Sinds JavaOne is de spanning rondom JavaFX stevig opgebouwd. Het bleef lang onduidelijk wat we precies konden verwachten en wat JavaFX betekent voor Java ontwikkelaars. Met de 1.0 release voor de deur is duidelijk dat JavaFX klaar is om de strijd aan te gaan met de concurrentie in de wereld van Rich Internet Applications. JavaFX is echter nog niet de oplossing voor ieder type applicatie.

JavaFX: Klaar om de wereld te veroveren?

JavaFX wordt door Sun gepositioneerd als de RIA oplossing voor de Java-wereld. Sun gaat hiermee direct de concurrentie aan met Flex van Adobe en Microsofts Silverlight. Interessant hierbij is om te zien dat JavaFX een hele andere insteek heeft dan de concurrenten. Het belangrijkste onderdeel van JavaFX is de nieuwe taal JavaFX Script en de bijbehorende API. Deze taal maakt het heel eenvoudig om zelf op een declaratieve manier dingen op het scherm te tekenen, terwijl Flex en Silverlight de focus leggen op het gebruik van (standaard) componenten. De keerzijde van de aanpak van JavaFX is dat er slechts een zeer beperkte set van standaard componenten in de huidige release geleverd wordt en dat je op dit moment zelfs de meest eenvoudige componenten zelf moet maken.

Geen standaard componenten

In de preview release van JavaFX is het aantal standaard componenten zoals tekstvelden, knoppen en checkboxen zeer beperkt. De componenten die er zijn, zijn in de basis gewoon Swing componenten. Daar is ook direct een probleem mee. Een scherm in JavaFX is namelijk opgebouwd uit *Nodes*, terwijl de Swing componenten dat niet zijn. Om deze componenten toch te kunnen gebruiken moet je deze binnen een container component *ComponentView* gebruiken, dat op zijn beurt wel een Node is. Niet echt ideaal. Bovendien zijn er nog geen native layout managers voor JavaFX. Ook hier ben je dus aangewezen op Swing componenten of een eigen layout mechanisme. In codevoorbeeld 1 is een voorbeeld van het gebruik van een *ComponentView* te zien.

```
stage: Stage {
content: [
ComponentView {
component: Button {
text: "Button"
action: function() {
                    }
                }
            }
        ]
    }
```

Codevoorbeeld 1

Componenten versus zelf tekenen

Het gebrek aan componenten betekent echter zeker niet direct dat JavaFX niet mee kan komen met Flex en Silverlight, de focus is alleen anders. Met Flex en Silverlight is het heel makkelijk om binnen de grenzen van de componenten hele mooie applicaties te maken. Met JavaFX kun je alles buiten deze grenzen. Applicaties waar veel formulieren worden gebruikt, kosten meer tijd om te ontwikkelen, maar hier krijg je volledige vrijheid op het gebied van design voor terug. En dat zijn nou juist de echte RIA applicaties; de applicatie die verder gaat dan de traditionele op formulieren gebaseerde applicatie met alleen een mooier jasje. Daarmee lijkt JavaFX misschien wel meer op Flash dan op Flex of Silverlight, met als groot verschil dat het belangrijkste onderdeel van JavaFX, de bijbehorende taal is en niet een tekentool. JavaFX is daarmee misschien nog het beste tussen Flash en Flex en Silverlight in te positioneren.

Omdat het heel eenvoudig is om met JavaFX zelf componenten te maken is het te verwachten dat binnen korte tijd allerlei component libraries op de

Paul Bakker
is trainer bij Info Support en geeft cursussen over onderwerpen zoals EJB3, JSF,Spring en JavaFX.

Afbeelding 1

(open source) markt zullen verschijnen. De eerste tekenen zijn daar al zichtbaar van. James Weaver heeft op zijn blog een aantal hele leuke voorbeeldcomponenten staan. Afbeelding 1 is bijvoorbeeld een menucomponent dat hij heeft gemaakt.

Web en desktop

JavaFX is zowel te gebruiken voor desktop- als webapplicaties. Om JavaFX binnen een browser te gebruiken wordt de Applet runtime gebruikt. In de laatste versie van de JRE (JRE6 update 10) is er veel verbeterd om de browserervaring te verbeteren. Niet voor niets wordt update 10 ook wel de 'Consumer JRE' genoemd. De downloadtijd voor de JRE is drastisch verbeterd door de JRE op te delen in kleine modules die pas gedownload hoeven te worden op het moment dat dit noodzakelijk is. Om de webervaring verder te verbeteren, is ook het opstartscherm, oftewel het splashscreen, van Applets aangepakt. Waar je vroeger een dominerend oranje Sun logo midden op je pagina kreeg tijdens het laden van een Applet, kun je nu als ontwikkelaar zelf een splashscreen maken. Dit kan zelfs een geanimeerd splashscreen zijn, dus er is zeker iets leukers van te maken dan het gele Sun logo. Een andere indrukwekkende functionaliteit die helemaal niet terug te vinden is bij de concurrenten is de mogelijkheid om Applets buiten de browser te slepen, waarna de applicatie buiten de browser blijft opereren, zelfs als de browser afgesloten wordt. De release van de nieuwe JRE heeft natuurlijk alles te maken met het creëren van een platform voor JavaFX. In de lente van 2009 komt JavaFX bovendien uit voor mobiele apparaten en moet je elke JavaFX applicatie op bijvoorbeeld een telefoon kunnen draaien. Hoe dit precies gaat werken is nog onbekend, maar het is wel duidelijk dat dit groots wordt aangepakt.

IDEs

Natuurlijk heeft Sun gezorgd voor ondersteuning in Netbeans. Andere IDE's worden tot dusver glashard genegeerd en de vraag is of dat handig is. Veel ontwikkelaars zitten te wachten op ondersteuning voor bijvoorbeeld Eclipse en om de adoptie van JavaFX te bevorderen zou Sun hier ook best wat meer aandacht aan mogen schenken. Aan de andere kant is het niet gezegd dat dit niet gaat komen en op dit moment is het af krijgen van JavaFX zelf natuurlijk een stuk belangrijker. Sun kan JavaFX echter ook als argument gebruiken om over te stappen op Netbeans. Het wachten is dan op partijen die wel met ondersteuning voor JavaFX komen in andere IDE's.

De ondersteuning in Netbeans is bruikbaar, maar nog wel erg beperkt. Er is een palette waarmee je shapes en componenten naar je editor kunt slepen om code te genereren. Handig op het moment dat je nog onbekend bent met de JavaFX API, maar daarna nauwelijks van meerwaarde. Het automatisch uitlijnen van code is wel onmisbaar. Zonder goede uitlijning is JavaFX code nog meer onleesbaar dan niet uitgelijnde Java code. Het is duidelijk dat Sun op de goede weg is met JavaFX in Netbeans, maar nog wel wat werk moet verzetten om het echt op voldoende niveau te krijgen.
Een ander type tool dat we kunnen verwachten is grafische tooling waarmee je een JavaFX applicatie tekent. Denk hierbij aan Flash. Dit is misschien niet de meest belangrijke applicatie voor ontwikkelaars, maar is wel belangrijk voor de adoptie van JavaFX onder designers. Of deze tooling op tijd afkomt voor de 1.0 release is zeer de vraag. Tot nu toe heeft Sun nog geen enkele demo laten zien en blijven de ontwikkelingen rondom deze tooling erg vaag.

Van design naar applicatie

Een tool die wel positief verrast, is Project Nile. Dit project is op JavaOne 2008 aangekondigd als de manier om designers en ontwikkelaars samen te laten werken. Nile was ook al beschikbaar in de preview release en bestaat uit plugins voor AdobePhotoshop en Illustrator. Met deze plugins is het mogelijk om een ontwerp uit deze grafische pakketten te exporteren naar een JavaFX formaat, waarna dit design direct in JavaFX code te gebruiken is en dat werkt erg goed.

Het exporteren bestaat uit een aantal stappen die hieronder zijn beschreven:

Een grafisch designer maakt een compleet websiteontwerp in Photoshop. Elk element van de pagina wordt hierbij in een aparte laag gestopt. Bij het exporteren wordt elke laag opgeslagen als een plaatje en er wordt JavaFX code gegenereerd die deze plaatjes inleest en als objecten in code aangeboden worden. Wat bijzonder is aan de gegenereerde code is dat niet alleen de plaatjes worden ingeladen, maar dat ook grootte en de positie op het canvas beschikbaar zijn. Dat betekent dat je alle objecten uit het ontwerp alleen nog op het scherm hoeft te zetten zonder hierbij na te denken over de positie en grootte van ieder

individueel object. Hierdoor is het een echte no-brainer geworden om een ontwerp om te zetten naar code. Ook aanpassingen achteraf aan het ontwerp zijn makkelijk door te vertalen naar de code. De beschreven stappen zijn terug te zien in afbeeldingen 2 en 3, en het gebruik van de gegenereerde JavaFX code in codevoorbeeld 2.

```
var dukeUI = DukeUI {};
dukeUI.JavaFX_Button.onMouseClicked = function(evt)
{
System.out.println("JavaFX rocks!")
};

Frame {
stage: Stage {
content: [
dukeUI.Duke_rocks,
dukeUI.JavaFX_Button,
dukeUI.JavaFX
        ]
}
}
```

Codevoorbeeld 2. Het gebruik van de geëxporteerde assets. DukeUI is het gegenereerde component dat alle losse elementen van het design bevat.

Project Nile is misschien wel één van de belangrijkste onderdelen van JavaFX en laat direct de meerwaarde van JavaFX zien ten opzichte van Flex en Silverlight. Beide concurrenten hebben hele mooie tooling om een pagina en componenten te skinnen, maar er is geen mogelijkheid om een volledig grafisch design in een enkele stap naar code te vertalen.

JavaFX Script

Het meest opvallende onderdeel van het JavaFX platform is de taal JavaFX Script. De taal is inmiddels bijna niet meer terug te herkennen als je de implementatie hebt gezien die tijdens JavaOne 2007 gedemonstreerd is. De afgelopen anderhalf jaar is de taal volledig opnieuw geïmplementeerd en op veel plaatsen ook gewijzigd. JavaFX Script is een gecompileerde taal met statische typering en lijkt daarmee meer op Java dan op populaire talen zoals Groovy en Ruby. De syntax is echter volledig anders en heeft een hele declaratieve aard. Dit declaratieve programmeren kun je combineren met normale Java code, aangezien dit ook geldig is in JavaFX. De declaratievesyntax leent zich erg goed voor het in elkaar zetten van UIs en dat is natuurlijk ook het doel van JavaFX Script. Zaken zoals eventhandlers schrijf je vervolgens weer met Java syntax.

Om het leven als UI-ontwikkelaar nog eenvoudiger te maken bevat de taal *bind* expressies

Jij wilt het beste uit jezelf halen?
Wij zorgen dat je de beste opleidingen volgt.

Als Java specialist ben je nooit uitgeleerd. Ontwikkelingen op de voet volgen is een drive die wij verlangen. Daarom zorgen wij ervoor dat je maximaal gebruik maakt van onze uitstekende opleidingstrajecten, waarbij certificeren van belang is. Kernwoorden die we gebruiken zijn J2EE, JSF, JBoss Seam, EJB3.0, Java Portlets, AJAX, Spring en Hibernate. Tools die daarbij gebruikt worden zijn IBM Websphere/Eclipse en Oracle. Wil je weten wat SPIE nog meer te bieden heeft, kijk op www. SPIE-ICT.nl

SPIE

(codevoorbeeld 3) en *triggers* (codevoorbeeld 4). Hiermee kun je zorgen dat de user interface automatisch wordt geüpdate als bepaalde objecten of collecties wijzigen.

```
translateY: bind frame.height - button.height * 3
```

Codevoorbeeld 3. Het gebruik van een bind expressie. De waarde van translateY wordt elke keer opnieuw berekend als de hoogte van het frame wordt gewijzigd.

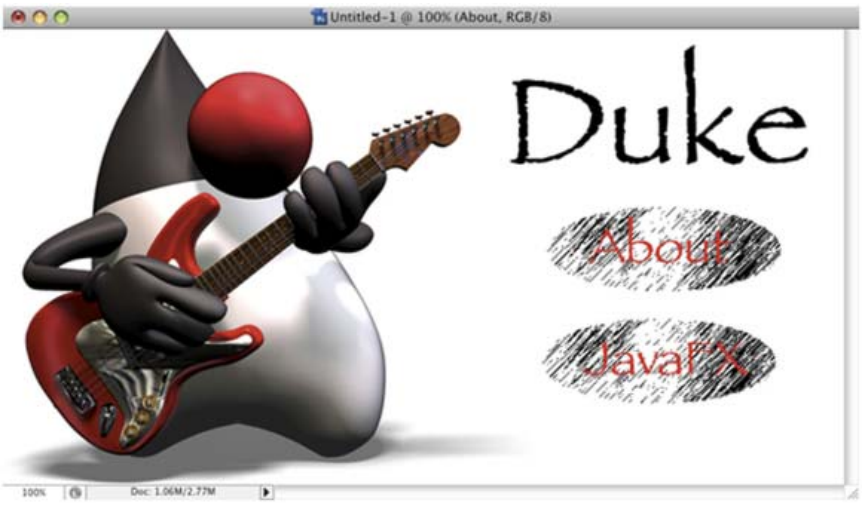
```
translateY: bind frame.height - button.height * 3

var items : String[] on replace {
System.out.println("Hallo");
};
insert "String 1" into items;
```

Codevoorbeeld 4. Het gebruik van een trigger. Elke keer dat er iets gewijzigd wordt aan items wordt er 'Hallo' naar de console geschreven.

Integratie met Java

JavaFX Script compileert naar Java bytecode. Dit betekent dat je vanuit JavaFX gewoon Java classes kunt aanroepen. Omgekeerd is ook mogelijk, maar omdat JavaFX vooral in de UI van de applicatie wordt gebruikt, doe je dit waarschijnlijk niet al te vaak. Integratie met Java is uitermate belangrijk. JavaFX Script is een taal die heel goed is in UI gerelateerde zaken, maar zich minder leent als general purpose taal. Een logische stap is daarom ook om alle niet UI gerelateerde zaken in Java te schrijven en alleen de UI in JavaFX Script.
Ondanks dat dit een goed idee is en dat ook de manier zal zijn om de meeste JavaFX applicaties te schrijven, zitten er nog wel wat haken en ogen aan de samenwerking met Java. Het probleem is dat Java classes wel te gebruiken zijn vanuit JavaFX, maar veel van de handige features van JavaFX op dat moment niet meer werken. Je kunt bijvoorbeeld niet properties op een Java class zetten met de syntax van JavaFX, maar je moet hier expliciet getters en setters voor gebruiken. Ook werken bindings niet, wat een groot probleem is voor het dynamisch houden van de UI.
De oplossing hiervoor is om wrapper classes in JavaFX Script te schrijven, die een normale Java class vertegenwoordigen. Op die manier kun je in de rest van de JavaFX code alle handige syntax en binding functionaliteit gebruiken. Het is natuurlijk wel jammer dat dit nodig is, aangezien de wrapper classes met de hand geschreven moeten worden.



Afbeelding 2. Een design met een aantal elementen in Photoshop.

Als je JavaFX in een webbrowser gebruikt, is dit probleem eigenlijk helemaal niet relevant. De Applet die in de browser draait, is al expliciet alleen de view. De overige code, bijvoorbeeld om een database te benaderen, leeft op de server en wordt met behulp van webservices aangeroepen. JavaFX gaat ook APIs bevatten om eenvoudig SOAP en Restful webservices aan te spreken. De API was niet op tijd klaar om mee te kunnen in de preview release, maar is wel beloofd voor de 1.0 versie.

Overigens kunnen Flex en Silverlight helemaal geen Java code aanroepen en moet je communicatie altijd via remoting of webservices doen.

Audio en Video

Misschien wel de grootste aankondiging van JavaOne 2008 was de ondersteuning voor audio en video voor het Java platform. Dit heeft alles te maken met JavaFX en was ook voor het eerst uit te proberen met de JavaFX Preview release. Audio en video kan native worden afgespeeld en performt daardoor ontzettend goed. Nu nog wachten totdat iemand een JavaFX versie van YouTube op de markt brengt. De technologie is er in ieder geval en Flash heeft er hiermee een stevige concurrent bij gekregen.

In de preview release is de platform ondersteuning nog wel zeer beperkt. Alleen Windows gebruikers kunnen van de video codecs gebruik maken en Linux wordt zelfs helemaal niet ondersteund. Dit wordt voor de 1.0 release rechtgetrokken, maar heeft wel al bij veel ontwikkelaars tot onvrede gezorgd.

Boeken en documentatie

Op dit moment is er slechts één boek over JavaFX op de markt: 'JavaFX Script' van James Weaver. Hoewel dit geen slecht boek is, is het boek nog gebaseerd op een veel oudere versie van JavaFX en is daardoor bijna niet meer bruikbaar. James Weaver heeft wel alle codevoorbeelden aangepast aan de nieuwe versie en online geplaatst. Het is te verwachten dat er binnen enkele maanden na de release nog wel een

Afbeelding 3. De lagen van het design.

aantal boeken gepubliceerd worden.

Op internet is de documentatie al niet veel beter. Bij het zoeken naar voorbeelden kom je erg vaak voorbeelden tegen die niet meer werken, waardoor de onduidelijkheid voor nieuwe JavaFX ontwikkelaars alleen maar groter wordt.

De documentatie die bij de preview release zit is up-to-date, maar zeer beperkt. De API documentatie bevat weinig voorbeelden en ontbreekt vaak zelfs volledig voor classes. Al met al nog niet geweldig, maar dit wordt na de 1.0 release ongetwijfeld heel snel beter.

JavaFX klaar voor gebruik?

Als je het volledige artikel hebt gelezen kun je de conclusie trekken dat JavaFX heel veel potentie heeft. Het heeft door de alternatieve aanpak zelfs de potentie om een stuk interessanter te zijn dan Flex of Silverlight. Tegelijkertijd zijn er ook nog zeker een aantal zaken niet voldoende op orde. Dat mag nu nog even, tenslotte is de 1.0 release nog niet gedaan, maar het is ook duidelijk dat waarschijnlijk niet alle problemen op tijd opgelost kunnen worden.

Afhankelijk van het type applicatie dat geschreven wordt, is dit meer of minder een probleem. Voor een echt grote desktop applicatie waarbij Java integratie heel belangrijk is, heb ik zeer mijn twijfels of JavaFX al volwassen genoeg is. Voor kleinere, zeer interactieve applicaties die afwijken van de standaard administratieve applicatie kan JavaFX wel al heel goed ingezet worden. Dit geldt nog meer voor web applicaties, waarbij zelfs grotere online applicaties goed bij JavaFX passen omdat hier de Java integratie geen rol speelt. Hierbij is het ontbreken van diepgaande browserintegratie (backbutton en bookmarking) en het missen van goede standaardcomponenten nog wel een probleem.

JavaFX voor Java ontwikkelaars

Misschien dat je langzaam het gevoel krijgt dat JavaFX meer een technologie voor designers dan voor developers is. Audio/Video, zelf componenten tekenen, integratie met grafische tooling, is dat nu iets waar Java programmeurs op zitten te wachten? Wat mij betreft hoort JavaFX zeker in de toolbox van een Javaan. Ontwerpers kunnen nog zulke mooie applicaties bedenken, wij zijn nog steeds degenen die de kennis hebben om zo'n applicatie te implementeren. JavaFX maakt het een heel stuk makkelijker om samen te werken met designers, en een nieuw type applicaties leven in te blazen. Probeer JavaFX zeker eens uit. JavaFX en Project Nile zijn te downloaden van de officiële JavaFX website. «

Referenties

Blog James Weaver: http://learnjavafx.typepad.com
JavaFX + Netbeansdownoad: http://javafx.com/
JavaFX reference: http://java.sun.com/javafx/reference/
JavaFXcursus: http://www.infosupport.nl/Training/CursusInfo?CourseCode=JAVAFX