

Interview with JavaFX Coding Challenge Grand Prize Winner Sten Anderson

[Reviews Interactive](#) recently sat down with [Sten Anderson](#), developer of the grand prize winning [JavaFX Coding Challenge](#) application, [Music Explorer FX](#). Sten has been working with Java since the late '90s and is currently a Senior Consultant for the software consultancy, Citytech, in Chicago. The complete interview with Sten follows. A shorter version and a podcast can be found [here](#).

Reviews Interactive: How long have you been using JavaFX?

Sten Anderson: I've been using JavaFX since the preview release last August (2008).

RI: How did you learn JavaFX?

SA: I learned mostly through the online resources and fellow bloggers' experiences, as well as a healthy dose of trial and error. However, I found it fairly easy to get up to speed in the language, which is more of a testament to the language design than it is my ability to learn new things.

RI: What was your inspiration behind developing this application?

SA: One of my areas of interest as a developer is application usability. I wanted to experiment with creating a maximally usable, highly visual, application that people of all sorts of technology backgrounds would be able to use. Most people respond to music, and since discovering new music is a perpetual background task for many people, it seemed like a good fit for the application.

I knew that the music 2.0 startup, The Echo Nest, was doing some pretty innovative things around, among other things, exploring the "recommendation space" of a musical artist. Since they exposed this information via public web services, it seemed pretty natural to build the application around their API.

RI: What aspect of JavaFX did you, as a developer, find most useful in creating your application?

SA: By far the most useful aspect of JavaFX, from my point of view as a Java developer, is its near-seamless integration with the Java language and platform. I would not have been able to write the application in the same time-frame without being able to lean on my existing Java knowledge.

RI: How does JavaFX make your application easier to use for the end-user?

SA: JavaFX excels at creating rich, interactive, visual user interfaces which I think are ultimately easier to use, by a broader range of users anyway, than say, well, applications that aren't. The longer I do this, the more I'm convinced that it's all about end-user experience. If an application isn't perceived as usable, users will not use it, or if they have to use it, they won't like it, and it will affect them negatively on all sorts of conscious and subconscious levels.

JavaFX is by no means a "silver bullet" of usability, but it's definitely pointed in the right direction. I've

heard JavaFX described as a sort of “DSL (Domain Specific Language) for creating user interfaces.” I think that’s an apt description, and I would also go farther with it to say that it’s a DSL for creating *compelling* user interfaces. That is, JavaFX makes it easier for a developer to create something good.

It does this through the language itself, via its declarative syntax, which means that the code and final product share the same hierarchy, and also through its increasingly rich API. For example, since the new components and layout managers are designed with each other in mind, it makes it more difficult for me to create something that looks bad. Or I suppose I should say, I have to go more out of my way to create something bad.

RI: As a developer, what do you like most about JavaFX?

SA: As a Java developer, I found the syntax of JavaFX a welcome respite from the more verbose Java language. Elements of JavaFX have a sort of “best of breed” of the language features that are currently in vogue such as the Type Inference of Scala, and the declarative “builder-like” syntax of Groovy builders. I actually find it very natural to switch between all of these languages. Long live the JVM!

RI: Do you have plans for creating future applications with JavaFX?

SA: Some of us at Citytech are informally working on a JavaFX/Seam (the JBoss web framework) project which we may show at JBoss World here in Chicago the first week of September.

RI: Have you tried out JavaFX 1.2?

SA: I’ve just started the upgrade process to JavaFX 1.2, so I can’t say much about it yet. I am eager to try out some of the new features though.

RI: If you could add one feature to JavaFX right now, what would that be?

SA: My guilty pleasure answer would be a full 3D scene graph, although I realize that’s of limited use for business type applications.

My more utilitarian answer would be some better threading support, or rather a better threading abstraction. I know JavaFX 1.2 has made some concessions to this via the FutureTask business, but you still have to ultimately dip down into Java to use it which seems like a bit of a leaky abstraction to me. So it would be nice to be able to write all of our asynchronous operations in pure JavaFX.

I can understand the rationale behind wanting to keep JavaFX stalwartly single threaded, but if I may cite the popular axiom, “Bad developers will move heaven and earth to do the wrong thing” – if we can’t do it in JavaFX, we’ll just end up making a bigger mess farther down the pipeline in Java.

RI: How do you think JavaFX has changed or will change the way developers create RIA applications?

SA: I think it has widened the playing field by giving Java developers something that can utilize their

existing skill set. So whereas before Java developers may have had to mingle with Flash or Flex, now they have the option of staying “closer to home” and not leaving the Java Ecosystem.

RI: How do you think JavaFX compares to other RIA development programs? In what ways is JavaFX better?

SA: I don't have any experience with non-Java RIA platforms, so I can't comment on how it compares to them. I will say though – and I think this really is JavaFX's winning differentiator – since JavaFX plays so well with Java, its newness is compensated by the extreme age and maturity of the core Java platform. So, put another way, for a Java developer, JavaFX is “better” simply because it's Java.

JavaFX of course isn't the only RIA option in the world of Java (the Groovy project, Griffon, deserves a special mention), but it's certainly one of the strongest. I've been very impressed with how often the JavaFX team has cranked out new releases. I think it's been a strong initial year for JavaFX. Hopefully the momentum will continue.

###