New & Cool

De markt voor RIA-toolkits is turbulent. Wisten we drie jaar geleden niet wat ze waren, op dit moment worden we om de oren geslagen met nieuwe begrippen als *rich content, widgets, real-time Ajax-communicatie, single-page interfaces* en ik-weet-niet-wat. Een paar jaar geleden waren we nog tevreden met Struts en een beetje gepriegel met HTML, CSS en Javascript, nu weten we van gekkigheid niet waar we onze web-UI frameworks moeten zoeken. Voor dit artikel hebben we een shortlist gemaakt van drie RIA-toolkits die goed passen in de gereedschapskist van de Java-ontwikkelaar: Adobe Flex, Google Web Toolkit en JavaFX.

Drie populaire RIA-toolkits speciaal voor Java-ontwikkelaars

Keuze afhankelijk van requirements

Bart Guijt
Senior Consultant bij Xebia IT Architects, bguijt@xebia.com
Vincent Partington
Business Unit Manager van Xebia IT Architects, vpartington@xebia.com

Laten we eerst eens kijken wat een RIA-toolkit eigenlijk is. Volgens Jeremy Allaire, die de eerste white paper over *Rich Internet Applications* – geschreven heeft, zijn dit de belangrijkste eigenschappen van een RIA-toolkit:

- er is sprake van een *runtime platform* waarop de uiteindelijke applicatie draait (zoals de Flash Player en Java VM)
- het heeft een *widget library* en *non-visuele componenten* (voor bijvoorbeeld *event handling*), die vervolgens ook extensible zijn
- het is in staat om te *communiceren* met een webserver, bijvoorbeeld via JSON, XML-RPC of SOAP
- applicaties voor dit platform kunnen mogelijk *offline*- en *online* werken
- het voorziet in eenvoudige *distributie* (vrijwel altijd via de browser)
- er is een *IDE* of een *IDE-plug-in* beschikbaar voor het platform die *wysiwyg*-design mogelijk maakt.

Deze criteria zijn opgesteld met *Adobe Flex* (toen nog *Macromedia Flash MX*) in het achterhoofd, maar zijn gelukkig prima geschikt om ook andere serieuze RIA-toolkits te onderscheiden.

Waarom zou je een RIA-toolkit gebruiken?

De meeste webframeworks die op de Java Servlet API gebaseerd zijn doen grote moeite om alle onderdelen van een Java-webapplicatie bij elkaar te houden. De Java-webapplicatie is een ratjetoe van XML-configuraties, Java-classes en HTMLtemplates die aan elkaar worden gelijmd door het framework: kijk maar naar Struts, Spring-MVC, WebWork, JSF, enzovoort. Alle onderdelen zitten via configuratiefiles en framework-regels aan elkaar, zodat de relatie tussen de verschillende onderdelen lastig is te doorgronden. Helemaal voor Java-ontwikkelaars die gewend zijn aan de *statically-typed* natuur van de Java Language. *Code completion* in de IDE en specifieke meldingen van de compiler zijn zekerheden die bij webontwikkeling grotendeels overhoop gegooid worden. Het beeld dat het framework een soort grabbelton is, wordt versterkt door het feit dat we zelf alle HTML, Javascript en CSS moeten maken. Hier doen de Java webframeworks niet veel meer dan een paar JSP *taglibs* en zijn we overgeleverd aan onze eigen HTML-skills; weer extra tijd en moeite voordat we een webapplicatie hebben gebouwd.

Vervolgens hebben we te maken met het feit dat de HTTP request/response-cyclus gemanaged moet worden. Hiervoor hebben de frameworks allerlei XML-files die met een *Finite State Machine* beschrijven hoe een gebruiker door de webapplicatie mag lopen. Gelukkig zijn we er achter gekomen dat deze manier van werken ongeveer gelijk staat met het GOTO-statement uit Basic, en we weten allemaal wat Edsger Dijkstra daar veertig jaar geleden over heeft gezegd[1]. Ten slotte hebben we rekening te houden met verschillende browsers, waterdichte security en gesmeerde performance.

Met een RIA-toolkit kunnen we praktisch al deze problemen oplossen. De webapplicatie draait

volledig op de client, en hoeft niet meer bij elke muisklik een beroep te doen op de webserver om te bepalen wat er nu weer moet gebeuren. De webserver wordt uitsluitend gebruikt voor het ophalen van resources (HTML, CSS, Javascript, JAR of SWF-files) en voor het aanroepen van webservices. Verder heeft een RIA-toolkit de beschikking over een keur aan *widgets* en *lay-out managers* die het tekenen van een webapplicatie een stuk makkelijker maken, en die ook de verschillen tussen de browsers kunnen oplossen. Uiteindelijk krijgt de gebruiker met RIA-technologie een betere applicatie, die voor de ontwikkelaar ook nog eenvoudiger te maken is.

Welke RIA-toolkits zijn er zoal?

Ten eerste hebben we Adobe Flex, die met het begrip 'RIA' begonnen is (toen nog in de vorm van Macromedia Flash MX). Dit is een op Flash-gebaseerd platform, met extra widgets en componenten die we van een dergelijke toolkit verwachten (buttons, input fields, listboxes enzovoort). Na Adobe Flex zijn er vooral veel RIA-toolkits op basis van Ajax: Dojo, YUI, Ext en Qooxdoo om er maar een paar te noemen. Toolkits op basis van Ajax hebben een paar prominente nadelen. Ze moeten met allerlei verschillende runtimes (browsers) overweg kunnen, en performance is altijd een aandachtspunt; dit in tegenstelling tot Flex, Java en Silverlight, die allemaal hun eigen proprietary runtime hebben. Het voordeel dat Ajax toolkits hebben is dat zij beter aansluiten op de features van de browser – denk aan vorige/volgende knoppen, historie en bookmarks, maar ook meer (integratie)mogelijkheden bieden voor *mash-up's, bookmarklets* en *browser-plugins*. Vervolgens hebben we RIA op basis van het Java-platform. We kunnen een applicatie starten als Applet of via Java Web Start (dit geldt natuurlijk ook voor JavaFX). Microsoft is ook aan het RIA-avontuur begonnen na wisselend succes met ASP.Net. Ze is bezig om Silverlight onder de aandacht te brengen, die niet alleen in Windows en Internet Explorer werkt, maar ook geschikt is voor Safari op Mac OSX en Firefox op zowel Windows and Mac OSX.

Waarom vergelijken we GWT, Flex en JavaFX?

In dit artikel vergelijken we drie RIA-toolkits die volgens ons belangrijk zijn voor de Enterprise Java-community.

- Als eerste kozen we voor **Google Web Toolkit** (GWT). Deze RIA-toolkit stelt je in staat een Ajax-applicatie van het kaliber Google Mail te bouwen alleen door Java-code te schrijven. Kennis van HTML, JavaScript en CSS is alleen nodig wanneer je custom widgets wilt bouwen.
- Vervolgens viel de keuze op **Adobe Flex**. Deze toolkit heeft op het eerste gezicht niets met Java van doen. Flex wordt echter zo veel gebruikt voor het bouwen van RIA-applicaties, ook die met een Java-backend, dat we deze niet buiten beschouwing kunnen laten.
- Ook niet omdat we hem willen vergelijken met een relatief nieuwe loot aan de RIA-boom, **JavaFX**. Deze RIA-toolkit is door Sun geïntroduceerd tijdens JavaOne 2007, zeer ambitieus als de Java-tegenhanger van Flex. Met JavaOne 2008 lijkt de toolkit opnieuw te zijn geïntroduceerd, met veel beloftes, haperende demo's en alfasoftware. Ondanks de moeilijke start kunnen we JavaFX niet over het hoofd zien, omdat Sun de belangrijkste leverancier is van Java-technologie. Daarom nemen we deze toolkit mee in de vergelijking.

De vergelijking

We gaan deze toolkits vergelijken op een aantal belangrijke criteria, op de manier waarop dit in autoreviews ook gedaan wordt. Per aspect leggen we uit waarom het belangrijk is en hoe de drie toolkits het daarop doen. Ten-slotte kennen we punten toe en komen we met een conclusie.

Verscheidenheid aan programmeertalen

Zoals we hierboven al hebben beschreven, is één van de dingen die het bouwen van een webapplicatie vaak lastig maken het grote aantal talen (programmeertalen, mark-up talen, configuratietalen enzovoort) dat je ervoor moet kennen. Vaak is kennis van Java niet genoeg, maar moet je ook HTML, JavaScript en CSS kennen. Bovendien moet je soms nog een XML-dialect kennen waarmee jouw webframework geconfigureerd moet worden, zoals de Spring configuratie-files die in Spring 2.0 een stuk complexer geworden zijn. Op dit punt kiezen de drie geselecteerde toolkits allemaal een andere insteek. Om in Flex een applicatie te bouwen gebruik je MXML, een XML-variant om je interface te definiëren, en ActionScript, een taal die gebaseerd is op ECMAScript (JavaScript) om het gedrag daarachter te programmeren.

Dit is een voorbeeld van MXML:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/
mxml">
    <mx:Array id="sampleArray">
        <mx:String>Sample Label 1</mx:String>
        <mx:String>Sample Label 2</mx:String>
    </mx:Array>
    <mx:Panel title="Example Panel">
       <mx:ComboBox dataProvider="{sampleArray}"></
mx:ComboBox>
    </mx:Panel>
</mx:Application>

Dit is een voorbeeld van ActionScript: package {
```

```
    public class Greeter {
        public static function sayHello():String
            var greet:String = "Hello, world!";
            return greet;
        }
    }
}
```

In een JavaFX-applicatie heb je niet te maken met twee talen. Je gebruikt JavaFX Script (dat niet lijkt op Java of op JavaScript!) voor de definitie van de user-interface en voor het programmeren van het gedrag. Daarbij moet wel gezegd worden dat JavaFX Script zowel declaratief als procedureel gebruikt kan worden om hetzelfde te bereiken. Vergelijk

```
Frame {
   title: "Hello World JavaFX"
   width: 200
   content: Label {
      text: "Hello World"
   }
   visible: true
}
```

met

```
var win = new Frame();
win.title = "Hello World JavaFX";
win.width = 200;
var label = new Label();
label.text = "Hello World";
win.content = label;
win.visible = true;
```

Een GWT-applicatie schrijf je helemaal in Java, net als je een Swing-applicatie zou schrijven. Tenzij je een custom widget wilt bouwen, want dan zit je tot aan je nek in de HTML, CSS en JavaScript, eventueel met Javascript-libraries als script.aculo.us.

Remoting

Er is grote verscheidenheid aan remoting-mogelijkheden bij de drie toolkits. GWT en Flex hebben allebei een optimale, proprietary protocol ter beschikking (respectievelijk GWT-RPC en AMF) naast standaard JSON en SOAP. Al deze protocollen werken over standaard HTTP. Flex beschikt ook over RTMP – een 'push' remoting protocol, waarbij de server real-time communiceert met de client. Flex is de enige die 'push' remoting out-of-the-box ondersteunt. JavaFX biedt zelf geen remoting-functionaliteit. Omdat het mogelijk is om direct Java-code aan te roepen, kan wel gebruik gemaakt worden van alle mogelijkheden die het Java-platform biedt, zoals SOAP, RMI of (via een externe library) Hessian.

Tool support

Flex heeft, met grote voorsprong, de beste tool support. FlexBuilder is een volledige IDE voor het bouwen van Flex-applicaties met een ingebouwde debugger. Voor het ontwikkelen van een GWT-applicatie kan gebruik gemaakt worden van Eclipse. Cypal levert een set plug-ins om het ontwikkelen van GWT-applicaties eenvoudiger te maken, maar deze zijn niet noodzakelijk. Voor JavaFX geldt hetzelfde, behalve dan dat de plug-in hier geleverd wordt door Sun.

Debugging

Het debuggen van een Flex-applicatie is eenvoudig; De Flex Builder heeft een ingebouwde debugger waarmee je zowel de MXML als de ActionScript kan debuggen. Voor GWT-applicaties ligt de zaak iets complexer. Google raadt aan om ze te debuggen in 'hosted mode'. Hiermee debug je de originele Java-code voordat deze vertaald is naar JavaScript. Dat is handig, want Java debuggen kunnen we al (bijvoorbeeld met de debugger van Eclipse). Maar als je een probleem met je *custom widget* of met de vertaling naar JavaScript wil debuggen, zal je met tools als Firebug aan de slag moeten. JavaFX biedt (nog?) geen debugging-mogelijkheden.

Performance

Het vergelijken van de performance van GWT, Flex en JavaFX is moeilijk aangezien de technologieën zo verschillend zijn. Het voordeel van RIA-toolkits is dat ze in ieder geval een deel van de user-interface-processing verplaatsen van de server naar de client, en zo de server ontlasten. Dit werkt goed voor gebruikers met snelle werkstations, maar kan betekenen dat sommige mensen het moeilijk krijgen om de applicatie te gebruiken.
Een feit dat in ieder geval pleit voor de mogelijkheid tot het bereiken van een hoge performance met GWT is het feit dat Google soortgelijke Ajax-based technologie gebruikt voor eigen webapplicaties als Google Mail en Google Docs (jammer dat Google hier geen GWT voor gebruikt trouwens).
Flex draait in de Flash plug-in en dankzij de daarin ingebouwde JIT-compiler is de performance van gecompileerde ActionScript beter dan die van JavaScript. Nadeel is echter dat het downloaden van een Flex-applicatie langer duurt dan het starten van een GWT-applicatie.
Java heeft als client-side technologie nooit overtuigd (wie gebruikt er nog applets?) en hoewel de JVM-technologie in de laatste tien jaar met grote sprongen vooruit is gegaan, blijft JavaFX het zorgenkindje qua performance van deze drie technologieën, met name met het downloaden/starten van de applicatie. Sun is echter bezig met een vernieuwde Java plug-in. Deze moet met Java SE 1.6 Update 10 opgeleverd worden. Sun belooft grote verbeteringen in installatiegemak.

Security

Het security-model van JavaScript is niet erg sterk, getuige security-problemen als Cross-Site

Het vergelijken van de performance van GWT, Flex en JavaFX is moeilijk aangezien de technologieën zo verschillend zijn

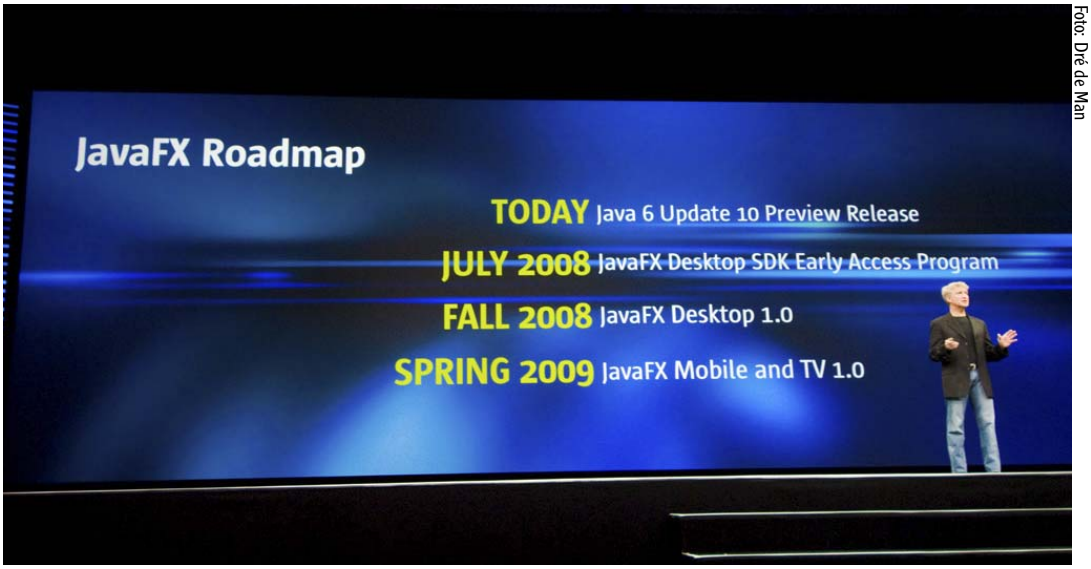

Foto: Dré de Man

JavaFX is nog steeds niet gereleased in een 1.0 versie

Scripting, Cross-Site Request Forgery en de ellende die je met JSON kunt uithalen. GWT probeert de ontwikkelaar hiervan af te schermen, maar immuniteit voor deze kwesties wordt niet beloofd. Vooral bij het zelf ontwikkelen van GWT-componenten in JavaScript liggen deze problemen op de loer.

JavaFX maakt daarentegen gebruik van het bekende 'sandbox'-gebaseerde security-model van Java. Hier zijn in ruim tien jaar geen grote problemen mee opgetreden. Wel betekent dit strakke security-model dat gebruikers soms gevraagd wordt of bepaalde acties wel toegestaan zijn voordat de Java Runtime ze uitvoert. Veilig, maar wel onhandig en verwarrend voor de gebruiker. Ook Flex maakt gebruik van een sandbox-model en is daardoor een veilig te deployen technologie. Wel zijn de mogelijkheden om de security aan te passen aan de eigen wensen bij Flex iets minder uitgebreid.

Mogelijkheid tot bookmarken

Zowel Flex- als JavaFX-applicaties draaien in een plug-in en daar kun je in principe geen bookmarks op aanbrengen. Maar Flex biedt de mogelijkheid gebruik te maken van fragment-identifiers om dat toch voor elkaar te krijgen. De ontwikkelaar moet hier wel zelf voor zorgen. GWT heeft wel support voor het gebruiken van de browser-history en bookmarks, waardoor je een favoriet onderdeel (een pagina dus) van jouw applicatie kunt bookmarken. Ook kunnen de gebruikers van jouw applicatie daardoor automatisch gebruik maken van de back- en forward-buttons.

Browser-ondersteuning

Ook op dit vlak wijken de drie toolkits erg af. GWT is compatible met alle belangrijke browsers (Firefox, Internet Explorer en Safari) zonder dat een plug-in nodig is. Maar dit geldt alleen zolang je je beperkt tot de standaard widgets. Wanneer je een eigen widget bouwt, moet je er zelf voor zorgen dat die werkt op alle browsers. Om een Flex-applicatie te runnen heeft de gebruiker de Flash plug-in nodig en om een JavaFX-applicatie te runnen de Java plug-in. Betrouwbare cijfers over de verspreiding van beide plug-ins ontbreken, maar waarschijnlijk ligt deze in beide gevallen boven de 90%, met een kleine voorsprong voor Flash.

Documentatie

Als je meer wilt weten over Flex kun je kiezen uit enkele tientallen boeken (zoek maar eens op Amazon). Over GWT is een handvol boeken verkrijgbaar en over JavaFX is nog maar één boek te vinden. Hetzelfde geldt ook voor de informatie die je online kunt vinden. Adobe biedt veel informatie over Flex (LiveDocs, Wiki, enzovoort), Google heeft ook een redelijke hoeveelheid aan informatie over GWT. Over JavaFX is eigenlijk weinig meer te vinden dan toen het nog F3 heette.

Marktpenetratie

Over JavaFX kunnen we heel kort zijn. Er zijn nog geen bekende sites die het gebruiken. GWT is niet veel beter. Wij kennen nauwelijks implementaties ervan en Google gebruikt het zelf niet voor zijn eigen bekende websites zoals Google Maps en Google Mail. Aan de andere kant, GWT kent een grote community met blogs, open source-libraries en frameworks, enzovoort. Flex kent daarentegen veel meer implementaties, zoals Yahoo! Maps, Photoshop Express en, dichter bij huis, de Schiphol Nomos-applicatie om de geluidsniveaus van Schiphol te zien.

De cijfers

Na alle punten apart besproken te hebben, komen we nu op het punt aan waar de cijfers toegekend

gaan worden. Hierbij maken we gebruik van de cijfers 1 tot en met 10, zodat we de cijfers kunnen optellen om tot een totaalscore te komen. 1 is slecht en 10 is perfect.

Eigenschap	GWT	Flex	JavaFX
Verscheidenheid aan programmeertalen	6	9	7
Remoting	8	9	6
Tool support	7	9	6
Debugging	8	9	6
Performance	8	8	6
Security	6	8	9
Mogelijkheid tot bookmarken	9	8	5
Browser ondersteuning	7	9	9
Documentatie	7	8	3
Marktpenetratie	7	8	3
Totaal	**73**	**85**	**60**

Een keuze

Een jaar na de introductie van JavaFX is er nog weinig reden om het toe te passen in een applicatie. De recente herintroductie bij JavaOne 2008 brengt geen verandering aan deze conclusie. De software is nog steeds niet gereleased in een 1.0 versie, en ook de Java-plug-in om de distributie van de JavaFX-applicaties te verzorgen is nog in bèta. Dat betekent dat we voorlopig moeten kiezen uit GWT of Flex. In de bovenstaande vergelijking komt Flex iets beter uit de bus, maar is GWT een serieus te nemen alternatief. De uiteindelijke keuze zal afhangen van de requirements als browser-compatibility of de kennis van de programmeurs. «

1 Zie "Macromedia Flash MX—A next-generation rich client" door Jeremy Allaire, maart 2002 - http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf_2 Zie "Go To Statement Considered Harmful" door Edsger Dijkstra, maart 1968 http://www.ifi.unizh.ch/req/courses/kvse/uebungen Dijkstra_Goto.pdf_3 Plugin penetratie: zie http://www.adobe.com/products/player_census/flashplayer/version_penetration.html en http://www.adobe.com/products/player_census/flashplayer/_