UIX in de praktijk bij CertiQ/TenneT

Project met JDeveloper en ADF

CertiQ/TenneT gebruikt een webapplicatie voor de afhandeling van certificaten voor duurzame elektriciteit, deze was gebouwd met web-PL/SQL. In het Accelerated Development Center (ADC) van Capgemini in Utrecht is deze applicatie, met behulp van JDeveloper en ADF (UIX en Business Components), in twee maanden tijd gemigreerd naar Java. Tevens is in deze tijd een ontwikkelteam opgeleid om het onderhoud van de applicatie te kunnen doen. Dit artikel is een verslag van deze ontdekkingsreis.

TenneT vervult voor de Nederlandse samenleving een spilfunctie in de elektriciteitsmarkt. Zij heeft een aantal belangrijke taken toebedeeld gekregen, gericht op het realiseren van een efficiënte marktwerking en transparantie. Op grond van de Nederlandse Elektriciteitswet is TenneT aangewezen als de onafhankelijke landelijk netbeheerder om het elektriciteitsverkeer in goede banen te leiden: In 2001 heeft TenneT de dochtermaatschappij CertiQ opgericht. CertiQ richt zich op het uitgeven van certificaten voor op milieubewuste wijze opgewekte energie, waaronder certificaten voor duurzame elektriciteit. Aan elektriciteit die uit een stopcontact komt, is niet te zien hoe het is geproduceerd. Om toch onderscheid te kunnen maken tussen milieubewust opgewekte elektriciteit en 'gewone' elektriciteit, heeft de overheid productiecertificaten ingevoerd. Certificaten vormen een 'garantie van oorsprong'. Ze zijn het bewijs dat elektriciteit milieubewust is opgewekt. CertiQ beheert het systeem waarin de certificaten worden uitgegeven. Onderdeel van dit systeem is een externe website waarop klanten van CertiQ kunnen inloggen om vervolgens handelingen te verrichten die te maken hebben met het verkrijgen en gebruiken van certificaten.

Web-PL/SQL

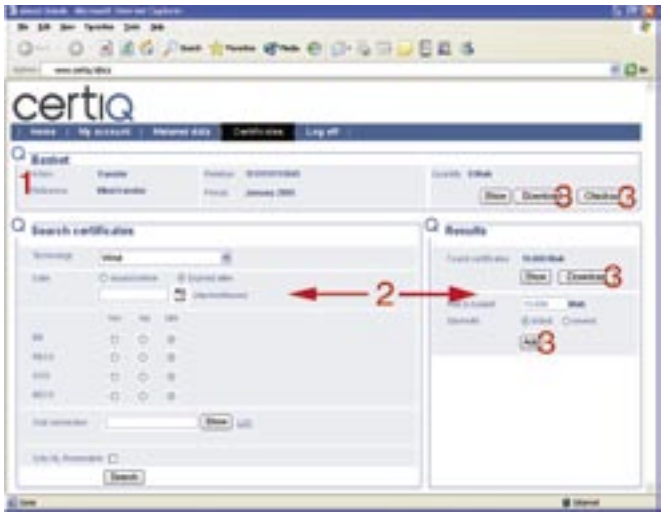
De externe website was gebouwd met web-PL/SQL, gegenereerd uit Designer. Binnen CertiQ liepen projecten die de gebruikersvriendelijkheid en de onderhoudbaarheid van de externe applicatie moesten verbeteren. De kennis van web-PL/SQL is niet breed voorhanden. Daardoor ontstond de wens om een veelgebruikte technologie te kiezen. Binnen Tennet was al ervaring opgedaan met Java en JDeveloper voor intranet applicaties. Java werd ook beschouwd als een goed alternatief voor de externe applicatie.

Besloten werd om volgens het 'Model View Controller' pattern te gaan werken met JDeveloper (9.5.0.2) met het Oracle Advanced Developer Framework (ADF).
De keuze viel op UIX als View, omdat met UIX ervaring is binnen TenneT en omdat er meer standaardcomponenten aanwezig zijn die de usability konden verbeteren waardoor een hogere productiviteit kon worden behaald. Dat de ervaring met UIX gebruikt is in de vaststelling van Java Server Faces gaf genoeg vertrouwen voor de toekomstvastheid van UIX. Voor de Controller werd de standaard van ADF gekozen: 'Struts'. Vervolgens restte de keuze tussen Toplink of ADF Business Components (voorheen BC4J) voor het Model. Toplink lijkt een goede keuze als een object-georiënteerd ontwerp is gemaakt en objecten moeten worden opgeslagen in de database. De externe applicatie moest op basis van een 'relationeel' ontwerp worden gekoppeld met de database. Hiervoor is ADF Business Components een betere en productievere keuze en gemakkelijker te begrijpen voor ontwikkelaars met een relationele achtergrond. Daarnaast is de integratie van Business Components en de andere componenten momenteel beter dan met Toplink.

> ***ADF Business Components is gemakkelijk te begrijpen voor ontwikkelaars met een relationele achtergrond***

Opleidingstraject

Vanuit het ontwikkelteam van CertiQ/Tennet gingen drie ontwikkelaars meedraaien in het bouwteam van het ADC. Het doel was om deze drie ontwikkelaars klaar te stomen om het onderhoud van de applicatie te doen nadat deze in productie is genomen. Het bouwteam deed dit ook voor de 'oude' externe applicatie en door dit opleidingstraject zou deze situatie gehandhaafd kunnen blijven.
De ontwikkelaars hadden vooral ervaring met Designer en Developer en hadden alleen een basiscursus Java gedaan. Tijdens de bouw van het systeem werd een JDeveloper cursus gevolgd bij de Oracle University. Tevens is als start van de Bouw een tweedaagse seminar gegeven door Oracle Consulting om de principes van UIX beter onder de knie te krijgen.
Natuurlijk werden ook diverse tutorials geprobeerd als vingeroefening. De cursus bij de Oracle University en de seminar waren ook in tutorial-vorm. Toen uiteindelijk echt gebouwd moest gaan worden bleek dat er dan ook nog veel geleerd moest worden, zeker om zelfstandig oplossingen te kiezen.



Afbeelding 1. Schermontwerp door usability expert (Nummering zie Proof of Concept)

Afbeelding 2. Het uiteindelijke resultaat

Usability

Voordat de realisatie van de user interface zou beginnen, is in een workshop met de eindgebruikers de basis voor de user interface bepaald. Onder leiding van een usability expert van Capgemini zijn keuzes gemaakt voor zoekschermen, downloadfunctionaliteit en logische schermopbouw.
De usability expert heeft op basis van de workshopresultaten voorbeeldschermen uitgewerkt. Deze zijn vervolgens met de gebruiker besproken en aangepast. De aangepaste schermen dienden als input voor het Proof of Concept en de te ontwikkelen templates. In afbeelding 1 is een voorbeeld te zien van de schermontwerpen in afbeelding 2 het uiteindelijke resultaat. De verschillen tussen het ontwerp en de lay-out zijn ontstaan door uitlijningsproblemen en door de moeilijkheid van het gebruik van dynamische elementen in de user interface (deze werden gebruikt voor het menu).

ADF en UIX

JDeveloper is geheel voorbereid om te werken met ADF. Het is eenvoudig de scope van een applicatie te zetten, zodat alleen de gekozen onderdelen van ADF worden geboden. ADF heeft voor het View- en Model-gedeelte van een applicatie diverse alternatieven, voor de controller is Struts de default optie.
Een groot deel van de ontwikkeling binnen JDeveloper gebeurt visueel, waardoor het proces sneller verloopt. Het is echter nog altijd mogelijk om naar de code te gaan en daar rechtstreeks code aan te passen. In een enkel geval is dit de enige manier om zaken aan te passen. Het visuele aspect zorgt voor snel inzicht. Het kloppend hart van de controller (Struts) is de struts-config.xml. In afbeelding 3 is de struts-config.xml te zien van de in afbeelding 4 afgebeelde diagram die JDeveloper hiervan maakt.

```
<action path="/OverviewActions"
        className="oracle.adf.controller.struts.actions.
DataActionMapping"
        type="org.tennet.view.actions.OverviewActionsDataAction"
        name="DataForm" parameter="/WEB-INF/uix/overviewActions.uix"
        unknown="false">
  <set-property property="modelReference"
                  value="WEB_INF_uix_overviewActionsUIModel"/>
  <forward name="Search"
           path="/ShowActionsAction.do"/>
</action>
<action path="/ShowActionsAction"
        className="oracle.adf.controller.struts.actions.
```

```
DataActionMapping"
        type="org.tennet.view.actions.ShowActionsAction"
name="DataForm"
        unknown="false">
  <set-property property="modelReference" value="showActionsUIModel"/>
  <forward name="success"
           path="/ShowActions.do"/>
 </action>
<action path="/ShowActions"
        type="oracle.adf.controller.struts.actions.DataForwardAction"
        className="oracle.adf.controller.struts.actions.
DataActionMapping"
        parameter="/WEB-INF/uix/showActions.uix" name="DataForm"
        unknown="false">
  <set-property property="modelReference"
                value="WEB_INF_uix_showActionsUIModel"/>
 </action>
```

Afbeelding 3. Deel van struts-config.xml

Afbeelding 4. Grafische weergave van struts-config.xml

UIX is één van de alternatieven voor het View-deel. Het definieren van een UIX-scherm gaat ook grafisch. De code van een scherm is echter een XML. UIX kent twee extensies: UIX voor een gewone pagina, en UIT voor een template. Er kunnen verschillende templates voor een pagina worden gebruikt en het toepassen van een template gaat simpel met behulp van wizards.

Ontwikkelen van templates

Oracle ADF gaat standaard uit van de BLAF, browser look-and-feel. Het is echter aan te raden om de eigen gemaakte templates te baseren op de minimal look-and-feel. Deze template wordt standaard meegeleverd en is bovendien eenvoudiger te bewerken. Een belangrijke bron van informatie voor zowel de templates en de stylesheets is JDeveloper zelf. De help file bevat al enorm veel informatie, buiten deze informatie is er nog veel informatie beschikbaar via de website van Oracle en die van Jonas Jacobi (http://www.orablogs.com/jjacobi/)

Voor het bouwen van de externe applicatie zijn uiteindelijk twee templates ontwikkeld:

- pageLayout - hierin staan de logo's, het menu, de message box voor fouten en andere teksten, en het content gedeelte voor de rest van de pagina.
- blockLayout - deze zorgt voor de rand en het kleine logo om de stukken content die bij elkaar horen. Op het getoonde scherm (afbeelding 2): 'Basket', 'Search certificates' en 'Results'

Het menu bleek het lastigste om aan te passen in de standaard templates van Oracle. Oorspronkelijk was een dynamisch drop-down menu bedacht, maar dit bleek erg moeilijk te realiseren. Er is gekozen om het dropdown gedeelte te tonen op een regel onder het hoofdmenu. Om dit te realiseren dienden echter wel lege pagina's (dummies) te worden gemaakt om de balk zichtbaar te maken.

Proof of Concept

De basis voor de templates is gelegd tijdens de proof of concept. Binnen het ADC wordt volgens de systeemontwikkelmethodiek RUP gewerkt. Eén van de belangrijkste principes van RUP dat risicovolle onderdelen zo vroeg mogelijk in een project moeten zijn opgelost. De proof of concept (POC) die voor TenneT/CertiQ is gemaakt diende dit doel.
De gebruikers van het externe systeem dienen in te loggen via een Radius Server. Binnen het ADC was hier geen ervaring mee. Daarom werd besloten de Radius-authenticatie in de POC mee te nemen. In het belangrijkste scherm van het systeem worden certificaten gekoppeld aan een basket.
Een basket is een voorgenomen actie met een hoeveelheid certificaten. Dit scherm was onderdeel van de usability-workshop en werd ook als belangrijkste onderdeel van de POC beschouwd.
Eén van de lastigste onderdelen van deze pagina was de onderliggende verzameling data. Standaard UIX gaat er vanuit dat een scherm gebaseerd is op entiteiten en als resultaat een lijst met items. Zeker als op dat scherm ook nog een update moet worden gedaan. In het scherm uit de POC moesten drie schermen worden gecombineerd (voor nummers zie afbeelding 1):

1. Een standaard create/update scherm
2. Een zoekscherm met als resultaat een totaal aantal MWh.
3. Een scherm waar meerdere 'stored procedures' onder hangen die het gevonden aantal MWh gebruiken.

Het zoekscherm was het meest problematische onderdeel. List of values (LOV) kunnen worden gedefinieerd als pop-up of als drop-down list, en geven het resultaat terug in een veld uit de query waarop de pagina is gebaseerd. Het zoekscherm was echter gebaseerd op de volgende query:

```
SELECT SUM(MWH_QUANTITY) as Total
FROM CERTIFICATES Certificates
WHERE (CTE_TNY_TECHNOLOGY = :1)
AND   (CTE_COUNTRY_CODE = :2)
```

Deze query gebruikt het resultaat van de LOV in de where-clause. De oplossing is gevonden door het gebruik van een transient veld, dit is een tijdelijk veld op de pagina dat niet in de database wordt opgeslagen. Verder werd de lijst met country's toegevoegd aan de bindings van de pagina. Bindings zijn binnen ADF de interface tussen het View/Controller deel van

Adv. Array Informatiekwaliteit

de applicatie en het model. Het volgende stukje UIX code toont de code zoals het voor het zoekscherm werkte:

```
<messageChoice model="${bindings.baseElementId}" name="country">
    <contents childData="${bindings.CountryList.rangeSet}">
        <option value="${uix.current.CountryCode}" text="${uix.current.
CountryName}"/>
    </contents>
</messageChoice>
```

MessageChoice is de tag die de dropdown list definieert. Model geeft aan in welk veld het resultaat terecht moest komen: in dit geval dus het transient baseElementId. In name werd de naam van het veld gedefinieerd waarmee het veld kan worden aangeroepen op de pagina. De childData geeft aan waar de onderdelen van de lijst moeten worden opgehaald. Binnen option geeft value aan welke waarde moeten worden opgeslagen, terwijl text aangeeft wat voor de gebruiker als waarde zichtbaar is. De tweede parameter in bovenstaande query werd nu gevuld met de waarde uit baseElementId.

Partial page rendering

Het derde blok (zie in afbeelding 2 het blok met titel 'Result') moest alleen worden getoond als er een resultaat was gevonden, dus na het uitvoeren van de query. Hiervoor kent UIX standaard functionaliteit: 'partial page rendering', het conditioneel opbouwen van de pagina op basis van inhoud. Hiermee kan na het drukken op een knop, ofwel het afvuren van een event, een element op de pagina worden ververst. Het is echter ook mogelijk om delen van de pagina aan de hand van een reguliere expressie te laten tonen of niet. Deze laatste optie is bij deze pagina gebruikt met behulp van van de 'rendered' property.

```
<tableLayout width="100%" rendered="${requestScope.FoundTotalMWh !=
null}">
...
</tableLayout>
```

Als een UIX pagina wordt getoond of ververst wordt deze vanuit de XML-definitie opgebouwd (rendered). Alles wat de property 'rendered' op true heeft staan wordt getoond. In bovenstaand stuk UIX wordt alles getoond, dat zich binnen deze tableLayout tags bevindt, wanneer de property 'FoundTotalMWh' gevonden wordt in de request (de aanvraag van de pagina).

Default waarden vanuit het menu

Een aantal menu-opties zijn gericht op het aanmaken van een basket voor een speciale actie. Als de gebruiker bijvoorbeeld de optie 'Transfer' aanklikt, moet een basket van dat type worden aangemaakt. In het menu was dit als volgt gecodeerd:

```
<contents>
  <link text="Transfer" destination="CreateBasket.do?bet_
type=Transfer"/>
              </contents>
```

Door de link op deze manier te definiëren wordt de parameter bet_type op het request gezet. Het request is de aanvraag die wordt gedaan als de link in het menu wordt aangeklikt. Binnen Struts is een datapage, een pagina en een actie in één aangemaakt die in Struts gedefinieerd was via de CreateBasket action. De pagina is in dit geval de create basket pagina waarop verschillende waarden worden ingevuld alvorens de basket wordt vastgelegd. De actie die bij deze pagina hoort is een create actie, de create van de basket. Elke keer als deze pagina wordt aangeroepen wordt automatisch een basket aangemaakt.

Er zijn verschillende momenten in de aanmaak (rendering) van een pagina waarop code kan worden toegevoegd. In analogie met de database zijn dat triggers. De meest gebruikte is de prepare-model code, die afgaat voordat het model wordt klaargezet voor de pagina. Op dit punt wordt de volgende code opgenomen voor de create-basket pagina:

```
protected void prepareModel(DataActionContext actionContext)
    {
     super.prepareModel(actionContext);
     // Fill default values based upon request en context.
     HttpServletRequest request = actionContext.getHttpServlet-
Request();
     //Get the current row of the datacontrol
     DCBindingContainer binding = actionContext.getBindingContainer();
     DCIteratorBinding dNameBinding =
                              binding.findIteratorBinding("BasketsIterat
or");
     Row row = dNameBinding.getCurrentRow();
     //Get Reguest and the Bet_type parameter
     //Fill the default values;
     row.setAttribute("BetType",request.getParameter("bet_type"));
    }
```

Afbeelding 5. Parameter bet_type wordt gebruikt voor een default value

Versiebeheer

In het ADC wordt standaard gebruik gemaakt van Clearcase als versiebeheer-tool. De combinatie Clearcase /JDeveloper heeft tot verschillende problemen geleid. Doordat niet automatisch een file werd uitgecheckt, bleef deze write protected. Deze protection zorgde ervoor dat bepaalde wijzigingen niet werden weggeschreven, JDeveloper maakte hier geen melding van.

Herhaaldelijk heeft dit uren gekost om één en ander weer te herstellen.
Uiteindelijk is gekozen om JDeveloper elke file die wordt veranderd, automatisch uit te laten checken (gereserveerd voor de user) en een lijst van standaard files default uit te checken maar niet gereserveerd voor de user. Zeker voor een beginnende ontwikkelaar is onduidelijk, welke files precies wijzigen.
Overigens is het openen van bepaalde files al genoeg om deze te laten veranderen, terwijl de inhoud niet werkelijk verandert. Hier maakt JDeveloper een slordige indruk.

Conclusie

JDeveloper is een volwassen tool om Java-applicaties mee te ontwikkelen. De verschillende visuele editors maakt bepaalde handelingen gemakkelijker en sneller. UIX biedt veel mogelijkheden om een goede en gebruikersvriendelijke lay-out te ontwikkelen. Het heeft echter wel tijd nodig om UIX onder de knie te krijgen. Bepaalde aspecten zijn voor verbetering vatbaar, bijvoorbeeld de LOV afhandeling met lookup-usages. Voor een ingewerkt en ervaren team is zeker een hogere productiviteit te behalen met UIX. Een aandachtspunt voor JDeveloper is de combinatie met een tool voor versiebeheer. Uiteindelijk is de applicatie op tijd opgeleverd aan TenneT/CertiQ. Het bouwteam voelt zich ingewerkt genoeg om onderhoud te doen. Het team heeft tijdens de acceptatietesten overigens al onderhoud gepleegd. Bij het ter perse gaan van dit artikel is de applicatie in productie gegaan.

De meest gebruikte trigger is de prepare model code, die afgaat voordat het model wordt klaargezet voor de pagina

Ruben Spekle (ruben.spekle@capgemini.com),
Jettro Coenradie (jettro.coenradie@capgemini.com) en
Léon Smiers (leon.smiers@capgemini.com)