Ansichten zu JavaFX

von Alexander Casall, 11.02. 2016

(siehe https://jaxenter.de/javafx-anwendungen-aus-der-praxis-so-schoen-kann-ein-java-ui-sein-34877?utm_source=jaxenter.de&utm_medium=referral&utm_campaign=seealso)

Auf der JavaOne im November 2015 haben Dirk Lemmermann (Freelancer) und ich (Alexander Casall) eine Session zu Real-World-Anwendungen mit JavaFX gegeben. Wir haben einige Apps vorgestellt, die wir entweder selbst für unsere Kunden entwickelt haben, oder an deren Entwicklung wir zumindest beteiligt waren. In diesem Artikel fassen wir unseren Vortrag zusammen und zeigen die vorgestellten Anwendungen. Zusätzlich habe ich einige andere JavaFX-Entwickler gebeten, ihre Apps hier kurz zu präsentieren.

Insgesamt haben wir 20 JavaFX-Anwendungen aus der echten Praxis zusammengetragen, die wir Ihnen in einer vierteiligen Serie vorstellen wollen.Vielen Dank an Dirk Lemmermann, Jan Gassen, Rob Terpilowski, Sean Phillips, Angelica Leite, Denny Israel und Manuel Mauky für die Informationen zu ihren JavaFX-Anwendungen. Bevor wir uns über die Anwendungen unterhalten haben, habe ich meinen Gesprächspartnern einige Fragen zu JavaFX gestellt:

1. Nach den praktischen Erfahrungen mit deiner JavaFX-Anwendung:
   Was sind für dich die Highlights von JavaFX?
2. Was denkst du im Allgemeinen über JavaFX?
3. JavaFX, Swing, SWT, HTML5 – wer macht das Rennen?
   Oder noch besser: Was sollte wann verwendet werden?
4. Wie zufrieden bist du mit der Arbeit von Oracle an JavaFX?
5. Was vermisst du bei der Arbeit mit JavaFX?

Interview with Dirk Lemmermann

Can you tell us about the highlights when you used JavaFX?

The CSS support, the property bindings, and the nice separation of controls and skins are my personnel highlights when using JavaFX. CSS and property bindings are a fantastic boost for productivity and the usage of skins makes sure there is a nice separation of concerns and you end up with clean code.

What is your general opinion about JavaFX?

I know that JavaFX had a really tough start with versions 1 and 2. When I was first asked by a customer of mine to migrate my FlexGantt framework from Swing to JavaFX I hesitated. The only reason I eventually did code in JavaFX was the fact that my customer was willing to pay for the work (as opposed to just the license). Luckily version 8 became available at that time (in 2013) and my initial resistance quickly faded away. Version 8 was a pleasure to work with and has steadily improved over time. I know that there are still many aspects in JavaFX that could be and must be improved but for me, with the desktop as the target platform, JavaFX is a big step forward compared to Swing and switching from Swing to JavaFX must be a no-brainer for most development teams.

JavaFX, Swing, SWT, HTML5 – Who wins – or better, when to use what?

I believe that SWT and Swing are excellent and very mature frameworks but that they have reached their end of life cycle. This does not mean that people will stop using them from one day to another. There will still be a lot of work for years to come that will be based on them. Existing Swing and SWT projects need to be maintained. Maintenance might mean bug fixing or adding new features. I still sell framework licenses for Swing every once in a while. So the remaining competitors for new projects are JavaFX and HTML5 and they both have their pros and cons, always relative to the project for which they are being evaluated. There are too many criteria to mention here, but if your application is an intranet client / server solution with a lot of data, limited number of users (in the hundreds), complex logic, and high interaction then JavaFX is the perfect choice as it allows you to implement the entire solution with a single lan-

guage / technology that is also independent of any browser. JavaFX was not invented to replace Websites but to provide a high quality platform to support complex processes in medium and large enterprises.

How satisfied are you with the work of Oracle on JavaFX ?

In general I am very satisfied but there are sometimes areas where you think that the solution is only 90% complete. The TableView is probably the most dominant example for this. There is even internal code and comments in the TableView skin that hint that the missing 10% were very well on the radar of the Oracle engineers but that they just didn't have enough time and / or resources to implement them all (e.g. fixed columns / rows).

What do you miss in the work with JavaFX?

Sometimes I have to work too much with skins to implement a feature that I would expect to be exposed on the controls themselves.

Interview with Jan Gassen

Can you tell us about the highlights when you used JavaFX?

There were no real "highlights" when working with JavaFX. I however pretty much enjoyed debugging the GUI with ScenicViewer. It really simplifies things when you can inspect elements in the GUI similar to using the web inspector in a browser.

What is your general opinion about JavaFX?

I always liked working with JavaFX. It's pretty easy to get started with and many things can be implemented with ease.

JavaFX, Swing, SWT, HTML5 – Who wins – or better, when to use what?

We are a team of Java developers, so choosing a Java based framework for creating the CenterDevice desktop client was obvious. With JavaFX and FXML it is very convenient to create graphical UIs and connect elements with code. Since we only use custom-designed controls for our desktop client, having a bit more native look and feel as provided by SWT wasn't that important to us.

How satisfied are you with the work of Oracle on JavaFX?

I can currently only think of only one thing where we had something to moan about. Over different versions, JavaFX had issues supporting multiple displays with different pixel densities. When moving a window from one display to another, the size of its content doubled for the JavaFX version we started with. At a later version, the behavior seemed to be correct. Unfortunately with yet a later version, the size of the content was only half the original size when moving the window from one screen to another.

What do you miss in the work with JavaFX?

We particularly missed a better integration into the underlying OS. To create a custom tray menu, we had to fall back to AWT components, which turned out to be rather time-consuming. On OS X, it was hard the replace the package name in the automatically created application menu. Another, not necessarily JavaFX related, issue was the lack of support for automatically configured proxies. In this case, it required quite some effort to get the proxy settings from the OS.

Interview with Rob Terpilowski

Can you tell us about the highlights when you used JavaFX?

The animated transitions and effects such as blurring or drop shadows make a huge difference in the user experience of the application when implemented properly. These are small details that sometimes get glossed over, but when introduced to an application can create a very polished UI. The effects and transitions were something that were possible to do with Swing, but it was so painful. I don't remember how many times I had to override the paintComponent() method to customize the look of a particular component, but all of this is baked into the JavaFX framework, allowing you to do these things in literally a few lines of code.

What is your general opinion about JavaFX?

Overall I am pleased with JavaFX as a successor to Swing. The addition of property bindings, which eliminate the need for event listeners in many circumstances helps cut down on some of the code complexity. I also like the fact that there is a very clear seperation between the model, view, and controller, where the view is FXML, the model can be a collection of JavaFX properties, and the controller utilizes dependency injection to have the UI components passed in. There are some nice tools for doing JavaFX development, including NetBeans for coding, SceneBuilder as a WYSIWYG design tool and ScenicView to help visual provide information about the application while it is running.

JavaFX, Swing, SWT, HTML5 – Who wins – or better, when to use what?

For a new application I would not consider Swing or SWT, which leaves either JavaFX or HTML5 as the remaining options. In this case there is not a clear winner, but a set of tradeoffs one needs to consider when making a decision. With HTML5 you have the advantage of being able to deploy your application across many different platforms (phones, tablets, and Desktops), as well as multiple OSs (Windows, Mac, Linux). There is also the benefit of a huge development community and large selection of open source tools and frameworks. The ease of deployment across platforms comes at a cost however, in that you must operate within the constraints that are placed on you by the browser. The amount of time debugging issues across different browsers or OSs is often overlooked or underestimated by teams when deciding whether or not to go the desktop or web app route. We recently worked on a project where a very large chunk of time had been consumed in order to get a piece of functionality working correctly in IE 9 on Windows. With JavaFX the drawback is that the user has to download and install something to their desktop, which is becoming very old fashioned. But if this is not an issue, then you are free to develop outside the constraints of the browser and use the full power of the Java language and the ecosystem that backs it. For applications that are used internally within the company I feel that it makes a lot of sense to deploy these at desktop applications for this reason. Deployments are not an issue in this case as we can automatically push out new installations or updates to PCs in our network automatically. We also bundle a private JRE with the application so we don't need to worry about which version(s) of Java the user has installed on their PC.

How satisfied are you with the work of Oracle on JavaFX?

Jonathan Giles and his team have been doing great work at Oracle adding improving and enhancing the JavaFX libraries. That being said, it would be nice if Oracle officially stated what their long term plans are with JavaFX. When Oracle let go of some of their evangelists (who were big proponents of JavaFX), just before JavaOne it started a rumor mill of what may have been behind the move. The uncertainty this has created, and lack of official communication from Oracle will likely deter some development teams who may be on the fence about whether they should port legacy Swing application to JavaFX or HTML5. Over time this will potentially affect how large the JavaFX community eventually becomes.

What do you miss in the work with JavaFX?

The amount of 3rd party component libraries (both open source and commercial) that are available for JavaFX is still somewhat limited at this point, but that should change as the JavaFX community continues to grow.

Interview with Sean Phillips

Can you tell us about the highlights when you used JavaFX?

Animation, Easy Layering/Transparency, Object Grouping and Performance. These are the things I love about JavaFX and you will notice these are all the things that were hard(ish) to do in Swing.

What is your general opinion about JavaFX?

I love it… it really frees my imagination when designing interfaces. I starting learning Java GUI development when Swing was young. Swing was great at the time but it forced you into certain design patterns. Step outside those patterns and you would have a difficult time of things. JavaFX provides all the capabilities you would need to build almost any custom interface with far less restrictions, smaller code footprint and great performance.

JavaFX, Swing, SWT, HTML5 – Who wins – or better, when to use what?

I don't think it is a choice of winners, more a matter of what makes sense. Clearly HTML5 makes a lot of sense for many simple business tools that have straight forward interface requirements and smallish data requirements. Your cookie-cutter data entry form and simple high level statistical plotting dashboard for example.

JavaFX makes more sense when you have highly custom interfaces and large data management requirements, especially for Enterprise applications intended for focused usersets. The data visualizations I must build have way too much information for a browser to manage. JavaFX gives me the beauty and performance to build these software tools with a small, modern, manageable code base. The recent development that Android is moving to the OpenJDK is great news for JavaFX as now JavaFX apps will be much easier to port and have even better performance now using fantastic tools like Gluon's JavaFX-Ports and Charm.

How satisfied are you with the work of Oracle on JavaFX ?

I'm happy with Oracle's desktop development and their team for that is great. I am disappointed with the JavaFX 3D offering as there is so much potential to be had with having such a seamless 2D/3D mixed framework but the feature availability is too low for the majority of developers. JavaFX 3D was so lacking that Jose Pereda, Jason Pollastrini and myself took it upon ourselves to create an open-source offering of 3D components, F(X)yz, to help fill that gap. Hopefully the underlying OpenGL calls will be opened up to the developer with the release of JavaFX 9. That would allow a lot of frameworks and community libraries to port to and expand into the JavaFX space.
I also feel Oracle really missed an opportunity with the Android mobile market with JavaFX. If they had jumped on mobile architecture support much earlier, the rise of Android devices would have ignited JavaFX adoption. This created a development gap for several years until Gluon stepped forward with some of their great offerings.

What do you miss in the work with JavaFX?

A true pluggable windowing application framework like the NetBeans Platform. All the great parts of using an application framework like the Platform, the reusable workflow components, docking framework, Services and Lookup APIs, must be developed from scratch. This is why we build hybrid applications that leverage the NetBeans Platform as an application framework but embed complex and custom JavaFX components