built system.

[00:35:04] JM: Are there some specific frictions in the platform that you recognize today that you wish were a little smoother?

[00:35:12] KP: I think my biggest friction has always been the or-based development, and which is why I think that source-based development or source tracking is so important. I know we've got other friction points among our developer ecosystem. And a lot of developers seem to think that we have what are called governor limits. Governor limits exist to help make sure that a bad actor or a bug in someone's code can't use up all the system resources for a particular instance of Salesforce. Because we are multitenant, your code runs next to my code in the same sort of resource pool. And so governor limits help prevent your code from taking over all the resources at once. Be very generic about it.

And I know a lot of developers, especially new developers, find those to be frustrating. I found that they are actually – For me, personally, I find that they're actually a way to help me write better code, because if I can't get something done within a governor limit, then I'm probably not doing it in an optimal or a safe way. I actually don't find those developer limits to be terribly big friction point. But I do know that some do.

Other friction points are a lot of – We have made a lot of changes over the past 4 or 5 years to our tooling. And so there is sort of this consistent learning curve that has to keep going on as we make DX more feature-rich, etc. But again, I don't know that that's a friction point.

[00:36:54] JM: You've mentioned the Salesforce, basically the app store, in some detail. Tell me more. If I was to browse through that app store, what kind of apps would I see?

[00:37:05] KP: So you generally find apps that would help enterprise businesses be more efficient or apps that meet a certain feature set that Salesforce doesn't naturally provide. So, for instance, one of my favorite apps on the app store deals with helping organize tasks and task lists and making sure that there is a nice consistent UI for making sure these tasks are done. You might also look in there and find – And that sort of like make it more efficient sense.

Then you also find apps that the give features or entire applications of suites of features that you don't find in Salesforce natively. So one of the big examples of that is an app that lets you do sort of content generation, content document generation with merging in field data from your Salesforce org to generate, say, a sales proposal document. And it's not just documents. You

can do spreadsheets and that sort of thing.

So that's a big one, a big area where you can do things like generate a PDF in Salesforce natively. But if you want to do your complex mail merges or that's sort of work, looking at the app store for an example of an app to do that is a great way to save a lot of time and money to get that done sooner. You can also find – There is an app for a little while. I'm not entirely sure if it's still there. There was an app for a little while that helped you document your Salesforce org, which is great if you've got a large team being able to help document and say, "Okay, this particular function of code is used X, Y, and Z. And we can't touch it because of A, B and C." So that was a great app too.

[00:38:47] JM: I know there are entire companies built on top of Salesforce. Like we did an interview with AeopleAI one time. What's the difference between people who build entire companies on top of Salesforce versus people who build like small apps on top of Salesforce?

[00:39:02] KP: That's a good question. I know some examples of both, and I think that the companies that build their entire application on top of Salesforce, they do so because Salesforce provides a unique platform that meets a lot of their goals out-of-the-box. And so for them, it's a faster to market type situation. I think sort of ISVs can sell their products in sort of an almost white labeled way. So they're getting Salesforce licenses, but you're seeing only the product being purchased.

And then the other way to do it is you write a package or an app for Salesforce that you turnaround and people can install in their already existing org. And so with those two things, that's the big distinction between those who their entire app is based on top of Salesforce. They tend to sell their app as sort of the white labeled Salesforce license. And then the other option is more generic ISVs who they may only write software for Salesforce, but it's a bit of software that runs inside Salesforce that has a number of use cases.

So, for instance, again, that document generation platform. It doesn't matter what industry you're in You may have a need to do complex mail merges inside of it and to generate documents. The flipside to that is the sort of like their entire business model is based on it is a risk management company. And they help generate plans for companies to survive hurricanes

and tornadoes and natural disasters, continuity business plans. And they offer their product as sort of that it runs in Salesforce, but it looks like the risk management company and it works entirely based on that.

So those are the two sides, two different ways of doing it. When you write an app that you're going to sort of white label, what you're doing is you're saying, "We've got a singular use case that we're going to solve for. Not more generic use case. So you've got a risk management plan app. And it doesn't matter whether or not the customer, the risk management plan uses Salesforce or uses other ERP systems, other CRM products, other platforms. You can turn around and still use this risk management platform that runs inside the Salesforce platform. But if you've got that generic, "I need to create documents." It doesn't matter what business you're in. Whether it's risk-management or billboard selling, you still need to be able to generate that PDF of that document and send it out.

[00:41:43] JM: You mentioned some about the process of submitting something to the app store. And I'd like to know more about the security review process. What are the common security holes or security vulnerabilities that need to get cleared up?

[00:42:00] KP: Oh! That's a great question. There are lots. I'm not going to be able to exhaustively talk about it. But the security review process goes through and looks for, in general, a couple of red flags. So one is make sure that it honors the best practices for security and data privacy. So imagine an app that comes in and the app writers are early on the learning curve for the Salesforce platform, but they get their app in there. And what it does is it goes and it makes a query, but it's not honoring the user permissions. So people can run this query and it doesn't matter whether or not I can see a record as the end user. This app is still going to show me that record. And that would be a security issue that would have to be fixed before it go into the app exchange.

So we're were looking for – And I say we. I'm not on that app exchange security review team. I've interacted with them. I have great respect for them. They are sharp as a crack, sharp as a whip. But they're looking for mostly security related things. Are you making authenticated callouts to a trustable third-party service? Does that third-party service – Do you control that? Or is it something that might be hijacked and then we would have an issue with data coming back

in?

So they’re looking at all aspects of the security of the code and the app that you've written. There're all sorts of things related to. Looking at choices in Apex of whether or not you’re using with sharing or without sharing, that sort of thing.

[00:43:35] JM: If there's someone out there that is just getting started and wants to figure out how to build on Salesforce, what would you recommend to them to get started?

[00:43:44] KP: So we have a thing called Trailhead. Trailhead is the fun way to learn Salesforce. And you can go to the trailhead.salesforce.com and there you can learn, I'm not going to say everything, but you can learn pretty much everything about software development on the Salesforce platform. You’d also find information on how to use Salesforce if you are, say, a new employee at a company that uses Salesforce. But there is a developer trail there, and that developer trail will walk you through with a few modules or projects on how to get started with developing for Salesforce, and whether that's using Flow or Process Builder, whether that's writing Apex code or triggers. There are modules for all of those things there on Trailhead.

They’re generally not terribly long. You generally sit down and do a badge over lunch. Some of them take longer, of course. Some of them are super short, but on general, you could sit down and do one or two over lunch. And then when you get done with that, the thing that's interesting about Trailhead is because it's inside of an org and you’re doing work inside of org, we can actually use our own APIs to login to that org and check your work.

So if you are going to do a badge on writing flows, and we’re going to tell you and walk you through how to build a flow that does X, Y, and Z. At the end of that, we can check to make sure you have a flow that is named what we told you to name it that does X, Y, and Z. And so you end up – When you you've done these badges, you've done more than just sat down and read something and answered a quiz. Some of our badges just have quizzes, but a lot of the developer ones also have what we call hands-on challenges. And those hands-on challenges are checked in the background to make sure that you've done it right.

So it's not just I read something and I think I understand it. Let’s say I've read something and

then I did it, and I know I did it correctly because they checked my work. And that's a huge way of learning the Salesforce ecosystem. Now, you can sit down and start doing that for free right now and just sit down and login and start taking those, those badges on the trails. And you can get plugged into the ecosystem by looking at the blog, developer.salesforce.com/blog.

Yeah. There're all sorts of ways. And in some degree, it's sort of overwhelming the number of ways you can start getting plugged in. So I recommend going to Trailhead. And then if you want to do some development work, look at the developer badges, look at the developer trail and start work that way.

[00:46:21] JM: What else would you like to add about the Salesforce platform and the developer ecosystem on top of it?

[00:46:27] KP: Salesforce is a great platform and a great ecosystem to have a career in. I really believe that. I have friends of mine who I went to college with who are game developers who are developers from mobile platforms. And to some degree, the work-life balance we have between them is radically different. My friend who's a game developer works crazy hours. And my Salesforce developer friends, they tend to work business hours. Salesforce is an enterprise business platform. And so that gives you a sense of what kind of work you would be doing, which means that you don't have to necessarily worry about, "Am I going to have to be up till four in the morning for three weeks to meet a deadline?

That's not to say it doesn't happen. But I think it's less likely. We'll put it that way. It's a great ecosystem. One of the things that it's really struck me about the ecosystem for as long as I've been in it is the friendliness and willingness to help. So I know there are opportunities for people to ask questions all over the Internet and then you often get a wide range of answers. What I'd don't typically get in the Salesforce ecosystem is those answers that are like, "That's a dumb question." Or, "Haven't you read the instructions? Generally, the response from ecosystem that I've experienced is generally been, "Well, we're going to assume you've read the instructions, but something didn't quite make sense. Let's walk through it together." And it's just been a much more welcoming and friendly environment. I understand not everyone has that experience. But that's been my experience. And I think, again, yeah, come on over. We've got milk and cookies.

[00:48:16] JM: well, Kevin, thank you so much for coming on the show. It's been pleasure talking to you.

[00:48:19] KP: Thanks Jeff. It's been great being here.

[END]