Enterprise

'Never a dull moment' in de wereld van Java. Op die regel is ook 2009 zeker geen uitzondering. Neem nou de overname van Sun Microsystems door Oracle. Eén van de ontwikkelingen die ook al enige tijd zijn schaduw vooruit werpt en in september 2009 tot voorlopig hoogtepunt komt is JEE 6, de nieuwste release van de Java Enterprise Edition. Java Magazine besteedt dit hele jaar aandacht aan de belangrijke componenten in JEE 6 - met in dit nummer de schijnwerper op JSF 2.0, de tweede major release van JavaServer Faces, die in mei 2009 werd vastgesteld.

Het jaar van JEE6 (3)

Java Server Faces 2.0

Lucas Jellema
is Java & SOA specialist
bij AMIS.
(lucas.jellema@amis.nl)

JSF 1.0 zag in 2004 het daglicht en in 2006 maakte JSF 1.1 deel uit van de JEE 5 release. JSF zette eindelijk een standaard View & Controller technologie voor web applicaties neer. Het definiëren van die standaard heeft geleid tot een snelle ontwikkeling van allerlei libraries met veelal rich JSF componenten die component-gebaseerd-web-applicatie-ontwikkelen naar een nieuw niveau bracht. En tot een groeiende groep ontwikkelaars met JSF vaardigheden die van toepassing zijn met iedere JSF bibliotheek, of het die van Sun, JBoss, Oracle, Apache of IBM is.
JavaServer Faces 2.0 is de eerste 'major update' van JSF, hoewel de 1.2 release in 2006 al de nodige aanscherpingen introduceerde. De spec-lead - Ed Burns (van Sun) - stelde dat hij nadrukkelijk niet innovatie nastreefde met de 2.0 release, maar wel oogsten en consolidatie: verwerken in de specificatie wat er aan innovaties rondom JSF in de markt is ontstaan in de afgelopen jaren.
Belangrijke nieuwe voorzieningen en voorbeelden van dat oogstwerk in JSF 2.0 zijn de prominente rol van Facelets - in plaats van JSPs - als de manier van pagina's beschrijven, voorzieningen voor AJAX, eenvoudiger configuratie, ondersteuning voor resources als plaatjes en JavaScript, een eenvoudiger manier om nieuwe componenten te ontwikkelen en diverse nieuwe standaardcomponenten. In dit artikel kijken we in maar detail naar de meest opvallende zaken in JSF 2.0.

JSF 2.0 en JEE 6

JSF 2.0 bevat verbeteringen van en correcties op eerdere ideeën en keuzes, volledig nieuwe functionaliteit en verbeteringen in bestaande en nieuwe JEE 6 specificaties zoals de Servlet API, Java Contexts and Dependency Injection, voorheen bekend als WebBeans, en Bean Validation.
JSF 2.0 applicaties kunnen worden gedeployed op een JEE 5 Application Server of iedere server die de Servlet 2.5 API implementeert. Verder is JSF 2.0 volledige backwards compatible: bestaande JSF 1.x applicaties blijven werken in servers en met bibliotheken die de 2.0 release ondersteunen.

Facelets

Facelets 2.0 - XHTML pagina's met naast de gewone HTML elementen ook tags voor de JSF componenten - is de aangewezen manier om JSF pagina's te beschrijven. JSPs blijven ondersteund als View-implementatie maar veel nieuwe features in JSF 2.0 komen niet beschikbaar voor JSF pagina's die met de JSP tags zijn gedefinieerd. Op termijn is een volledige migratie van JSP naar Facelets sterk aanbevolen. Met Facelets-technieken zoals XHTML templates is de ontwikkeling van custom componenten aanzienlijk eenvoudiger dan in JSF 1.x.

AJAX in JavaServer Faces

Een set uniforme AJAX-faciliteiten is toegevoegd aan de JSF standaard, geïnspireerd door prominente libraries zoals (JBoss) RichFaces & Ajax4JSF, ADF Faces & Trinidad, ICEFaces en Dynamic Faces.
Een belangrijk doel van de AJAX-ondersteuning is ook bevordering van de (nu flink tekort schietende) interoperabiliteit: iedere JSF bibliotheek die AJAX ondersteuning biedt voor haar componenten kan van hetzelfde onderliggende mechanisme gebruikmaken. De tijd van conflicterende JavaScript code en botsende JSF lifecycle handlers zou daarmee achter ons moeten liggen.
Er is slechts één AJAX component in JSF. Dat kan op twee manieren worden gebruikt: als kind van een component dat een AJAX request moet kunnen initiëren of als wrapper om een groep componenten.

```
<h:inputText id="it1" value="#{mijnbean.waarde}">
   <f:ajax render="@this"  />
</h:inputText>
```

Hier staat gespecificeerd dat als de waarde in de it1 inputText component wijzigt - als het JavaScript onChange event wordt afgevuurd - een AJAX-request moet worden uitgevoerd.
Dit is zo'n beetje de meest compacte toepassing van deze tag, ondermeer vertrouwend op een aantal default instellingen. Het event attribute wordt gebruikt om aan te geven welk event een AJAX operatie moet starten. Ieder component kent een default event voor AJAX-requests - bijvoorbeeld click voor buttons en change voor input velden.
De JSF lifecycle wordt ingedeeld in twee fasen: de execute fase die bestaat uit Apply Request Values, Process Validations en Update Model en de render phase die alleen de Render Response omvat. In geval van een AJAX request kan een selecte groep componenten door de execute phase worden verwerkt, terwijl een mogelijk andere groep componenten wordt gerenderd. JSF 2.0 introduceert hiervoor het concept van Partial View Processing, dat wordt toegepast voor AJAX maar eventueel ook in andere situaties.
De beide fasen zien we terug in onderstaand codevoorbeeld. Tijdens de execute fase wordt uitsluitend het component it1 verwerkt - we hadden hier execute kunnen weglaten aangezien @this de default waarde is voor dit attribuut - (en dus ook alleen het waarde property van managed bean mijnbean gevalideerd en geupdate) terwijl in de render fase naast it1 ook de componenten met id waarden eenAnderComponentId en enNogEenComponent worden gerenderd en in de client geupdate.

```
<h:inputText id="it1" value="#{mijnbean.waarde}">
    <f:ajax event="change" execute="@this" render="@this
eenAnderComponentId
enNogEenComponent"
              listener="#{bean.
listenerToExecuteDuringAJAXProcessing}"
              onEvent="myOwnJavaScriptFunctionToBeCalle
dOnBeginCompleteAnd
SuccessofAJAXRequest"
   />
</h:inputText>
```

Het listener attribuut wordt gebruikt om een server side methode te configureren die voor de update model stap wordt aangeroepen. Het attribute onEvent specificeert een JavaScript functie die aan het begin en eind van het AJAX request en na succesvolle update van de DOM wordt aangeroepen.
Het antwoord op een AJAX request bevat informatie om een klein deel van de pagina te verversen, via DHTML (JavaScript) manipulatie van het browser DOM (Document Object Model). Die operatie is ingebouwd in de nieuwe client side (JavaScript) onderdelen van het JSF framework.

De JSF lifecycle wordt ingedeeld in twee fasen: execute en render

Resource Handling

JSF-componenten maken veelal gebruik van JavaScript en vaak ook van plaatjes en soms CSS stylesheets. Idealiter zijn deze resources met de componenten zelf verpakt in JAR files. Als de componenten in JSF pagina's worden toegepast moeten de resources door browsers geladen worden. Uitdagingen die daarbij spelen: de resources moeten gevonden kunnen worden - ook in JARs- , resources moeten niet vaker dan één keer geladen worden en er is behoefte aan ondersteuning voor meerdere versies van resources naast elkaar - ook verschillende varianten voor verschillende locales.
JSF 2.0 biedt een standaardfaciliteit voor resources, ingebouwd in het framework waardoor bijvoorbeeld geen apart servlet filter geconfigureerd hoeft te worden. Resources kunnen uit de web applicatie (onder /resources) worden geladen maar ook uit JAR files (/META-INF/resources). JSF componenten en render-classes en ook validators en converters kunnen met een annotatie (@ResourceDependency) aangeven dat wanneer zij in een pagina worden gerenderd, het framework moet zorgen dat de aangegeven resource in de browser geladen wordt. In de pagina's zelf kan met de tags graphicImage, outputStylesheet en outputScript worden aangegeven dat resources moeten worden geladen:

```
<h:graphicImage          name="superModel.png"
library="prettyFaces"/>
<h:graphicImage value="#{resource['superModel.png']}" />
<h:outputScript  name="catwalk.js"  library="amis.fashion"
target="head"/>
```

In outputScript wordt met het target attribuut aangegeven of het script in de HTML head of body sectie of in een bepaald form moet worden geladen.
De ResourceHandler (pluggable zoals vrijwel alle faciliteiten in JSF) bepaalt op basis van de aangegeven resource identifier en de huidige locale welke file daadwerkelijk aan de browser moet worden teruggegeven. De ResourceHandler geeft altijd de laatste versie van een resource. Dit betekent dat we nieuwe versies van resources als plaatjes, stylesheets en JavaScript libraries kunnen hot-deployen, terwijl de applicatie in lucht blijft.

Bookmarking van JSF pagina's

JSF 2.0 introduceert View Parameters, geinspireerd door de Seam PageParameters. Een belangrijk doel van View Parameters is ondersteuning voor GET requests naast de tot nu toe min of meer heilige POSTs en daarmee het effenen van de weg voor het bookmarken van pagina's in JSF applicaties. Een f:viewParam wordt binnen de body van een pagina gedefinieerd.
Nieuwe componenten - h:link en h:button - doen een GET request, in plaats van de POST die door commandLink en commandButton wordt uitgevoerd. In het GET request kunnen de waarden van parameters (f:param) worden meegestuurd, maar ook parameters die in de navigation rules. Als de pagina waar de navigatie naar wijst view parameters definieert, worden die parameters ook in de URL opgenomen.

Bijvoorbeeld:

```
<h:link outcome="editEmployee" value="Edit">
  <f:param name="lastName" value="#{emp.lastName}"/>
</h:link>

met in de faces-config.xml een navigatie regel:

<navigation-rule>
    <from-view-id>*</from-view-id>
        <navigation-case>
            <from-outcome>editEmployee</from-outcome>
             <to-view-id>/employeeEditor.xhtml</to-view-
id>
            <redirect/>
        </navigation-case>
</navigation-rule>

En in de doel-pagina employeeEditor.xhtml de view
parameter employeeId:

        <f:metadata>
      <f:viewParam id="employeeIdParam" name="employeeId"
value="#{employeeBean.id}" />
                          <f:event type="preRenderView"
listener="#{employeeEditor.prepareModelForRendering}" />
         </f:metadata>
```

Is de URL die door de h:link wordt samengesteld - en die dus ook als bookmark kan worden gebruikt - als volgt opgebouwd:

```
http://localhost:8080/myapp/employeeEditor.xhtml?lastNam
e=Jansen&employeeId=42
```

Hierbij zijn de waarden Jansen en 42 het resultaat van de evaluatie van EL expressies #{emp.lastName} en #{employeeBean.id} op het moment van renderen van de pagina en waar de pagina zelf (employeeEditor.xhtml) is bepaald door het outcome attribuut van de link component op basis van de navigation rules alvast te evalueren.
Op een pagina kan een event listener worden geregistreerd voor het preRenderView event. Deze listener wordt aangeroepen nadat de view parameters zijn verwerkt maar voordat de view daadwerkelijk wordt gerenderd. Dit is een uitstekend moment om de data context die nodig is tijdens het renderen klaar te zetten op basis van ondermeer de parameterwaarden.

Validatie

Default Validators kunnen worden geconfigureerd in de faces-config.xml of via een annotatie. Een default validator wordt altijd toegepast op een input-component, tenzij expliciet uitgesloten. Nieuwe standaard validators zijn RegexValidator en BeanValidator Als de BeanValidator geconfigureerd is en Beans Validation (JSR-303) is beschikbaar, dan worden managed beans gevalideerd als ze via EL expressies in het value attribute van UIInput componenten worden gerefereerd. Bijvoorbeeld:

```
<h:inputText id="it1" value="#{eenofanderebean.waarde}"
/>
```

En het volgende fragment in de Java class onder de bean eenofanderebean:

```
@NotNull
@Max(50)
public void setWaarde( Integer value) {
   this.waarde = value;
}
```

In de Validatie stap in de JSF lifecycle worden de validaties uitgevoerd die beschreven zijn door de @NotNull en @Max annotaties voor de waarde die is ingevoerd in de it1 component.

Unified Expression Language

De Unified Expression Language is een belangrijke specificatie in JEE 6. Verbeteringen in EL zijn direct beschikbaar in JSF. Een prominent voorbeeld van zo'n verbetering is de ondersteuning voor het aanroepen van methodes anders dan accessors. Hierdoor worden dit soort expressies mogelijk:

```
<h:outputText     value="#{bean.helloWorldBegroeting
('Hendrik Jan','Tuinman')}" /><br />
<h:outputText value="#{bean.namenLijst.size()}" /
```

Er wordt ook gewerkt aan een syntax voor method-Expres-sions die parameters aankunnen.

Overige nieuwe features

Een beknopte opsomming van enkele andere waardevolle nieuwe faciliteiten in JSF 2.0:

- Vereenvoudiging van faces-config.,xml en het gebruik van annotaties - inclusief EJB injectie inmanaged beans, een @ManagedBean annotatie waarmee classes als managed bean kunnen worden geregistreerd en annotaties voor de registratie van Componenten, Renderers, Validators, Converters en PhaseListeners.
- Een centrale, pluggable Exception Handler die alle exceptions ziet langskomen en kan afhandelen.
- Ondersteuning voor System Events waarop de ontwikkelaar listeners kan registreren. Dit mechanisme is een verfijnde uitbreiding op wat er nu al kan met PhaseListeners. Voorbeelden van system events: PostConstructApplicationEvent, PostRestoreState, PreValidate, PostValidate, PreRender en PreRenderView
- Nieuwe scopes voor het vasthouden van managed beans: de view scope - deze scope blijft in leven zolang dezelfde view gerenderd blijft (over AJAX requests en postback aanroepen heen). De flash scope scope overleeft de navigatie naar een andere pagina, maar verliest dan zijn inhoud.

Conclusie

JSF 2.0 is misschien geen revolutie, maar wel een ingrijpende uitbreiding van JavaServer Faces. JSF 2.0 beschrijft een uniforme aanpak voor AJAX, die voor ontwikkelaars straks in alle implementaties herkenbaar en toepasbaar zal zijn en die hopelijk tot betere interactie en onderlinge verdraagzaamheid leidt tussen JSF implementaties.
Andere opmerkelijke vernieuwingen in de JSF 2.0 release: gebruik van Facelets, integratie met Bean Validation, nieuwe scopes voor managed beans, verbeteringen in de configuratie met ondermeer annotaties, ondersteuning voor GET requests en het bookmarken van JSF pagina's en de slimme ResourceHandler. «

JSF 2.0 is geen revolutie, wel ingrijpende uitbreiding

- De JSR-314 specificaties voor JSF 2.0 kan je nalezen via: http://www.jcp.org/en/jsr/detail?id=314. Project Mojarra, de referentie-implementatie van JavaServer Faces 2.0, is beschikbaar op: https://javaserverfaces.dev.java.net/.
- De codevoorbeelden en hyperlinks die in dit artikel zijn opgenomen alsook enige aanvullende resource -verwijzingen kan je terugvinden om zelf ook te proberen op http://technology.amis.nl/blog/6062/javaserver-faces-20-next-generation-jsf-technology-specification.

NIEUWE WORKSHOP! 13 tot en met 15 oktober 2009 – Holiday Inn Leiden

Mastering the Requirements Process

Unieke driedaagse workshop met internationaal gerenommeerde topspreker Suzanne Robertson

Over requirements bestaan de nodige misverstanden. Dat is jammer, want systeemontwikkeling kan niet zonder. Als u wilt dat de energie die in de ontwikkeling wordt gestoken zich daadwerkelijk uitbetaalt, móéten de requirements kloppen. In deze workshop gaat u aan de slag met een compleet proces voor het aan het licht brengen van de echte requirements, het testen van de juistheid ervan en het vastleggen van requirements op een manier die niets aan duidelijkheid te wensen overlaat.

Met deze workshop ontwikkelt u de volgende vaardigheden:

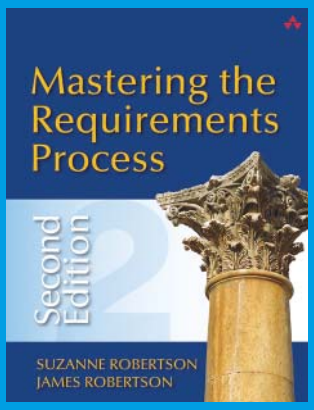
- u kunt de behoeften van uw klant tot op de millimeter nauwkeurig vaststellen,
- u ontdekt wat de échte business is en hoe deze verbeterd kan worden,
- u kunt requirements schrijven die compleet, herleidbaar en testbaar zijn,
- u leert de scope van het project af te bakenen,
- u kunt alle belanghebbenden traceren – en u houdt ze betrokken,
- u leert gebruik te maken van moderne technieken als storyboarding, prototypes en wiki's,
- u leert snel tot de juiste requirements te komen.

Bestemd voor ú

Deze workshop is echt iets voor u als u betrokken wilt zijn bij het opleveren van goede systemen – dat wil zeggen, systemen die ook echt gebruikt gaan worden. Waarschijnlijk bent u businessanalist, systeemanalist, projectleider, requirementsbouwer, consultant, productmanager of programmamanager. Maar deze workshop is ook interessant voor gebruikers of softwareafnemers die er zeker van willen zijn dat het requirementsproces hun precies datgene oplevert wat ze nodig hebben. Meld u snel aan: er is slechts plaats voor 24 deelnemers!

Het IIBA, het International Institute of Business Analysis, heeft vastgesteld dat de cursus *Mastering the Requirements Process* in overeenstemming is met versie 2.0 van de BABOK. De cursus wordt gegeven door het Atlantic Systems Guild, een door het IIBA erkende opleider.

IIBA Endorsed Education Provider



GRATIS BOEK VOOR DEELNEMERS

Slechts 24 deelnemers kunnen deelnemen aan deze workshop. Iedere deelnemer ontvangt een exemplaar van het boek Mastering the Requirements Process. Dit standaardwerk is geschreven door Suzanne en James Robertson. Alleen diegenen die deze workshop gevolgd hebben, kunnen volgend jaar deelnemen aan het vervolg.

Kijk snel op www.arrayseminars.nl voor het complete programma!

Array SEMINARS

in samenwerking met COMPUTABLE