Een belangrijk thema tijdens een conferentie als JavaOne is uiteraard de ontwikkeling van applicaties. Naast alle aandacht voor de programmeertaal Java, software-engineering en platform, infrastructuur en architectuur, is de user-interface dit jaar misschien wel meer onderwerp van gesprek dan in recente jaren.

"Rich, active, mash up and very social"

Trends in Application Development

Niet alleen de verdere verrijking en AJAX-ificering van de web user-interface speelt een rol, ook de rentree van de applet en de opkomst van RIA-technologie als Silverlight en met name Flex is van groot belang – en wie weet kan JavaFX ook een steentje bijdragen. Het is duidelijk dat Web 1.0 applicaties – statisch HTML, full page refresh, weinig tot geen interactiviteit, nagenoeg geen grafische elementen anders dan statische boilerplate illustraties – passé zijn. Daarmee trek je geen webbezoekers en daarmee kan je ook de professionele computergebruikers die het jarenlang met groenletterige terminals hebben volgehouden, niet meer tevreden stellen. Overigens is productiviteit de term die deze nieuwe gebruikersinterfaces rechtvaardigt.

Een gerelateerd onderwerp dat grote invloed heeft op de inrichting van (internet) webapplicaties, maar in toenemende mate ook bedrijfsapplicaties, is 'Social Networking'. Community-aspecten, samenwerken (wiki), uitwisselen en delen en communiceren, via verschillende kanalen en met multimedia, die zo succesvol zijn gebleken op het internet, worden in hoog tempo ook onderdeel gemaakt van commerciële websites en professionele applicaties.

Zich snel ontwikkelende technologie voor server-push, zoals Comet, maakt het mogelijk om actieve user-interfaces te ontwikkelen – applicaties die ongevraagd vanuit de server worden ververst met actuele informatie, de laatste events of nieuwe RSS-onderwerpen.

Een laatste belangrijk aspect in de ontwikkeling van applicaties is de 'mash-up': de samenstelling van user-interfaces uit diverse onderling gesynchroniseerde 'widgets' die informatie betrekken uit verschillende services en databronnen. De widgets kunnen client-side zijn (JavaScript component) maar ook (deels) server-gebaseerd, mogelijk zelfs gedistribueerde portlets. Typische voorbeelden van mash-up-elementen zijn Google of Yahoo Maps-widgets die een postcode koppelen aan een kaartillustratie, aangevuld met gesynchroniseerde wereldtijdklok, weerbericht en verkeersinformatie.

Rich

Ooit werden Java Web Applicaties gebouwd met applets, in de tweede helft van de jaren 90. Java was in die tijd bijna equivalent met Swing en Applet. Na de eerste teleurstellingen, voor wat betreft productiviteit en de complexiteit van Swing, maar ook de beperkingen van browsers, plug-ins en bandbreedte, hebben we de applet grotendeels laten vallen en zijn we overgestapt op Flash-applicaties (voor consumenten websites) en HTML en servlet-gebaseerde webapplicaties.

Jarenlang was dat tamelijk overzichtelijk en vrij stabiel. Maar recentelijk zijn er allerlei ontwikkelingen gaande die deze overzichtelijkheid sterk onder druk zetten. Uiteraard is de opkomst van DHTML, CSS en vooral AJAX een aanjager van de verrijking van HTML-applicaties. Daarnaast zijn de ontwikkeling en recentelijk de open sourcing van Flex (voor de Flashplayer VM) een belangrijke motor achter de ontwikkelingen, zeker in combinatie met de sterk verbeterende aansluiting van Flex op Java-businessservices. Sun kwam in 2007 met de lancering van JavaFX (Script, Mobile en Platform) – kort samengevat een technologie om op eenvoudige wijze interactieve, rijke grafische applicaties te ontwikkelen voor mobiele toepassingen, webbrowser en desktop-applicatie. Hoewel JavaFX nog steeds niet beschikbaar is – een preview van de SDK is ons beloofd voor juli 2008 met de productierelease in het najaar – zet Sun de kruistocht voor de rijke, multiplatform UI wel door, ondermeer met als motto: How to make applets not suck".

Java 6 Upgrade 10 (the Consumer JRE)

De release van Java 6 Upgrade 10 – voorheen bekend als de Consumer JRE – is aanzienlijk belangrijker dan deze naamgeving doet denken. J6U10 moet toepassing van Java binnen browsers aanzienlijk gaan stimuleren. Belangrijke kenmerken van deze upgrade: snellere download en snellere start, snellere en betere grafische prestaties, betere JavaScript-integratie (verbeterde opvolger van LiveConnect), kleinere geheugenbelasting, nieuwe Nimbus look & feel en grotere stabiliteit. Deze Java plug-in draait grotendeels buiten de browser; er kunnen meerdere JRE-versies naast elkaar draaien en een crash in een daarvan raakt de andere noch de browser zelf. J6U10 vereist IE 7 of Firefox 3 (op dit moment in bèta).

Met J6U10 gaat het onderscheid tussen in-browser applicaties en desktopapplicaties verdwijnen – net als met Flex en Adobe AIR overigens. Je ontwikkelt niet specifiek voor deployment binnen of buiten de browser. JNLP (Java Network Launching Protocol), bekend van Java WebStart voor desktopapplicaties, kan voor J6U10 ook gebruikt worden voor ondermeer de configuratie

van applets. Applets krijgen ook toegang tot JNLP-functies als PersistenceService en FileOpenService. Eén van de consequenties hiervan is dat een applicatie die als applet in de browser draait met een simpele drag & drop naar de desktop kan worden gesleept en als stand-alone applicatie verder draait.
Op dit moment is J6U10 in bèta. De productierelease wordt 'laat in de zomer' verwacht. Deze release vormt de basis voor het JavaFX Desktop-platform dat in het najaar zal worden vrijgegeven.

De uitdaging met Java: Visueel en Grafisch Aantrekkelijk

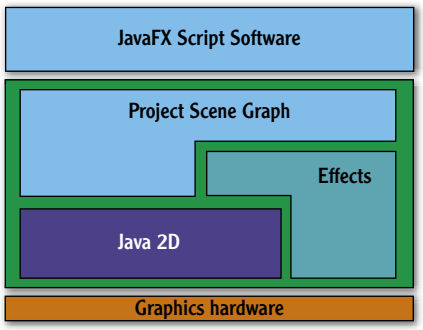
Met Java is het heel goed mogelijk om grafisch aantrekkelijke user-interfaces te ontwikkelen. Met de Swing en Java2D (en Java3D) libraries kunnen fraaie en zelfs spectaculaire pagina's en effecten worden ontwikkeld. Via Java ME kunnen (afhankelijk van het display van de mobiele apparaten) deze pagina's mobiel worden getoond, in de vorm van applets kunnen deze in webpagina's worden opgenomen en uiteraard kunnen ook op zichzelf staande desktopapplicaties worden ontwikkeld. Toch gebeurt dit maar weinig. Swing en Java2D worden als te complex en niet productief ervaren, waardoor Java veel minder dan bijvoorbeeld Flash/Flex en nu ook Silverlight het imago heeft van een taal en platform voor grafisch aantrekkelijke applicaties.
Sun heeft vorig jaar op JavaOne 2007 het JavaFX-platform aangekondigd. JavaFX: "simple, elegant and leverages the full power of Java". Met JavaFX zou het mogelijk moeten zijn om rijke, grafische user-interfaces te bouwen die in allerlei omgevingen – write once, run anywhere – gerund kunnen worden. Dit jaar zou JavaOne hopelijk de volwassenwording, of op zijn minst de concretisering van het JavaFX-platform brengen, was mijn hoop en stille verwachting. En eerlijk gezegd: het viel niet mee. De SDK komt in het najaar (oktober?) – met een bèta in juli 2008 - en ondersteuning op mobiele platforms zal pas in 2009 beschikbaar zijn. Op dit moment is er feitelijk nog steeds geen concrete software die ontwikkelaars kunnen gebruiken. (je kunt de ontwikkelingen rond JavaFX volgen op http://javafx.com/ en http://learnjavafx.typepad.com/)
Waar het heen gaat is natuurlijk wel wat duidelijker geworden. En wat je kunt doen met JavaFX en wat het eigenlijk is ook. JavaFX Script is een declaratieve scripttaal die de grafische mogelijkheden van Swing en Java2D – en het nieuwe Effects-package (com.sun.scenario.effect) – op eenvoudige wijze ontsluit. Dankzij JavaFX kan een ontwikkelaar op productieve wijze aan de slag met functionaliteit die in zijn 'pure' vorm nogal complex is. JavaFX kan worden ingebed in Swing-applicaties en andersom kunnen Swing-componenten met aanhangende code onderdeel zijn van JavaFX-panels.
Dat zou bijvoorbeeld kunnen betekenen – zeker met de verbeterde integratie tussen Applet en JavaScript – dat met JavaFX aantrekkelijke grafische widgets worden ontwikkeld die in HTML gebaseerde webpagina's worden ingezet, net zoals nu Flash-objecten kunnen worden geïntegreerd.

JavaFX is goed in het construeren van grafische objecten en het manipuleren van media zoals foto's en filmpjes. JavaFX biedt standaard faciliteiten voor vergroten/verkleinen, verplaatsen, vervormen, clipping en kleuringen. Bovendien kan iedere overgang in de waarde van eigenschappen – inclusief plaats, kleur, vorm, grootte

- geanimeerd worden. Met nauwelijks enige inspanning wordt de verandering van de x-coördinaat van een object gewijzigd van 0 in 100 met een geanimeerde overgang. Voor de animatie wordt een TimeLine-object gecreëerd die een collectie KeyFrames bevat. Ieder KeyFrame geeft de toestand aan op een bepaald tijdstip van één of meer variabelen. Een KeyFrame geeft aan hoe de transitie van de waarde plaatsvindt (lineair, ease in, ease out, ease both discrete (sprong)):

```
var animation = Timeline {
keyFrames: [
KeyFrame {
time: 0s
values: xValue => 0
},
KeyFrame {
time: 2s
values:
xValue => 100 tween Interpolator.LINEAR
},
]
}
```

Een erg krachtig aspect aan JavaFX is de *binding*. Eigenschappen van een grafisch object – kleur, grootte, positie, aantal punten van een ster – kunnen geassocieerd worden met een property in een ander object. Als die property wordt gemanipuleerd, bijvoorbeeld door een koppeling via een inputcomponent zoals een slider of de muis, wordt de eigenschap van het grafische object direct bijgewerkt. Zo kan eenvoudig een drag-en-drop of een zoom in/zoom uit-voorbeeld worden ontwikkeld. In de verte lijkt dit binding-mechanisme overigens wel wat op het gebruik van EL-expressies voor de eigenschappen van JSF-componenten.
Overigens is JavaFX (Script) niet de enige weg waarlangs Java-ontwikkelaars eenvoudiger toegang krijgen tot grafische toepassingen. Sun's project Scene Graph (https://scenegraph.dev.java.net/) werkt aan een library die het binnen gewone Swing-applicaties gemakkelijker zal maken om hiërarchische representaties van grafische objecten te creëren en manipuleren. Scene Graph vormt een laag om de Java 2D en Effects API's heen die zich eenvoudiger laat aansturen. Je geeft aan waar je welke objecten wilt hebben, maar je maakt je niet druk over hoe ze getekend of gemanipuleerd moeten worden.



Afbeelding 1.

Enige opwinding onder de aanwezige ontwikkelaars werd veroorzaakt door de aankondiging van JWebPane, een nieuwe Swing-component en extensie van JComponent, die een ingebouwde webbrowser biedt die geïntegreerd kan worden in een Java-applicatie. JWebPane is gebaseerd op WebKit – een open source webbrowser engine (zie http://webkit.org/). JWebPane ondersteunt HTML 4 & Street HTML, CSS, JavaScript.

Een simpel voorbeeld van het gebruik van de JWebPane in jouw eigen code:

```
JWebPane webpane = new JWebPane();
jframe.add(webpane);
...
webpane.load(new URL("http://www.sun.com"));
```

Rich met HTML

Web 2.0, de zoektocht naar rijkere web user-interfaces, en de toepassing van AJAX hebben met name betrekking op html-georiënteerde applicaties. Tientallen rich client JavaScript-libraries zagen de afgelopen jaren het daglicht, zoals Dojo, YUI (Yahoo), Scriptaculous, DWR, Prototype, jQuery, ext-js, enzovoort. Daarnaast zijn er frameworks als Google Web Toolkit, Phobos en Wicket die wat meer Java- en server-gericht zijn. Ook zijn er frameworks voor de JavaServer Faces-standaard die rich & AJAX-enabled zijn, zoals ICEFaces, JBoss Rich Faces en Oracle ADF Faces.
Al met al zijn er veel verschillende oplossingen voor rijke html-applicaties. Vervelend is dat die oplossingen in de meeste gevallen niet goed gecombineerd kunnen worden. Een RichFaces-applicatie uitbreiden met een widget uit de Scriptaculous-library of een pagina ontwikkelen met een mengeling van Yahoo-, Dojo- en jQuery-componenten is geen sinecure, evenmin als JSF-applicaties ontwikkelen met componenten uit verschillende 'rich' libraries als MyFaces Trinidad en ICEFaces.

Project jMaki is een initiatief van Sun (https://ajax.dev.java.net/) dat streeft naar een AJAX-framework dat het mogelijk maakt om widgets uit allerlei verschillende JavaScript-libraries te combineren en zelfs te integreren binnen JSP, JSF, Ruby en PHP webapplicaties. jMaki creëert wrappers voor de widgets – eenvoudige html- en JavaScript-files – die de widgets een consistente interface geven. (Maki is Japans voor het Engelse werkwoord 'to wrap'). Alle widgets hebben een vergelijkbare manier van configureren in de JSP-, JSF- of PHP-pagina. Simpel gezegd voor bijvoorbeeld JSPs: met de <a:widget> tag wordt een jMaki-widget in de pagina opgenomen die via het values-attribute wordt geconfigureerd. Aan de JSP is niet te zien of het widget van GWT, Dojo of Scriptaculous afkomstig is.
JMaki definieert ook gestandaardiseerde datamodellen – zoals tabel-datamodel en tree-datamodel – die voor widgets uit verschillende libraries kunnen worden gebruikt (zoals de table-widget uit Yahoo en de table in Dojo). Bovendien biedt jMaki een centrale, cliënt-side event-bus (Glue), waar alle widgets op geregistreerd kunnen worden als Publisher (ik ben geklikt, ik ben gesleept, ik ben gewijzigd) of Subscriber (als hij geklikt wordt, wil ik dat graag weten). Hiermee wordt het mogelijk om widgets uit totaal verschillende libraries niet alleen te combineren, maar zelfs te laten synchroniseren. JMaki biedt verder een proxy servlet dat namens client-side widgets externe data-services aanroept om RSS-, JSON- of XML-data (updates) te verkrijgen.
Project jMaki lijkt een mooie oplossing te bieden voor het combineren van JavaScript-widgets uit een breed palet aan libraries en die bovendien te integreren in meer server-side gebaseerde webtechnologie als JSP, JSF of PHP. Het

project is al enkele jaren oud, maar lijkt nog steeds bezig volwassen te worden – misschien heeft het wel erg veel ambities tegelijk. Dat het wel degelijk potentie heeft, bewijst de website www.travelmuse.com die op basis van ondermeer jMaki een rijke, interactieve, 'mash-up' user-interface biedt.

OpenAjax

Met jMaki is een goede en concrete stap gezet naar combinatie en integratie van rijke, AJAX-widgets, maar beperkt zich tot JavaScript-libraries en gaat uit van de bestaande verschillen en eigenaardigheden. Het OpenAjax-initiatief (www.openajax.org) is een industriebrede alliantie die onder andere de interoperabiliteit tussen AJAX enabled frameworks - en browsers – wil bevorderen. Van IBM, Oracle,Sun en Microsoft tot Dojo Foundation, Axapta en Nexaweb is een groot aantal leveranciers onderdeel van OpenAjax. Een eerste concrete resultaat van OpenAjax is de definitie van de OpenAjax Hub. De hub is een centraal event-mechanisme, vergelijkbaar qua functie met Glue in jMaki, dat client-side widgets onderling laat 'communiceren' . De volgende versie van Hub zal interactie tussen meerdere frames en tussen client en server gaan beschrijven. Ook zullen voorzieningen voor 'secure mash-up's' (SMash) worden beschreven. Een ander project onder OpenAjax is MetaData, een standaard voor de beschrijving van widgets, zodat ontwikkelaars en tools op een uniforme wijze met widgets van verschillende libraries en leveranciers aan de slag kunnen.
De ambities en plannen van OpenAjax zijn veelbelovend, net als de lijst van partijen die zich bij OpenAjax hebben aangesloten. De druk vanuit klanten en gebruikers neemt ook flink toe – waarom kunnen we jullie library niet combineren met die 'coole widgets' of ' toffe component' ? Als je de vraag nu stelt aan een leverancier is het standaard antwoord: OpenAjax zal dat regelen. Het is alleen op dit moment moeilijk te voorspellen wanneer ondersteuning van de OpenAjax standaarden wijd verbreid zal zijn. Eerder dan in de loop van 2009 valt dat in elk geval niet te verwachten. In de tussentijd en als overgang naar OpenAjax lijkt jMaki een aantrekkelijke optie.

JavaServer Faces 2.0

Naast deze geschetste ontwikkelingen wordt in het kader van de JEE 6 release – voorzien voor begin 2009 – gewerkt aan de ontwikkeling van de JSF 2.0 standaard. In een presentatie door de "spec leaders" Roger Kitain en Ed Burns werd uiteengezet welke thema's de kern vormen van deze 2.0 release. Ze stelden nadrukkelijk dat een standaard specificatie, zoals JSF, in principe bedoeld is voor het oogsten van best practices, aangeslagen innovaties en gemeenschappelijke elementen in aanpak. JSF 1.0 was hierop een enorme uitzondering: daar ging het vooral om zaaien en planten van een tamelijk nieuwe aanpak, ondanks het feit dat JSF bouwde op de fundamenten van ondermeer Struts en jarenlange ervaring met JSP en MVC. JSF 2.0 moet veel meer het vastleggen worden van wat de afgelopen jaren in de meer dan een dozijn verschillende JSF-bibliotheken aan innovatie is gedaan.
De belangrijkste doelen voor JSF 2.0 werden geschetst als: sterk vereenvoudigen van het ontwikkelen van JSF-componenten door iedere ontwikkelaar zelf, ondersteuning voor AJAX (zodat een gemeenschappelijke oplossing voor de AJAX-requests en gedeeltelijke paginaverversingen in de core JSF life-cycle wordt ingebouwd, in samenwerking met het OpenAjax-initiatief en mede geïnspireerd door DynaFaces, eenvoudiger Page Description Language (PDL) en loslaten van JSP als standaard voor paginadefinities (en adoptie van Facelets), configuratie eenvoudiger maken met ondermeer annotaties voor Validators, Converters en Managed Beans. Een vijfde hoofddoel is het verbeteren van de compatibiliteit tussen de JSF-implementaties van verschillende leveranciers.
Daarnaast zijn andere aandachtsgebieden benoemd in de JSF 2.0 plannen. Bookmarkable url's, betere foutboodschapafhandeling, betere resource-delivery (CSS, Images, JavaScript) als aparte stap in de life-cycle, nieuwe scopes, viewScope (tussen request en session in, lijkt op JSP page scope) en componentScope, interactie met Portal 2.0 en WebBeans.

Social

Tot de meest succesvolle thema's op internet van de afgelopen jaren hoort ontegenzeggelijk het 'social networking'. Sites als Hyves, LinkedIn, Digg It, Orkut, Facebook, FLickr, MySpace, YouTube, Twitter, maar ook Blogs, Wikis en Forums worden intensief gebruikt voor het leggen en onderhouden van contacten, samenwerken en delen van allerlei soorten informatie, taggen en beoordelen, communiceren via diverse kanalen en publiceren van allerhande personalia. Op initiatief van Google heeft een flink aantal van de vooraanstaande social networks en leveranciers van standaardpakketten de OpenSocial API vastgesteld, waarmee interoperabiliteit van de social networks wordt bereikt.

http://code.google.com/apis/opensocial/

De OpenSocial API kent verschillende aspecten:

- Een JavaScript API en de OpenSocial Client architectuur. Hiermee wordt het voor ontwikkelaars mogelijk om Social Gadgets te ontwikkelen. Dit zijn iGoogle-achtige componenten die onderdeel kunnen worden gemaakt van Social WebSites. Bijvoorbeeld: een ontwikkelaar maakt een gadget voor een dienst 'wie gaat met mij mee op vakantie'. Als dit gadget gebruikmaakt van de OpenSocial API kan het worden ingebed in sites als Facebook en MySpace en kan het binnen die sites communiceren met beschikbare services voor Vrienden en Contacten, Activiteiten en persistence.
- Een RESTFul API. Eenvoudige webservices die programmatische toegang bieden tot dezelfde data in sociale websites die ook beschikbaar is in de Javascript API. Met deze services kan je in je eigen website en webapplicatie data en basisvoorzieningen van LinkedIn, Flickr en andere sites gebruiken.

Om je eigen Social Applicatie en Server te ontwikkelen, of een goede ontwikkelomgeving in te richten en de OpenSocial APIs in actie te zien, kan je gebruikmaken van Apache Shindig. Dit is een Apache Incubator-project met grote betrokkenheid van ondermeer partijen

als Google – dat de iGoogle server heeft ingebracht, Ning en Hi5. Shindig beoogt een OpenSocial Container out-of-the-box te zijn. Shindig implementeert de Social Gadget Server (op basis van iGoogle-technologie), de OpenSocial Javascript API's en de Server Side RESTful API.
Je kunt jouw eigen database met 'social data' of social data-services koppelen aan Shindig. Zie http://incubator.apache.org/shindig/.
Sun presenteerde tijdens JavaOne ook zijn eigen Project SocialSite (https://socialsite.dev.java.net/) dat zowel een aanvulling op als een alternatief voor Shindig lijkt te willen zijn. Doelstelling is om Social Networking eigenschappen toe te voegen aan Java maar ook PHP en Ruby applicaties.

Actief

WebiInterfaces worden niet alleen steeds rijker – fraaier, met coole widgets en bewegende visuele presentaties – ze worden ook steeds actiever. Een actieve UI is een interface – bijvoorbeeld een dashboard – die ververst zonder actie van de gebruiker. De server stuurt berichten naar de client die door de browser worden opgevangen en gebruikt voor instantane updates van onderdelen van de pagina.
Er zijn tenminste twee categorieën te definiëren voor dit soort server-to-client updates:

- Near real-time zoals beurskoers, biedingen op een veiling, chat-berichten, data locks en wijzigingen, en alerts (server outage, inbraak en brand) en
- Vertraagd, bijvoorbeeld RSS, temperatuur en verkeer.

In het verleden werden beperkte vormen van 'actieve update' geïmplementeerd met client-side polling-mechanismen, waarbij om de paar seconden een hernieuwd – asynchroon request naar de server werd gestuurd, om te vernemen of er mogelijk data-updates beschikbaar waren. Server-geïnitieerde push is een nieuwe aanpak die aan alle kanten de kop op steekt. Hierbij is het de server die events of nieuwe data aan de client meldt – en worden er dus niet onnodig requests uitgevoerd. Voor deze manier van werken zijn diverse aanpakken ontwikkeld – Comet is de verzamelnaam, hoewel ook wel eens over *Pushlets* wordt gesproken – die alle schaalbaarheiduitdagingen kennen. De server moet op de een of andere manier http-connecties open houden met de client – aangezien echte server-push niet bestaat; deze aanpak wordt streaming genoemd. De andere aanpak wordt Long Polling genoemd. De client doet een AJAX-request dat de server pas beantwoord als er een te rapporteren event beschikbaar is, waarna de client direct weer een request doet. Vanuit het Dojo-project is het Bayeux-protocol ontwikkeld, een implementatie van het Server-Push-concept dat eenvoudig te gebruiken is, ondermeer met Dojo client-side componenten.

Asynchronous Request Processing (ARP) is de wonderolie die de schaalbaarheidproblemen gaat oplossen, in plaats van dedicated threads te gebruiken voor alle requests. In Servlet 3.0 en de nieuwe NIO-libraries met ondermeer Continuations komt ondersteuning voor het asynchroon afhandelen van requests. Requests kunnen ondermeer tijdelijk bevroren en later weer hervat worden. Sommige webservers – bijvoorbeeld Jetty en Glassfish – hebben nu al ondersteuning voor Comet-style Server Push-applicaties. In Glassfish wordt die ondersteuning geïmplementeerd met behulp van Project Grizzly – een inmiddels stand-alone library (zie https://grizzly.dev.java.net/) die ook buiten de context van Glassfish kan worden toegepast, gebaseerd op de J2SE NIO-libraries. Server Push voor actieve user-interfaces is inmiddels beschikbaar in een aantal producten, waaronder Dojo, ICEFaces, Backbase en ADF Rich Faces. Met behulp van Grizzly en over enige tijd met standaard Servlet (3.0) is eenvoudig toepasbare en robuust schaalbare server-client push-functionaliteit beschikbaar in iedere webapplicatie.

Mash-uUp

Als het over services gaat – en daar hebben we het tegenwoordig nogal eens over – is de term mash-up meestal niet ver weg. Mash-up kan je zien als het combineren van services zodanig dat het geheel meer is dan de som der delen. De services verrijken en versterken elkaar onderling. Of het nu de makelaar is die de verhuizing regelt, het restaurant dat de taxi verzorgt, of dat het een bundeling van softwarematige services is, mash-up's geven individuele services-meerwaarde aan de gebruiker.
De mash-up in softwaretermen gaat vaak ook gerelateerde gegevens. Eén service geeft aanvullende informatie over de data gepresenteerd door een andere service. Geografische informatie van Yahoo Maps of Google Maps speelt daarbij vaak een rol. Bijvoorbeeld: vanuit de ene service wordt een reisbestemming gekozen, de andere service toont die bestemming op een kaart, terwijl andere services het lokale weerbericht, een activiteitenagenda van lokale VVV en de top vijf van regionale restaurants tonen.
Een Mash-up als combinatie van softwareservices komt vaak tot uitdrukking in een portal-achtige pagina, hoewel het woord portal een beetje belast lijkt te zijn door niet ingeloste hooggespannen verwachtingen uit het recente verleden. JavaOne liet twee implementaties zien van de mash-up, de een klassiek – server-side - de ander wat vernieuwender – client-side.

Server Side - Portal

Een portal is een server-side 'up-masher'. De portal-server verzamelt de bijdrage van alle geconfigureerde portlets – zowel lokale als remote portlets (die op basis van WSRP 2.0 door remote portlet-containers worden aangeboden). Een portlet is een webapplicatie die zich via een bepaald contract laat configureren en bij aanroep html rendert die zich aan bepaalde afspraken houdt, zoals geen <HTML> en <BODY> tag. De standaard die deze interface en de configuratie van portlets beschrijft heet JSR-286 en is in maart van dit jaar definitief vastgesteld. De referentie implementatie is Apache Pluto (http://portals.apache.org/pluto/).
Redenen dat de ontwikkeling van 'klassieke' portals in de afgelopen jaren is gestagneerd – naast de complexiteit van diverse portal-producten – zijn ondermeer het ontbreken van serieuze AJAX-ondersteuning en vooral de gebrekkige interactie tussen portlets onderling. De meerwaarde van een portal voor het bouwen van een mash-up zou voor een belangrijk deel moeten komen uit het bieden van een infrastructuur waarmee portlets data kunnen delen en elkaar van events op de hoogte kunnen stellen (zodat synchronisatie kan plaatsvinden). Portlets op basis van JSR-168, de Portlet 1.0 specificatie, kunnen dat niet.
JSR-286, ook wel Portlet 2.0 genoemd, brengt een aantal nieuwe elementen.

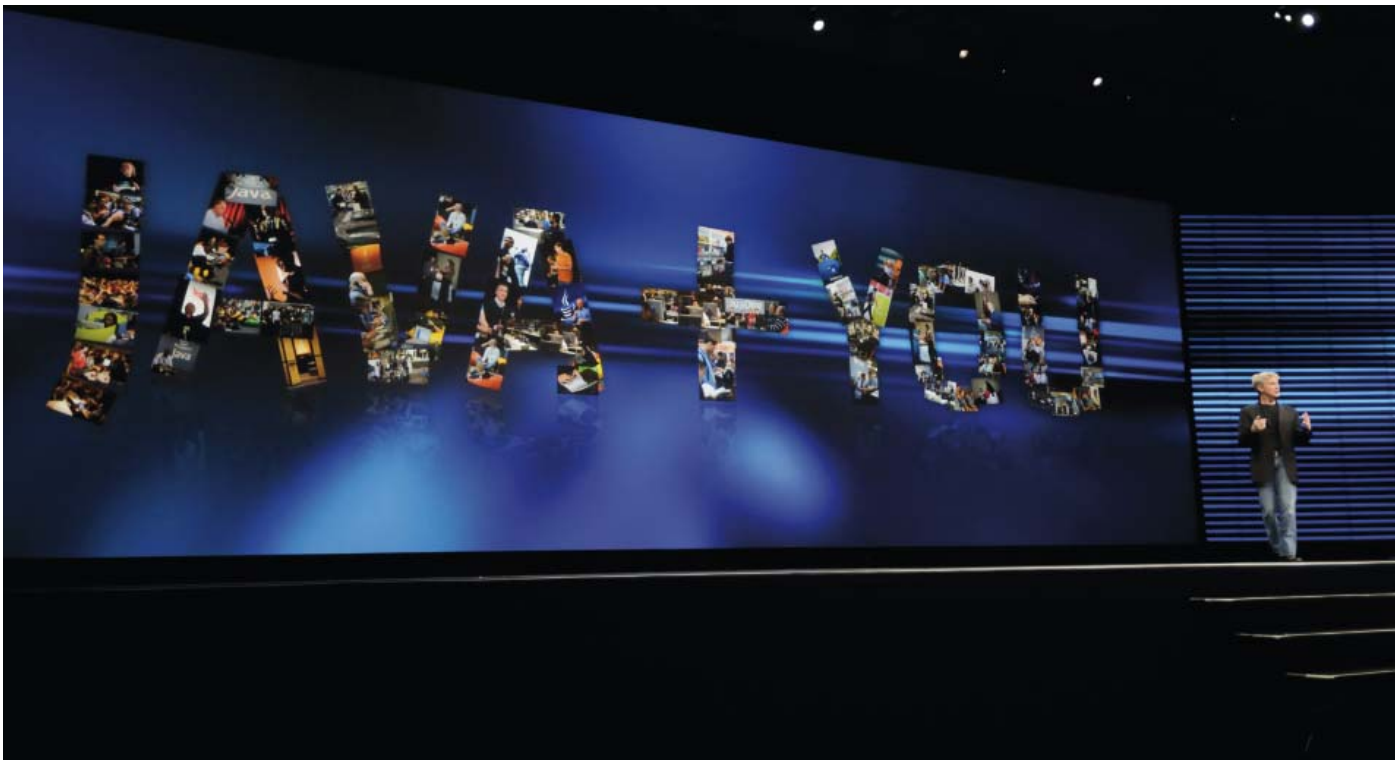


Foto: Dré de Man

Ondersteuning voor events is er een. Portlets kunnen aangeven dat ze events publiceren en/of events willen consumeren.Het portal-framework zorgt er voor dat een event aan alle portlets wordt gemeld alvorens de render-phase voor die portlets start. JSR-286 beschrijft ook public parameters die door alle portlets kunnen worden gedeeld. Bijvoorbeeld een locatiecoördinaat of de code van het geselecteerde product.

Een belangrijke verbetering is ook het mechanisme voor het downloaden van resources. JSR-286 ondersteunt het downloaden door de portlet zelf, met het introduceren van een nieuwe lifecycle-methode. Zo kunnen portlets nu ook html-headers en cookies zetten. Een ander belangrijk element is ondersteuning voor AJAX. De AJAX-calls worden onder JS-286 via de portal-server naar de portlet doorgesluisd. De portlet heeft nu ook tijdens een AJAX-request toegang tot de portlet state. JSR-286 is nauw gelieerd aan WSRP 2.0 – de standaard voor Remote Portlets – maar voor portlet-ontwikkelaars of portal-beheerders is het transparant of een portlet lokaal of remote is.

Naast de Apache Pluto referentie-implementatie komen er op korte termijn implementaties van JSR-286 in ondermeer Oracle BEA WebLogic Portal, IBM WebSphere Portal en Oracle WebCenter.

De belangrijkste open source implementatie is hoogstwaarschijnlijk Liferay. Zeker na de aankondiging van Sun tijdens JavaOne 2008 dat OpenPortal – het open source project gebaseerd op Sun Java System Portal Server – wordt opgevolgd door WebSynergy, een op Liferay gebaseerde productfamilie. Sun gaat actief bijdragen aan de ontwikkeling van Liferay – als committer en medebestuurslid – en op termijn zullen alle OpenPortal-projecten (waaronder de JSF Portlet-bridge en het Mirage CMS) in Liferay opgaan.

Cliënt-side

Een mash-up kan client-side tot stand worden gebracht. In plaats van portlets die min of meer zelfstandige webapplicaties zijn die door een portal-server worden gecoördineerd en gecombineerd, kan ook gebruik worden gemaakt van client-side widgets – rijke Javascript-componenten – die vanuit de client zelfstandig dataservices benaderen. iGoogle is een voorbeeld van zo'n client-side mash-up. JavaOne 2008 toonde een groot aantal voorbeelden van dit soort widgets die gebruikmaken van JSON, ATOM, RSS of gewone XML-services om hun data te verkrijgen of te manipuleren. Deze widgets kunnen onderling met bijvoorbeeld jMaki Glue of de vanuit OpenAjax te ontwikkelen event-bus communiceren.

Conclusie

Andere artikelen in dit Java Magazine bespreken andere onderwerpen die tijdens JavaOne 2008 aan bod kwamen. Ik wil hier afsluiten met een kort overzicht van de kreten, termen en projecten die mij als opvallendst zijn bijgebleven. Een kleine kanttekening is hierbij wel op zijn plaats. Enerzijds is deze conferentie de wereldtentoonstelling op het gebied van Java. Aan de andere kant is het gevaarlijk JavaOne als enige maatstaf te nemen voor wat belangrijk is. Sun heeft een wel erg grote invloed op de inhoud van het programma en ook duidelijke eigen belangen bij het wel of niet toelaten van sprekers en sessies. Projecten van Sun krijgen mogelijk meer aandacht dan strikt genomen gerechtvaardigd is gezien hun huidige status in de community, terwijl belangrijke ontwikkelingen met grote aanhang mogelijk geen of te weinig zendtijd krijgen op de conferentie.
Dat gezegd hebbend, zijn dit de onderwerpen die mij het meest opvielen: Hudson (Continuous Integration Server), jMaki, Comet/Server Push, WebBeans, OpenPortal/Liferay, Groovy, OpenSocial API, Applet, OSGi, EclipseLink, NetBeans, Glassfish, JSON (ATOM, RSS), JMX en de ontwikkelingen in Java (SE) 7. «

Lucas Jellema
AMIS Sevices