A look into the future with Flex

The future of Adobe Flex is all about enabling the collaboration between the designer and the developer. Kicking off the second day of Devoxx University, Chet Haase and Matt Chotin took a close look at the Flex 3 component model, the use of Catalyst and the next version of Flex, code name Gumbo.

The Devoxx audience only needed a very short introduction to Flex, allowing Matt Chotin to dive deeper into the Flex 3 component model. Chotin, who holds the position of Flex Product Manager at Adobe, went into great detail explaining the Flex component lifecycle. “That’s the mechanism the framework uses to create, manage and destroy components”, he said. Chotin used the example of a video component to show the audience how a component is initialized, defined and destroyed throughout its lifecycle. Step by step, Chotin laid out the Flex component model as an important foundation for building UI components.

No more linearity

Next up was Chet Haase, Senior Computer Scientist on the Flex SDK team at Adobe. He is also known as the co-author of the book ‘Filthy rich clients: developing animatic solutions’ and the author of the book ‘Parleys’. In order to enable the collaboration between the designer and the developer, Catalyst allows the designer to go back to Photoshop or Illustrator to make changes in his graphical design. These changes are taken into account by Catalyst as well, resulting in a nicely structured project that the developer can start working on right away.

“Gumbo and Catalyst are expanding the designer’s freedom and responsibilities.”

Efficient collaboration

That’s of course where Flash Catalyst steps in. “We want the designer to have the freedom of designing the entire interface of an application, not just some skins”, said Chet Haase. “With Catalyst, the designer can actually create interactive components, starting from just a graphical piece that he put together in Photoshop or Illustrator.” In order to enable the collaboration between the designer and the developer, Catalyst allows the designer to go back to Photoshop or Illustrator to make changes in his component model as an important foundation for building UI components.

In reality, a lot of the designer’s work ended up in the trash can. “There clearly was a broken process”, said Chet Haase. “It was not possible for the designer and the developer to go back and forth with their part of the work. There was only a small possibility of achieving iterative design, mainly because the designer and the developer use totally different tools. “A different take on linear component building didn’t make much of a difference. This second approach starts at the developer’s side, with the designer coming in second, adding some graphical skinning to the solution. “With this approach, the interactivity of the solution is entirely created by the developer, although this should be one of the designer’s main challenges.”

No more linearity

Next up was Chet Haase, Senior Computer Scientist on the Flex SDK team at Adobe. He is also known as the co-author of the book ‘Filthy rich clients: developing animatic solutions’ and the author of the book ‘Parleys’. In order to enable the collaboration between the designer and the developer, Catalyst allows the designer to go back to Photoshop or Illustrator to make changes in his component model as an important foundation for building UI components.

In reality, a lot of the designer’s work ended up in the trash can. “There clearly was a broken process”, said Chet Haase. “It was not possible for the designer and the developer to go back and forth with their part of the work. There was only a small possibility of achieving iterative design, mainly because the designer and the developer use totally different tools. “A different take on linear component building didn’t make much of a difference. This second approach starts at the developer’s side, with the designer coming in second, adding some graphical skinning to the solution. “With this approach, the interactivity of the solution is entirely created by the developer, although this should be one of the designer’s main challenges.”

The success of Devoxx

Today, the Devoxx Conference kicks off with lots of keynotes and parallel sessions. However, these grounds were already filled to maximum capacity the last two days. The two University days, along with the different other sessions, attracted a record crowd to the Metropolis site in Antwerp. Changing the name from JavaPolis to Devoxx merely strengthened the success of this event.

The future of Adobe Flex is all about enabling the collaboration between the designer and the developer. Kicking off the second day of Devoxx University, Chet Haase and Matt Chotin took a close look at the Flex 3 component model, the use of Catalyst and the next version of Flex, code name Gumbo.

The Devoxx audience only needed a very short introduction to Flex, allowing Matt Chotin to dive deeper into the Flex 3 component model. Chotin, who holds the position of Flex Product Manager at Adobe, went into great detail explaining the Flex component lifecycle. “That’s the mechanism the framework uses to create, manage and destroy components”, he said. Chotin used the example of a video component to show the audience how a component is initialized, defined and destroyed throughout its lifecycle. Step by step, Chotin laid out the Flex component model as an important foundation for building UI components.

No more linearity

Next up was Chet Haase, Senior Computer Scientist on the Flex SDK team at Adobe. He is also known as the co-author of the book ‘Filthy rich clients: developing animatic solutions’ and the author of the book ‘Parleys’. In order to enable the collaboration between the designer and the developer, Catalyst allows the designer to go back to Photoshop or Illustrator to make changes in his graphical design. These changes are taken into account by Catalyst as well, resulting in a nicely structured project that the developer can start working on right away.

“It is not our purpose to have Photoshop and Illustrator replaced by Catalyst”, explained Haase. “We are only adding a new tool to the chain, allowing a swift interaction between the tools of the designer and the developer.” To conclude their talk, Chotin and Haase took a look at the new version of Flex, code name Gumbo, that is expected to tear down the wall between design and code. “This is the future of Flex”, they said. “As there are not that many people around who are excellent designers and developers at the same time, there is a clear need for a nice workflow between the designer and the developer.” The new capabilities in Gumbo that Chotin and Haase demonstrated look very promising indeed. “But we are still working on the compiler”, admitted Haase. “It’s not as fast yet as it should be.”
As a steady employer, Smals offers you interesting ICT-projects of social significance. Use your expertise in Java to build a social security internet portal for a wide audience, allowing over 400 million visits a year. Construct applications to quickly register all recruitment nationwide, even when initiated from a mobile. Or work for eHealth, the secured information exchange for healthcare services.

Engage in the future and defy society’s challenges in Java.

Do you want to build tomorrow?

Java Developer
Java Architect
Functional Analyst
Project Leader

Discover all our vacancies at www.smals.be.
More information at 02 787 53 12.
Recruitment - Rue du Prince Royale 102 - 1050 Brussels

Think big, act Smals.
www.smals.be
Every day, a steering comity member of Devoxx advises us on what you should not miss at Devoxx. It’ll be a heavy first Conference day, if you follow the advice of Sven Beaulieu.

After getting an update in the first keynote from Sun on how UI development in Java and the corresponding tools evolve, I am looking forward to get some insight on how IBM handled the RFID implementation for Devoxx this year. The latter is quite an interesting case considering the large number of attendees we have again! For the first presentation, I have some difficulties in choosing between the presentations of Brian Goetz and Mike Wiesner, but since I believe that applying security correctly is still very much underestimated, learning some good ‘Security Patterns’ might help in avoiding some common pitfalls in the future.

After lunch, I am interested to hear Dave Nicolette giving me the right arguments to convince developers as well as managers that ‘Effective pairing’ really works and gives an added value in projects when applied correctly. The second presentation in the afternoon I want to attend, is about solving a problem which I have not encountered a lot, but nevertheless takes a lot of time to implement when needed: ‘Entity Versioning’. I hope to get some insights in how Adam Warski solved this cleanly in his framework.

Next up is an update of the latest version of the ‘Spring framework’. More specifically I want to hear Arjen Poutsma and Alef Arendsen elaborate on the new REST and Portlet 2.0 support in 3.0. The last presentation of the day is again a hard one to choose. I would like to hear more about ‘Web Beans’ by Peter Muir, as well as about ‘RESTful Design Patterns and Anti-Patterns’ by Stefan Tilkov.

As a long time advocate of REST, I’ll go for the latter presentation to gain some extra knowledge about some REST design principles for future implementations.

Hope to see you all tonight at the Meet & Greet to discuss the different presentations! 😊

### TIME

<table>
<thead>
<tr>
<th>ROOM 8</th>
<th>ROOM 5</th>
<th>ROOM 4</th>
<th>ROOM 9</th>
<th>ROOM 6</th>
<th>ROOM 7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>08.00 - 09.30</strong></td>
<td>Registration, Breakfast and Exhibition</td>
<td>Keynotes</td>
<td>Keynote happens in room 8 and overflow in room 5 and 2nd overflow in room 4</td>
<td>Special Guest and Welcome (Stephan Janssen)</td>
<td>Java and RFID by IBM</td>
</tr>
<tr>
<td><strong>09.30 - 11.30</strong></td>
<td>Creating amazing user interfaces with Dojo and DWR by Joe Walker</td>
<td>Fingers in the air: a gentle introduction to software estimation by Giovanni Asproni</td>
<td>Hippo CMS 7 sneak preview by Arjé Cahn</td>
<td>Security Patterns revealed by Mike Wiesner</td>
<td>Lap Around Silverlight 2 and its Dynamic Language Runtime by Katrien De Graeve and special guests Kevin Dockx, Ivan Porto Carrero</td>
</tr>
<tr>
<td><strong>11.30 - 12.00</strong></td>
<td>Java SE</td>
<td>Web</td>
<td>Methodology</td>
<td>Partner</td>
<td>Security</td>
</tr>
<tr>
<td><strong>12.00 - 13.00</strong></td>
<td>From Concurrent to Parallel by Brian Goetz</td>
<td>Creating amazing user interfaces with Dojo and DWR by Joe Walker</td>
<td>Fingers in the air: a gentle introduction to software estimation by Giovanni Asproni</td>
<td>Hippo CMS 7 sneak preview by Arjé Cahn</td>
<td>Security Patterns revealed by Mike Wiesner</td>
</tr>
<tr>
<td><strong>13.00 - 14.00</strong></td>
<td>Lunch and Exhibition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>14.00 - 15.00</strong></td>
<td>Effective Pairing: The Good, the Bad and the Ugly by Dave Nicolette</td>
<td>The future of rich internet applications by Matt Chotin and Chet Haase</td>
<td>How to hack and secure your Java web application by Sebas-tian Deleersnyder</td>
<td>High Performance Java by Irao Hutchinson</td>
<td>The Threat is Out There! XML Threats and Data-Power by Bill Hines</td>
</tr>
<tr>
<td><strong>15.10 - 16.10</strong></td>
<td>Behavior driven development in Java with sasyb by John Ferguson Smart</td>
<td>The Feel of Scala by Bill Venners</td>
<td>What’s New and Exciting in JPA 2.0 by Mike Keith</td>
<td>The OW2 Code Base- Component Models in Action by Gaël Blondelle and Alexandre Lefebvre</td>
<td>Envers - Easy Entity Versioning by Adam Warski</td>
</tr>
<tr>
<td><strong>16.10 - 16.40</strong></td>
<td>Break</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>17.50 - 18.50</strong></td>
<td>Introduction to Web Beans by Peter Muir</td>
<td>Jython by Jim Baker and Tobias Ivarsson</td>
<td>RESTful Design, Patterns and Anti-Patterns by Stefan Tilkov</td>
<td>The JavaFX SDK by Richard Bair</td>
<td>Modularity in Java using OSGi by Peter Kriens</td>
</tr>
</tbody>
</table>
You program in Java, but still use a relational database.

Unlike relational databases, InterSystems Cache® is a perfect match as the back end for object-oriented programming. It’s the world’s fastest object database, and runs SQL queries up to 5 times faster than relational databases. Plus, with an innovation by InterSystems called Jalapeño™, Cache persists Java objects without relational mapping – reducing development time for Java programmers by as much as 40%.

Cache delivers massive scalability on minimal hardware, requires little administration, and incorporates a rapid Web application development environment. It’s available for UNIX, Linux, Windows, Mac OS X, and OpenVMS – and is deployed on more than 100,000 systems ranging from two to over 50,000 users.

We are InterSystems, a global software company with a 30-year track record of innovations that enrich applications.

Meet us at DEVOXX – booth 12 and attend our presentation on December 10 at 2 pm in room 6. Visit InterSystemsBenelux.com/DEVOXX to learn more and win a skype phone.
RFID enabled badges make life easier for participant, organizer and sponsor

The quantity feedbacks of Devoxx 2008 will be evaluated thanks to a new system. With RFID enabled badges all movements of participants are anonymously registered. Paperwork for participants at sponsor stands is minimized as a scan of their badge will provide the necessary coordinates. The biggest challenge for the team was not the technology itself, but the size and circumstances of the implementation.

At a conference with 3,000 participants, more than 130 sessions and 6 rooms it is difficult to get representational or quantitative feedback from the different sessions. Robin Mulkers, Software IT Architect at IBM: "Simply asking people to fill in evaluation forms is not a practical option anymore." Not only organisers but also sponsors at big conferences are looking for technologies to automate information forms." A team from IBM, Intermec and Skill-Team proposed Devoxx to develop an innovative project to make life easier for participant, organizer and sponsor.

RFID badges
All participants at Devoxx will be provided with RFID enabled badges. The RFID infrastructure can track attendance anonymously to the different sessions and pre-fill electronic contact forms for the sponsors in the booth. Robin Mulkers: "The RFID tags used are of latest generation UHF EPC Gen 2 passive tags and will be used for both proximity reading (10 cm) and distance reading (3 meters)." A small disadvantage might be the quality of the RFID tags location as a service. The solution is deployed on IBM System x servers. Figure 1 shows the whole configuration.

Privacy
Privacy was an important requirement for this project. The system is designed in such a way that the history of the locations visited by a participant is stored without any reference to the identity. It will be possible to get, for example, the participants' e-mail address from the RFID tag identifier but this association is not stored in the same system and remains the propriety of the Devoxx organization. Livens: "It is also theoretically possible to track where someone is positioned, but we will not use that." Participants are informed on the RFID badges in case persons have a problem with it, they can ask the registration desk to remove their chip. Mulkers: "I do not expect many persons to complain."

For the team the implementation of RFID technology on events is not new. Waltniel: "The technology has already been implemented during the innovation weeks and the New Year's reception of IBM. At Devoxx we do not talk about a demonstration or a showcase. It is a full implementation." And Mulkers concludes: "With this visibility, we would not do it if we were not sure about its reliability."
Each time I learn a new language, I learn something about programming. When I learned Java as a C++ programmer, for example, Java's interface construct taught me the value of multiply inheriting from pure abstract base classes. Although this style of programming was possible in C++, I didn't think about multiple inheritance this way in my C++ days, and I didn't use abstract base classes much in my C++ designs. Once I began programming in Java, however, I started using the style all the time. Learning about Java—in particular, its interface construct—changed how I approached OO design.

A similar effect has happened as I’ve learned to program in Scala. In the past two years I’ve worked quite a bit with Scala, a new statically typed language for the Java Platform that fuses object-oriented and functional programming concepts. Scala allows me to write code that’s almost as concise as Ruby or Python. I can call into Java libraries, including my existing Java libraries, from Scala as easily as I can from Java. Given Scala is statically typed, I enjoy the benefits of static typing such as types as documentation, code completion in IDEs, deterministic refactoring, and execution speed. (The performance of Scala programs is about the same as Java programs.) But Scala also gives me concise and type-safe ways of accessing some of the benefits traditionally associated with dynamic languages, such as the ability to add new methods to existing classes, or to pass types that don’t share a common hierarchy to a method.

How did Scala change how I think about programming? In short: I learned to appreciate the functional style. The functional style of programming emphasizes immutable objects, variables that can be initialized but not reassigned (final variables in Java), transformation of data structures, and methods and control constructs that result in a value but have no side effects. At the other end of the spectrum is the imperative style, which is characterized by mutable objects, variables that can be reassigned (normal variables in Java), indexing through data structures, and methods and control constructs with side-effects.

Although Scala is often touted as a functional programming language, it is not exclusively functional. Scala supports both functional and imperative styles. You can, if you choose, program in Scala much the same way you program in Java, which is likely a predominantly imperative style. This helps ease the Scala learning curve, but as you get more familiar with Scala, you might find yourself preferring functional alternatives. I did. Why? I discovered that functional style code tends to be more concise and less error prone than the corresponding imperative style code. Functional style code is often higher level, which makes it quicker to write and easier to read. As an example, consider this Java code, which determines whether a string contains an upper case character:

```java
boolean nameHasUpperCase = false; // This is Java
for (int i = 0; i < name.length(); ++i) {
    if (Character.isUpperCase(name.
        charAt(i)))
        nameHasUpperCase = true;
    break;
}
```

In Scala, you could write code similar to the previous two examples, but the most idiomatic way to write this in Scala is:

```scala
val nameHasUpperCase = name.exists(_.isUpperCase)
```

The `exists` method is a method that returns true for one of the passed characters. It returns true for one of the passed characters, i.e., that one of the characters is upper case—returns true. Otherwise it returns false.

Although the last one-liner may look cryptic to someone not familiar with Scala, once you know Scala, you’ll be able to see at a glance the purpose of this code. By contrast the other two versions will take just a bit more study. Another difference to note is that a potential off-by-one error exists imperative example, because you must explicitly indicate the upper index to which to iterate. This error can’t happen in the functional versions, and in this way, they are less error prone.

Lastly, I want to point out that I did not turn “completely functional” when I went to Scala. Although I’ve found that functional style code is most often more concise, cleaner, and less error prone, I’ve also found that sometimes the imperative style leads to clearer, more concise code. In such cases I use it. Scala allows me to use both imperative and functional styles easily, to combine them in the way I find most optimal for the clarity of the code.

---

**Bill Venners:** “Although Scala is often touted as a functional programming language, it is not exclusively functional.”
RoxorLoops opens Devoxx Conference

Introducing today’s keynote speakers at the Devoxx Conference is Belgian beatboxing phenomenon RoxorLoops. Like most of the other presenters at Devoxx, he’s just a man with a microphone.

RoxorLoops – born Senjka Danhieux – is a beatboxer. He masters the art of vocal percussion. Beatboxing is all about producing drum beats, rhythmic and musical sounds using the mouth, lips, tongue and voice. Sometimes it is also combined with singing or the vocal simulation of several musical instruments. In other words: it sounds as if you hear a full drum kit, or even a full band, where all the music is actually coming from just one single artist. RoxorLoops is one of the leading beatboxers in Europe. “I got into beatboxing at the age of seventeen, about seven years ago,” he says. “I saw a beatboxer perform and I just couldn’t believe that all I was hearing was coming from just one man.”

RoxorLoops learned making the first couple of basic sounds from other beatboxers and friends. “That was just the start,” he recalls. “Once you get the basic stuff, there’s a lot of work to do, to make it all sound exactly the way you want it.” RoxorLoops compares his art to learning a foreign language. “You can start with learning some basic vocabulary. But at the same time, you need grammar, you need to work on pronunciation, and so on. This way you learn making sentences and you can start building a story.” RoxorLoops’s hard work paid off. He won the Belgian championship in 2004 and got the second place at the world championship in 2005, making him the virtual European champion. “A championship is decided on a number of one-on-one battles, one beatboxer against the other,” he explains. “The jury quotes style, performance, ... Obviously, it’s sometimes hard to determine who’s the better beatboxer, just like it’s hard to compare competitors in a band competition.”

It’s all just programming

Over the years, beatboxing has become a fulltime job for RoxorLoops. He organizes local championships and runs beatbox.be, the website of the Belgian beatbox community. In Belgium, about fifty beatboxers are active in the scene. At Devoxx, RoxorLoops will introduce the audience to the world of beatboxing. “I always change my show a bit, depending on the audience I’m working for”, he explains. “Usually I start with demonstrating some of the basic sounds, from there on I build up the show. There are some elements that I need for the structure of the show, but there’s quite a bit of improvising as well.”

As it is the first time RoxorLoops will be performing for an audience that consists mainly of Java developers, he’s not sure yet which way his show will go. Or is he? “When you think about it, there’s a link between programming an beatboxing, I guess. All the beats and sounds I make are also programmed in a way. But instead of programming electronic beats on a computer, I only use my head and mouth.”

Beatboxing phenomenon RoxorLoops is introducing today’s keynote speakers.

THERE’S AN INTERESTING JOB FOR EVERY ONE HERE!

Thomas Jakemeyn, developer

What is a young dynamic fellow doing in a large company? Some facts about working together, responsibilities and career opportunities.

Thomas, you have been working for over a year as developer now at the Post. How do you like it?

“I like it very much. On a daily basis, we organize short stand-up meetings with the sub teams, so we know what every one is doing, where we are in the process, what work must still be done, who can assist a colleague, ... Every team member knows exactly what’s going on.”

What exactly are your responsibilities?

“I have several responsibilities. I am developer for two HR applications within The Post, where we look for solutions for the customer. Furthermore, I am Release Coordinator and Application Lead. As Release Coordinator, I maintain the overall view of the different versions of an application. I proactively follow up the progress of the application over the different environments – from development to acceptance to production. As Application Lead, I help searching for answers to customer needs within agreed time and budget.”

Do you have the feeling that there is room for personal growth?

“Certainly! You can either evolve towards Project Management, or technical expert or combine technical work with analysis, communication and contact with The Post’s business units and become a Senior Software Engineer. The Post’s ICT department invests a lot in training and encourages employees to follow courses to improve both soft skills and technical skills”

In a nutshell: what does working at the ICT department of The Post mean to you?

“Working in a dynamic environment, on interesting projects, facing a lot of challenges, to working with motivated and pleasant colleagues ... not a day passes without something interesting happening.”
Unleash Your Creativity

Sun’s open source leadership gives you the freedom to bring innovation to the market

Over 5.5 billion Java Powered devices worldwide

Distributed over 11 million Solaris 10 licenses

60,000 downloads per day

Learn more at http://sun.be/open

© 2008 Sun Microsystems, Inc. All rights reserved. All logos and trademarks are property of their respective owners.
Outsourcing is hot! Every major corporation around the globe is outsourcing all or part of their software development. They outsource to the large software houses in for instance India or in the US and UK which of course have offshore developments in India. And unfortunately the result is lots of unsuccessful projects. However, the failures are often never talked about because the vendor has got a fantastic contract and the customer won’t speak up about poor contracts that they reviewed and approved. This is wonderful business for the software houses. This reminds me of the saying: “My grandfather was a horse trader, my father was a used car salesman, I am myself in the software business.”

Why do these types of projects fail? They can all be summarized by saying that they were not smart. The buyer doesn’t know what he is buying and the vendor doesn’t know how to make it work. Would you buy a house from someone by just saying ‘give me what I need’, and then pay them a fixed price? No, of course not, but this or similar is what many outsourcing deals are about. After the agreement the buyer just waits to see what he will get say three years down the road. On that path he will have to sign off on requirements that look good and complete, but which due to the everchanging nature of software requirements are only correct to say 60%. Following that path you will get drawings that you don’t know if they ever can be built. At the end of that path you will have to take over the software without knowing for sure that it works for the people who are supposed to live with it. This is what people do. The vendors are protected because you agreed originally to sign off after each phase and you did so because you believed it would work. It doesn’t work for software. The vendor wants happy customers, but she just doesn’t know how to do these kinds of huge projects successfully. Or maybe the deal on the table is irresistible, and since, most often the vendor is more competent than the buyer and they won’t put themselves in a position to lose anything.

What I say here is controversial. However, someone has to say it. Now, it should also be noted that there are companies that are very successful in outsourcing. Their approach to outsourcing is very different based on an understanding on the nature of software, on the importance of starting small but scaling up as quickly as possible (but not quicker “grin”). Tomorrow I will talk about:

1. Quickly build an architectural roadmap.
2. Separate work that can be given a fixed price from work that cannot reasonably be calculated upfront.

Outsourcing made wrong
Ivar Jacobson

From Monday to Thursday, Parleys publishes a column by Ivar Jacobson. Ivar Jacobson is best known as the ‘father of use case’ and together with Grady Booch and James Rumbaugh (the three of them are known as the ‘Three Amigos’), he first created UML, Unified Modeling Language.

Is Java your ‘second skin’?
And do you feel very strongly about ‘diversity’ and ‘personality’?

If the answer is positive, be sure to visit us at Devoxx!
You will find us at Metropolis Antwerp, booth 29, from the 9th till the 11th of December included.

And do not forget to collect your free laptopskin, tailored to your personality! At Cegeka, ‘diversity’ shows in a varied offer of Java vacancies:

Are you in search of the challenge par excellence for a new career move within a close team? Our Java Software Factory is the appropriate answer! The Software factory is an analysis and development team of about 60 experts. The daily activities include Extreme Programming, refactoring, sprint sessions and scrum. The factory applies proven practices, based on Agile Software Development techniques. If teamwork and a direct and intensive collaboration with the customer have great appeal for you, we herewith invite you for a personal call about the possibilities in our Software Factory.

Java specialists are always welcome in our business unit ‘Expertise’. These colleagues are called in to assist at interesting projects at the customer’s site. Are you an expert in Java, J2EE, web and database technologies (JMS, JDBC, EJB, Spring, JPA, JSP, JTA, REST, Tomcat, Oracle ...)? And do you combine technical knowledge with a practical experience in the analysis, design, development, implementation and delivery of large-scale Java applications? A number of interesting vacancies is waiting for you at Cegeka!

Interested? See our vacancies at www.cegeka.be/nl/jobs or send your CV to jobs@cegeka.be.
at Devoxx, one of today’s keynotes is dedicated to JavaFX, the family of software products for creating RIA (Rich Internet Applications). RIA are web applications that have the features and functionality of traditional desktop applications, including interactive multimedia solutions. JavaFX allows developers to build applications for the desktop, for mobile phones, television and other platforms – all with the same code. On December 4, Sun officially launched JavaFX release 1.0. Everything that’s new and exciting about JavaFX 1.0 will be presented and discussed during today’s keynote on JavaFX. Parleys got a preview from Param Singh, Senior Director of JavaFX at Sun Microsystems.

How different is JavaFX 1.0 from the first ideas you had for it when you announced the platform at JavaOne 2007?

Param Singh: “Obviously, the definition of JavaFX has evolved quite a bit since that first announcement. At the time, the idea was to develop a platform for RIA across multiple screens. But soon it became clear that everyone wanted the ability to build RIA on top of Java, including animation, video, audio... At the same time, everybody wanted all functionality to be delivered in an easier way.”

Massive opportunity

JavaFX was first announced at JavaOne in 2007. Release 1.0 is live since last week. Is JavaFX arriving too late in the market?

Param Singh: “We took the time we needed to make JavaFX the comprehensible platform it is today. We had to make some important changes to the core of Java. We wanted to deliver a platform that is able to run across all screens. From this perspective, JavaFX is only rejuvenating Java. There are six million Java developers out there that can benefit from JavaFX. I don’t think JavaFX arrives too late.”

JavaFX 1.0 includes JavaFX runtime for the desktop, meaning that when people download the JRE (Java Runtime Environment), they get JavaFX as well. JavaFX really has become part of Java. The new JavaFX script language includes functionality for binding and animation, running ten to twenty times faster than Javascript. The advantage of JavaFX script is that you can learn it really quickly, and of course that you can actually have animation and binds in the language itself. And there’s more. The browser plug-in has been rewritten from the ground up, and we included ‘drag to install’ functionality. You can actually drag visually rich applets to your desktop and save them for later. Release 1.0 also includes a preview version of JavaFX mobile runtime. Right now, you can already run JavaFX applications on the mobile emulator to see what they would look like. The next version of JavaFX runtime – to be released in early 2009 – will include mobile runtime. Also important is that we added a plug-in for NetBeans IDE, which resulted in a sophisticated development environment to build, preview, and debug JavaFX applications.”

What about the use of video in JavaFX?

Param Singh: “We know that application developers always wanted to work closely together with the graphical guys. Most of the time, it was quite hard to make this happen. That’s why we decided to add plug-ins for Adobe Photoshop and Illustrator. Thanks to these plug-ins you can save graphic layers as JavaFX, wrapping the image automatically into the Java code. This actually works both ways, making the relationship between developers and designers a lot easier. In JavaFX 1.0, developers can export code from Photoshop and Illustrator into JavaFX code. It is our strategy to incrementally add functionality this way. As we reach a broader audience of designers, we are committed to deliver – directly or through partners – designer tools to visually create Java applications.”

What about the browser back-button support for JavaFX applets?

Param Singh: “We understand the importance of that. We are in the process of extending functionality to put browser back-button support in Java.”

Tight integration

What do you consider as the main differentiators for JavaFX when comparing the platform to Adobe Flex and AIR, or to Microsoft Silverlight?

Param Singh: “The integration between Java and JavaFX is very tight. Developers that work with JavaFX also don’t need to start learning how to work with yet another platform.”

How and when will the JavaFX applets be indexed by Google?

Param Singh: “That’s part of our roadmap. It all has to do with indexing content that is in the browser. We are investigating this as we speak. By JavaOne 2009, we will be ready to announce when we expect the indexation to be completed.”

Visual support

How have you been taking care of the visual aspect of the developer’s job?

Param Singh: “With JavaFX 1.0 we already are providing tools that allow developers to work closely together with their graphical colleagues. We figured it would be easier to get a visual preview of the graphical part of a development, rather than having to write all the graphical stuff in code. Right now, we are working with several third-party tool vendors to get their RAD tools (Rapid Application Development) totally visual and ready to generate JavaFX code. It is our strategy to incrementally add functionality this way. As we reach a broader audience of designers, we are committed to deliver – directly or through partners – designer tools to visually create Java applications.”

What will be the next steps in Sun’s strategy concerning JavaFX designer tools?

Param Singh: “With JavaFX 1.0 we already are providing tools that allow developers to work closely together with their graphical colleagues. We figured it would be easier to get a visual preview of the graphical part of a development, rather than having to write all the graphical stuff in code. Right now, we are working with several third-party tool vendors to get their RAD tools (Rapid Application Development) totally visual and ready to generate JavaFX code. It is our strategy to incrementally add functionality this way. As we reach a broader audience of designers, we are committed to deliver – directly or through partners – designer tools to visually create Java applications.”

What about the use of video in JavaFX?

Param Singh: “Video is important for JavaFX 1.0. We signed an agreement with On2 Technologies to bring comprehensive video capabilities to the JavaFX product family, using On2’s TrueMotion video codec. We decided to take one step at a time. We start working with On2’s codec now. Later on, we will add other codecs for both video and audio, using both native and cross-platform codecs.”

Param Singh, Senior Director of JavaFX at Sun Microsystems: “Java and JavaFX combined attack both the desktop and the browser. Add mobile to that next year, and it’s easy to realize how important JavaFX is.”
Brian Goetz has been a software developer for more than twenty years. He is the author of ‘Java Concurrency In Practice’ and is a senior staff engineer at Sun Microsystems. He is a frequent presenter at Java conferences around the world. Since he was a presenter at Javapolis before, he’s more than happy to return to Antwerp. At Devoxx 2008, Brian Goetz has two talks lined up: one about the additions to the concurrency library in Java 7, the other about how the JVM (Java Virtual Machine) is evolving to better support other languages.

Why are parallelism and concurrency such hot topics for programmers these days?
Brian Goetz: “Until quite recently, speed wasn’t really an issue. With clock rates doubling every eighteen months, all you had to do was wait for the next generation of processors to come along. When a program was a little slow when you first wrote it, you knew it was going to be okay a year later, when the next generation of hardware would be available. But that’s all in the past now. The 3 GHz processor is the fastest we have, and it’s been around for several years already. We hit the power wall, as they say.”

How does a developer begin to deal with concurrency in Java?
Brian Goetz: “First of all, you need to untangle the problem. Only then can you start thinking about the right solution. The thing is that you just can’t move a program from a single processor to a multi-processor environment. To make optimal use of the computing power that is available, you basically need to keep cutting up each problem, until you reach the smallest possible task.”

Best of both worlds
At Devoxx you will also talk about how the JVM is evolving to better support other programming languages. Can you give us a little preview?
Brian Goetz: “Well, we all know that Java has been very successful over the last ten years or so. Through the years, a whole ecosystem has come to life. It’s an ecosystem that allows us to solve all sorts of problems in Java. At the same time, we see a lot of people that want to start working with other languages, but without giving up all the benefits from Java. That’s where we got the idea to have other languages running on the JVM. We are examining what changes we can make to the JVM to make it a friendlier place for Ruby, Python and the likes. The idea is to have all the benefits we know from Java combined with the specific benefits of other languages. We are receiving a tremendous amount of input from the other communities. I strongly believe the result will present the developers with the best of both worlds.”

Probabilistic thinking
Little known fact: Brian Goetz is also a talented poker player. “It’s just a hobby”, he says, “nothing more, although it is nice to have a hobby that doesn’t just cost money but actually allows you to make some money as well.” Would it be correct to assume that playing poker requires the same way of thinking that is needed to solve complex software engineering problems? “In a way, yes. Playing poker is all about checking as assumptions: what happens if I do this, what happens if this move goes wrong, and so on. In that perspective, playing poker surely helped me in the way I look at complex systems. Playing poker and engineering aren’t always about true or false, yes or no. They’re both very probabilistic, like life itself.”

Brian Goetz, Sun Microsystems: “To handle concurrency, we need to change our way of thinking. We all have learned to think chronologically for a long time, because that was the way to work with the hardware that was available at the time. With concurrent hardware – as in the real world – things don’t happen one thing after the other. Everything happens at the same time.”

You published your views on the subject in the book ‘Java Concurrency In Practice’. How exactly do you help the developer?
Brian Goetz: “The book depicts the world we – Java software developers – live in. It says what we need to get by. To handle concurrency, we need to change our way of thinking. We all have learned to think chronologically for a long time, because that was the way to work with the hardware that was available at the time. With concurrent hardware – as in the real world – things don’t happen one thing after the other. Everything happens at the same time.”

Well, we all know that Java has been very successful over the last ten years or so. Through the years, a whole ecosystem has come to life. It’s an ecosystem that allows us to solve all sorts of problems in Java. At the same time, we see a lot of people that want to start working with other languages, but without giving up all the benefits from Java. That’s where we got the idea to have other languages running on the JVM. We are examining what changes we can make to the JVM to make it a friendlier place for Ruby, Python and the likes. The idea is to have all the benefits we know from Java combined with the specific benefits of other languages. We are receiving a tremendous amount of input from the other communities. I strongly believe the result will present the developers with the best of both worlds.”

What are the topics related to Java concurrency that you will present at Devoxx 2008?
Brian Goetz: “These days, programmers want to dive deeper into concurrency, whereas in the past they used to look at coarse grained parallelism only. The fork/join framework has been around for some time already, but so far it was never used that much in Java. Our goal is to broaden and enhance the concurrency library in Java 7. During my talk at Devoxx I will give an overview of the problems programmers are facing regarding this matter, as well as of the solutions that are available today.”
That’s all fine, but the configuration hasn’t evolved in a very long time, and is really starting to feel its age. You see, it’s basically a glorified scheduler. Modern, state-of-the-art build servers (or Continuous Integration servers, to use the techie term) can do so much more than that. For one thing, there is always a fair delay between when a developer checks in some changes and when the emails go out alerting everyone of a crash. The other day, for example, some guy in that team on the other side of the world committed some changes that broke the build. Not a big deal in itself, but he committed those changes at the end of the day over there. Sure enough, the email notifications went out, but by the time they did, the only one there to read them was the cleaner! So when we got to work the following morning, the build was broken, and we were stuck for the day!

We need to know about build failures faster than that. In fact, whenever someone makes a change that breaks the build, I want to know who made the change, what code was modified and why, what JIRA issues were involved, and what tests it broke. I know that sounds a lot, but there’s more. I want to know within minutes. Mail is sluggish, and not all the developers read their mail immediately. We should set up a system of much faster notifications, using technologies such as Instant Messaging or SMS, so that there can be no more missed build failures.

Best practices

It could help with the development practices initiative we were discussing the other day too. As you know, we have been trying to standardize our development practices and coding conventions, as people lose a lot of time switching between projects. Every team seems to do things so differently! Each time you switch to a project, you need a day just to understand the build script! If everyone observed the same basic best practices and conventions, switching from one project to another would be a lot smoother, and new team members could be productive a lot more quickly! We have been looking at tools like Maven, which help define a standard directory structure and build life-cycle across all projects. We’ve also been investigating code quality tools like Checkstyle and FindBugs, that help ensure that people follow the same coding conventions and observe common coding best practices. They even pick up the occasional bug before it even gets into Subversion! If we could integrate these into our build server, we could make sure that everyone sticks to the company conventions. With modern IDEs, it’s not too hard to do once you are aware of the standards. It would also be a great way to train new staff!

Our application architecture is a bit of a mess in places, too. The code needs to be more modular, with a better understanding of exactly what libraries each module needs. Joe has written a really useful API called cool-tools.jar. Everyone is using it. Problem is, everyone has a different version, and they’re all called cool-tools.jar! And they aren’t all compatible, so upgrading from one version to another to get a cool new feature is a nightmare!

A well-tuned build server would also make it easier to isolate those hard-to-find bugs. Take last week, for example, when the users were complaining about those big performance issues. They started happening sometime over the last few weeks, but no one knows exactly when, or why, they started. We do know that the tests have been taking a long time to run of late, but no one has been taking much notice, as there is no obvious test failure. If we had a build server that kept track of how long each test took to run, we could check to see when the tests had started to slow down, and what code changes were made at this time. So, even if the tests aren’t failing, we could still figure out why performance has taken a nose-dive.

Testing

Another area where we really need to get our act together is testing. Not QA testing, I mean developer testing. See, when a tester finds an issue, it is usually weeks after the code was written. It can take us developers quite a while to sift back into that old code and figure out what’s wrong. Now I’m not saying we don’t need tester testing. We do. It’s just that the sooner we detect errors or regressions, the easier they are to fix. So we need to make sure everyone is doing their bit, and that we have well-tested code in all of our apps. What we need are more efficient ways of testing. To test more efficiently with less test code. And we need to coach everybody with these new techniques, until testing becomes second nature, because, in reality, it’s not that simple.

There are plenty of good testing tools and techniques out there that can help us improve our game. Some people are using a technique called ‘Behavior-Driven Development’, to make sure that their tests are as relevant as possible, and that they are really coding to the requirements. Others are using new languages like Groovy and easy to make their tests more expressive, and to do more in-depth testing with less code. And others are using new features in the latest version of JUnit to write their tests more efficiently. There’s plenty to be done. So, as I said, we need a new build server, Boss.

If you want to find out more about John Ferguson’s Smart’s views, go to his talk in Room 8 at 15.10 today.
Escaping Java verbosity

One of the big trends of the latest years is people getting annoyed with the verbosity of Java and looking for ways to avoid it. Java is something Neal Ford calls a ‘high ceremony’ language, a language that requires you to write a lot of characters in order to convey your intent to the computer.

One of the Java’s biggest pain points probably is that it provides only one way to pass around small blocks of code via anonymous inner classes. Passing blocks of code is needed in many cases, e.g. listeners in UI code, functional-style programming, and so on. Many people are not too concerned about this. After all, IntelliJ IDEA and other modern Java IDEs cope with the verbosity quite efficiently. They can generate the boilerplate code for you with just a couple of key strokes, and in many cases can also fold the generated code away so that you don’t have to read it.

Closures

Some developers believe that the right way to go is to fix the Java language, in particular by adding closures. There are a number of proposals for that, and there’s even a complete implementation of one of these proposals which you can download and use today (http://www.javac.info). Unfortunately, it doesn’t look like any of the proposed syntaxes will be included in JDK 7. Currently, the main obstacle to using closures is the lack of IDE support, which is not likely to appear before Sun commits to supporting one of the specific syntax proposals.

Another option to escape from the verbosity of Java – switching to a different language that also runs on the JVM – is now gaining popularity. The number of attractive options grows each year: Groovy, Scala, JRuby and Jython are already pretty wide-spread, and the newest contender is Clojure. All of these languages are fully compatible with existin Java libraries and code bases, and at the same time much more concise and efficient. The major JVM-based languages (Groovy, JRuby, Scala, and Jython) are supported by IntelliJ IDEA 8.0. And, unlike some other IDEs, IntelliJ IDEA doesn’t treat the code in each of these languages as something isolated. You can use full code completion while calling Java methods from code in a different language, and you can navigate between methods in different languages just as easily as within Java code. Even refactoring work across languages. Renaming a method in Java will correctly update all of its calls regardless of the language.

In fact, our developers at JetBrains are making full use of the cross-language functionality, because each of the plugins is at least partially written in the language that it supports. The Scala plugin is entirely written in Scala, and the plugins for Ruby and Python contain significant chunks of high-level code (implementations of specific refactorings, inspections and intentions) that run under JRuby and Jython.

Language Oriented Programming

One more alternative is to use the Language Oriented Programming approach. The essence of the approach is to define a domain-specific language (or a number of languages) for your problem domain, define rules for generating compilable and runnable code from your DSL, and write the high-level logic of your project using those DSLs. That approach is still far from being widely spread in the industry, but for a number of years JetBrains has been working on a language-oriented programming tool called MPS (Meta-Programming System). The MPS programming language and the ‘Oslo’ toolkit, announced by Microsoft at the PDC conference this October, is also a language-oriented programming tool.

When you program in MPS, you start by defining a number of concepts for your language. Concepts are somewhat similar to classes in object-oriented programming languages and represent the main constructions of your language. For example, if you’re using a DSL to define the business rules of your application, each type of business rule can be comprised of one or more concepts. Instances of a concept (for example, specific business rules) are called models. Neither concepts nor models are stored as text, and you don’t define the grammar of your language. Essentially, with MPS you’re editing a syntax tree of your program directly (of course, with the full set of productivity tools, including error highlighting, code completion and refactoring).

Generators specify how models are turned into code in a different programming language. For example, you can use MPS to generate Java code, XML, CSS or JavaScript. A generator is essentially a template containing macros (loops, references and so on) which are substituted with the contents of your models. Languages in MPS can also reuse other languages. For example, if your business rules contain mathematical expressions, you can reuse the expression language included in MPS by default.

If you want to learn more about our approach to cross-language development, we encourage you to attend the presentation by JetBrains’ developers Dmitry Jemerov and Ilya Sergey, today at 16:40 in room 4.
Over the past year, there has been a focus within the Java community on scripting languages for the JVM. Among the most exciting of these scripting languages is PHP. As one of the technologies behind some of the most innovative applications on the web, many large sites from portals to social networks to blogs base their enterprise code on PHP.

This highly dynamic, easy-to-learn language also has a huge developer base and a thriving open source community. Some of the best contributions of the PHP community are the large space of open source applications and UI design. Because of this, PHP’s strengths in many ways complement those of Java. With PHP on the JVM, the best of both worlds is now available to Java- and PHP-based enterprises.

How does PHP on the JVM work?

Quercus is Caucho Technology’s implementation of PHP in Java. (I am most familiar with Quercus, so I will discuss only its technical details in this article.) All the language features and libraries were recreated in cleanroom Java without referring to the C implementation of PHP, avoiding the need for JNI. Quercus is implemented as a Java Servlet and runs PHP scripts much in the same way as JSP scripts are run. When a request is received for a certain PHP script, the Quercus Servlet reads that script from the file system and either interprets it in its PHP interpreter or compiles it to Java and runs it directly on the JVM. Subsequent requests are served directly by the compiled PHP script until the original source is changed.

Calls to PHP database functions are passed through JDBC connection pools for efficiency and database load reduction. Quercus understands existing PHP scripts, but also helps integrate PHP with Java by allowing PHP scripts to access Java objects and call methods on them. This feature lets developers use custom Java code or frameworks in conjunction with PHP.

Java and one for PHP applications, the two application stacks are joined together for one easy-to-manage deployment. Java monitoring tools are also now available to manage the PHP applications as well.

Use open source PHP applications as application frameworks. The success of many of the open source PHP applications is their ability to use a plug-in or componentized architecture to add functionality. Using Java code from PHP to write plug-ins for an existing PHP application creates an easy, extensible framework for Java. For example I recently worked with a Caucho customer who used Quercus to run MediaWiki and integrated a Spring-based application.

Why run PHP on Java?

Compiled PHP performance in the Quercus implementation has shown to be up to 85% than the standard C implementation of PHP, even with opcode caching enabled. Running PHP on the JVM also means that any improvements to the JVM/JDK are automatically available to PHP. For example, the updates in Java 6 over Java 5 have shown significant performance boosts in Caucho’s internal tests.

Java application servers can take advantage of mature clustering capabilities for failover that are often difficult to construct with non-Java deployments. Most application servers contain or can use database connection pooling to failover database connections and provide better response time. Quercus on the Resin application server also provides automatic session replication for PHP applications.

While some Java integration options for PHP exist, they often use inefficient mechanisms like XML-RPC. By instead calling Java directly from within the same process by running the PHP code in the JVM, the overhead of network and XML processing is avoided. Calling PHP from Java is also much simpler with the JSR-223 javax.script API.

PHP in its C implementation is interpreted, meaning that many errors that may be caught in compiled languages are not found until runtime. By compiling PHP to Java, the Java compiler acts as a static analyzer to find bugs in PHP code. Java also has extensive developer tools such as profilers and memory analyzers. PHP developers can use these directly on the compiled code to find inefficiencies in their code.

PHP and Java are a natural fit. As two of the most popular languages in web development today, PHP is known for its great UI and open source applications while Java is known for its enterprise capabilities and open source libraries. By taking the best of both languages and the communities behind them, Java-based sites can get to market faster, have better looking sites, and maintain the stability, scalability, and performance they’ve always enjoyed.

### Table: PHP Features vs. Java Features

<table>
<thead>
<tr>
<th>PHP FEATURES</th>
<th>JAVA FEATURES</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Huge developer base</td>
<td>+ Enterprise reliability and performance</td>
</tr>
<tr>
<td>+ Great UIs</td>
<td>+ High quality, open source application frameworks</td>
</tr>
<tr>
<td>+ High quality, complete open source apps</td>
<td>+ Enterprise-level development and monitoring tools</td>
</tr>
</tbody>
</table>
CSC is like a campus, a state of mind, where you get the professional freedom to develop yourself as an individual and as a team member. With 91,000 employees worldwide, our single mission is to put IT to work in practical, bottom-line ways.

In Belgium, CSC helps large clients from all major market sectors to achieve real business results by the use of innovative solutions. We bring together the combined strengths of 500 professionals who all share the same client-intimate approach: we listen to our clients, respect their culture and their way of working, and work with them as a true team.

CSC is currently looking for JAVA Consultants:

- You have a Master or Bachelor degree.
- You are passionate about technology, with an analytical and a solution-oriented mind.
- You are an excellent teamplayer and you are willing to work in a dynamic and demanding environment.
- You have good communication skills in Dutch, French and English.

We offer you challenging opportunities in a dynamic and team-oriented environment.

If you are interested in working for a world-class organization that strongly values independence and work-life balance, send your cover letter and CV (with reference “Devoxx1208”) to jobs-belgium@csc.com or visit our website: www.csc.com/be.

Join CSC

www.freedomworksbetter.be
Quite a crowd at Devoxx

On this second day of the Devoxx University, the sponsor booths were already set up. The place was really packed with Java-specialists hanging out and socializing. We countered a number of them.