Rijk en aantrekkelijk

Analyse EK voetbal met adf 11g data visualization tools

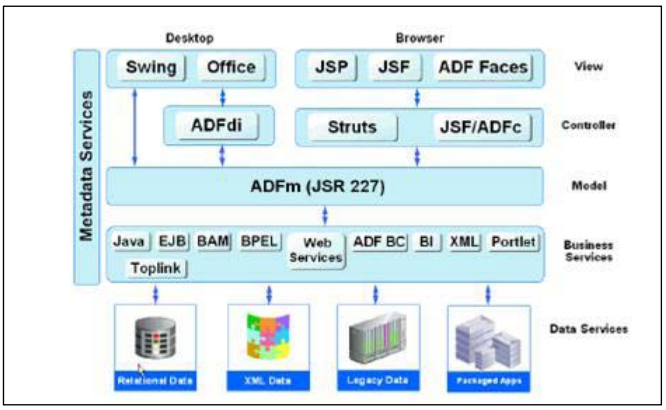
In dit artikel stellen de auteurs de Oracle ADF 11g Data Visualization Tools voor. Aan de hand van voetbalstatistieken - hoe kan het anders in deze periode - proberen de auteurs met de nieuwe DVT van Oracle ADF inzicht te krijgen in de kansen van 'onze jongens' op EURO2008.

Oracle werkt momenteel hard aan het uitbrengen van de productierelease van JDeveloper 11g met daarin de ADF Faces 11g Rich Client Components. Het bedrijf besteedt veel aandacht aan de eisen rondom Web 2.0 interactieve webapplicaties. Volgens Oracle wordt een grote sprong voorwaarts gemaakt in de snelheid waarmee ontwikkelaars rijke en interactieve applicaties kunnen maken die goed performen en goed onderhoudbaar zijn. Oracle ADF 11g Faces Rich Client is een set standaard JSF-componenten met ingebouwde Ajax-functionaliteit. Door gebruik te maken van Ajax kunnen desktop-achtige applicaties op standaard internettechnologie draaien, terwijl JSF zorgt voor server-side control.

Op 3 juni was Oracle's senior productmanager Katrina Obradovic, verantwoordelijk voor de data visualization-tools, op bezoek bij AMIS in Nieuwegein. Voor een publiek van nogal ervaren ADF-ontwikkelaars van allerlei organisaties in Nederland vertelde zij over de DVT-componenten. Ook sprak zij over het ontwikkelproces achter de Oracle-schermen dat door diverse vooral interne klanten wordt beïnvloed en de nabije toekomst van JDeveloper 11g en DVT in het bijzonder. Uiteraard liet ze enkele spectaculaire demo's zien, zoals die recent op JavaOne 2008 zijn gepresenteerd. Na het diner ondersteunde zij de bezoekers bij het doen van een hands-on workshop met DVT. In dit artikel beschrijven drie ADF-specialisten - Eric van Mourik (Truston), Frank Houweling en Luc Bors (beiden werkzaam bij AMIS) hun bevindingen naar aanleiding van deze sessie.

Introductie van Data Visualization-componenten

Een van de onderdelen van 11g ADF Faces Rich Client is de set Data Visualization Components, ook wel Data Visualization Tools (DVT) genoemd. DVT kan worden gezien als de opvolger van Oracle BI-beans. DVT's zijn rijke en interactieve ADF Faces-componenten die de gebruiker veel mogelijkheden biedt om data te analyseren met behulp van grafieken en tabellen. Oracle heeft met DVT een centrale charting-component die uiteindelijk in alle Oracle-producten met grafieken of andere datavisualisatie gebruikt gaat worden - zoals Enterprise Manager, Hyperion, BAM (Business Activity Monitoring) en met name Oracle Fusion Applications. Hierdoor ontstaat een consistente manier van werken voor ontwikkelaars, maar minstens zo belangrijk is dat eindgebruikers een consistente look and feel krijgen. Gebruik van DVT's kent dezelfde voorwaarden als de rest van ADF: een run-time licentie van ADF is vereist, of een licentie voor Oracle Application Server. Voor de ontwikkeling met ADF - inclusief DVT - is geen licentie nodig; gebruik van JDeveloper is gratis.
De verwachte releasedatum voor 11g is 2008. Wij verwachten dat dit ergens heel diep in de tweede helft van 2008 wordt. Tot die tijd moeten we het doen met de technical previews, waarvan momenteel de vierde versie beschikbaar is. In die versie zijn de DVT's verder geperfectioneerd, en we kunnen stellen dat Oracle met de DVT's een mooie feature heeft toegevoegd aan het ADF. Het voert te ver om alle vijftig componenten te behandelen, maar een aantal zullen wij beschrijven.



De ADF-architectuur - van datasource naar use-rinterface. De DVT-componenten zijn onderdeel van ADF Faces

DVT binnen ADF-Architectuur

Door de architectuur van ADF is het mogelijk om grafieken te koppelen aan alle datasets die via een ADF data-control (ADFm JSR227) beschikbaar kunnen worden gemaakt.

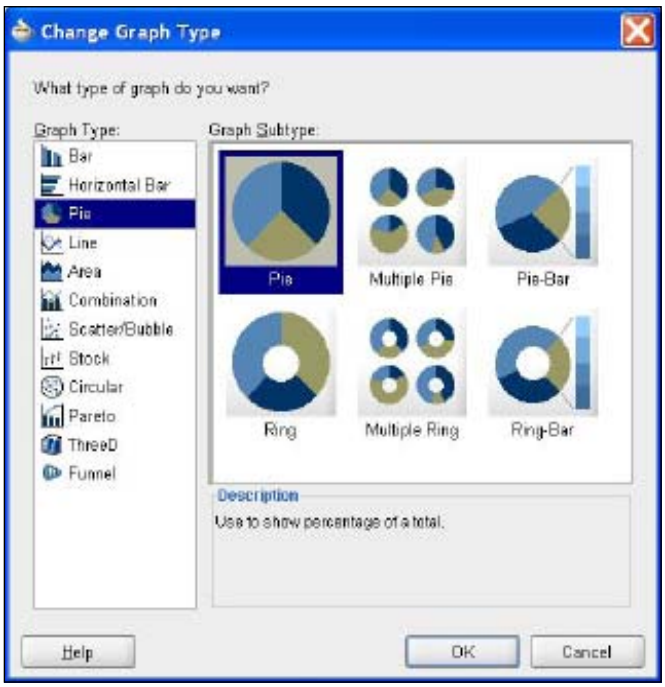
Binding aan een standaard rowset en binding aan hiërarchische data-controls wordt ondersteund. Concreet houdt dit in dat grafieken gemaakt kunnen worden met data afkomstig uit een relationele database via ADF BC, en vanuit bijvoorbeeld een BAM-server of BI-server. Ook kunnen webservices gebruikt worden om grafieken van data te voorzien. Al met al maak je een grafiek met simpele drag & drop van een collection van het data-control-palet, net zo als je een data-bound form of tabel-component op een pagina plaatst. Daarnaast kunnen de data voor een DVT-component door een managed bean worden aangeleverd.
De DVT-componenten doen mee met skinning, en ondersteunen Partial Page Refresh. De DVT-componenten kunnen zelf een PPR-event initiëren, maar kunnen ook reageren op een andere component. Ten slotte is het mogelijk om nagenoeg alle eigenschappen van attributen van een DVT door middel van EL-expressies dynamisch te beïnvloeden.
In de huidige ADF 10.1.3 zijn er overigens ook al mogelijkheden om eenvoudige grafieken te maken, maar dat vereist heel wat meer werk dan met de 11g DVT's. De 10.1.3 ADF JSP Graph-tag is nog geen JSF UIComponent dus moet je gebruik maken van een <f:verbatim> tag. Deze moet dan weer in een <af:panelGroup> om PPR mogelijk te maken. Het maken van een interactieve grafiek is met de nieuwe DVT's veel eenvoudiger geworden.

Overzicht componenten

De ADF Data Visualization Components bestaan uit de volgende hoofdgroepen: Graph, Gauge, Pivot Table, Geographic Map en Gantt. Hier zal nog de Hierarchy Viewer bij komen voor de visuele presentatie van bijvoorbeeld organogrammen of bill-of-materials, maar die is nog niet opgenomen in JDeveloper 11g TP4. Voor alle groepen geldt dat het ontwikkelen met de componenten volledig declaratief kan plaatsvinden; gebruikmakend van het data-control-palet, componentpalet, de visual editor, property inspector, zoals we dat ook zien bij ander JSF-componenten in JDeveloper 11g. Voor alle data visualization-componenten is er bovendien een previewer aanwezig, waarmee je op basis van live data kunt zien wat het resultaat van je werk is, zonder de pagina eerst te moeten compileren en runnen.

Graph

Graphs en gauges zijn beide bedoeld voor het grafisch presenteren van data. Het verschil tussen graphs en gauges zit vooral in het aantal 'data points' dat ze weergeven. In een graph worden veelal meerdere voorkomens van dezelfde gegevenssoort tegen elkaar afgezet. Denk bijvoorbeeld aan een bar-graph, waarin het voorraadniveau van een aantal magazijnen tegen elkaar wordt afgezet. Ieder magazijn wordt daarbij gerepresenteerd door staaf. Een extra gegevensrij (magazijn) vertaalt zich in een extra staaf binnen de bar-graph. Alle graphs kunnen als Flash of al SVG worden gerenderd en ondersteunen vormen van gebruikersinteractie. Denk daarbij aan zaken als in-/uitzoomen, scrolling, het aanpassen van het tijdvlak waarover de data wordt getoond door een bepaald deel van de tijdsas te selecteren, animaties, enzovoort.
In totaal zijn er meer dan vijftig verschillende soorten graphs beschikbaar. De meeste daarvan zitten verscholen in de Advanced Graph-component. Deze component presenteert zich standaard als bar-graph, maar heeft een graph-type-attribuut waarmee het mogelijk is de bar-graph te veranderen in een ander type. Sommige types stellen overigens wel specifieke eisen aan de structuur van de onderliggende data.



De lijst van graphs met als voorbeeld enkele verschijningsvormen van de pie chart

Naast de Advanced Graph-component zijn er zeventien zogenaamde simplified graphs. Dit zijn veelgebruikte graphs, waarvan het type niet te wijzigen is. Een simplified graph heeft echter wel gewoon allerlei attributen en child-tags waarmee de graph kan worden gecustomized. Alle graphs, advanced of simplified, maken gebruik van dezelfde engine.

Het voert te ver alle beschikbare componenten hier te bespreken, maar beschikbaar zijn onder andere:
• Bar Graph, in diverse varianten. Bedoeld om data te visualise-

ADV

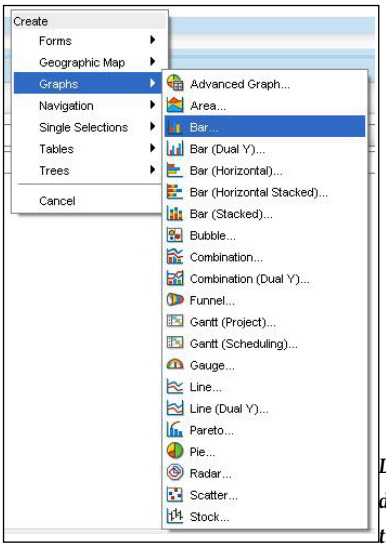
ren door middel van een reeks verticale of horizontale staven.

- Pie Graph, in diverse varianten. Maakt het aandeel van een bepaalde gegevensgroep (productie van een vestiging) in het totaal (totale productie) inzichtelijk.
- Line Graph, in diverse varianten. Presenteert data als lijn, als een reeks punten of als punten die onderling met een lijn verbonden zijn. Denk aan een overzicht van de gemiddelde dag- en nachttemperatuur per maand.
- Area Graph, in diverse varianten. Bedoeld om trends over een periode te visualiseren.
- Combination Graph. Graph waarin bars, lines en areas kunnen worden gecombineerd.
- Scatter/Bubble Graph. Visualiseert data door middel van een cirkel van een bepaalde grootte en met een bepaalde positie binnen de graph. Een dergelijke graph kan bijvoorbeeld worden gebruikt om de correlatie tussen salaris (x-as), jaren ervaring (y-as) en productiviteit (omvang van de cirkel) inzichtelijk te maken.
- Stock Graph. Toont de hoogste, laagste en laatste waarde van (bijvoorbeeld) de koers van een aandeel.
- Radar Graph. Bedoeld om cyclische patronen te tonen. Bijvoorbeeld de maandelijkse verkoopresultaten over de afgelopen drie jaar.
- Pareto Graph. Toont data in bars en bevat daarboven een lijn die het cumulatieve percentage van de bars toont.
- ThreeD Graph, in diverse varianten. Om data te visualiseren in een graph met drie assen/ dimensies.
- Funnel Graph. Hiermee kan de voortgang van stappen in een proces worden gevisualiseerd. De graph bestaat uit een aantal cilinders op een horizontale as, waarbij iedere cilinder een processtap vertegenwoordigt. Per stap worden de actuele gegevens (cijfers) afgezet tegen doelstellingen door de cilinder vol te laten lopen.

Binnen de graphs componenten is een enorm arsenaal aan JSF-tags beschikbaar waarmee verschijningsvormen kunnen worden beïnvloed. Veel tags zijn generiek toepasbaar op alle types graphs (common child tags), andere alleen voor een of enkele type(s) (specific child tags). De tags kunnen onder andere gebruikt worden om animatie-effecten te regelen; data-points te tonen met een zelf te bepalen icon; de verschijningsvorm, fonts, kleuren, titels en legenda van de graph aan te passen, maar ook voor zaken als het formatteren van numerieke waarden, of het selecteren van een range op een tijdsas.

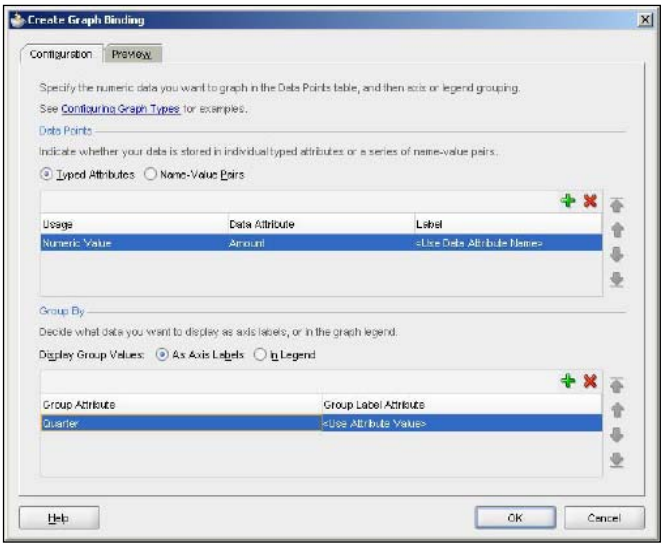
Staafgrafiek

Met een simpele drag en drop-actie kun je bijvoorbeeld een taart- of staafgrafiek maken. Een collectie kun je vanuit de data-control als Bar (staafgrafiek) droppen op een pagina.



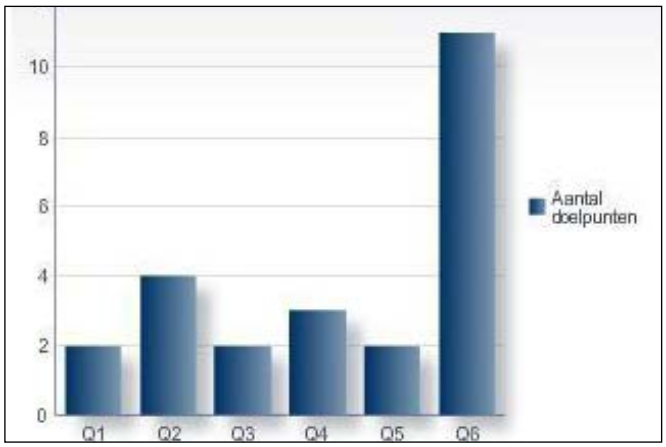
Lijst van graphs getoond bij een drag & drop vanaf het data-control-palet

In de wizard kun je de bindings aangeven voor de waarden en de groepen. Je zou achter het tabblad preview al kunnen bekijken hoe het eindresultaat er uit ziet, maar de pagina kan ook direct gerund worden.



Graph Data Binding Editor om te configureren hoe de graph wordt gevoed vanuit de data-control

Om een voorbeeld te gebruiken laten wij aan de hand van een eenvoudige staafgrafiek zien op welk moment het Nederlands elftal doelpunten scoort in een wedstrijd. De tv-kijkers onder ons kunnen blijkbaar beter niet in het laatste kwartier van hun plek gaan, want dan is de kans dat ze een doelpunt van Oranje missen het grootst. Wij slaan op Europese Kampioenschappen vooral in het laatste kwartier toe. Wat was er ook al weer met 'die Mannschaft'?



Staafgrafiek met het aantal doelpunten per wedstrijd-kwartier

Data uit een Managed Bean

Om vanuit een managed bean de data voor een grafiek aan te bieden - dus niet via ADF Data Binding - moet gebruik worden gemaakt van het tabularData attribuut van de Advanced Graph-component. De Simplified Graph-component heeft dit attribuut niet. Hoe gaat dit in zijn werk? Maak eerst een methode aan in een bean die voor de data zorgt. Om bij ons voorbeeld te blijven, deze methode bevat de gegevens van voor- en tegendoelpunten van het Nederlands elftal op de laatste vijf Europese Kampioenschappen.

```
public List getVoorEnTegen()
{
    ArrayList list = new ArrayList();
    String[] rowLabels  = new String[] {"VOOR", "TEGEN"};
    String[] colLabels  = new String[] {"1988", "1992",
"1996","2000","2004"};
    Double [] [] values = new Double[][]{
        {4, 6, 8, 10,4,8},  {2, 2, 2,3,5,3}        };
    for (int c = 0; c < colLabels.length; c++)
    {
     for (int r = 0; r < rowLabels.length; r++)
       {
        list.add (new Object [] {colLabels[c], rowLabels[r],
            new Double (values[r][c])});
       }
    }
    return list;
}
```

Managed Bean-methode die de data voor een Bar Chart (met twee dataseries) levert

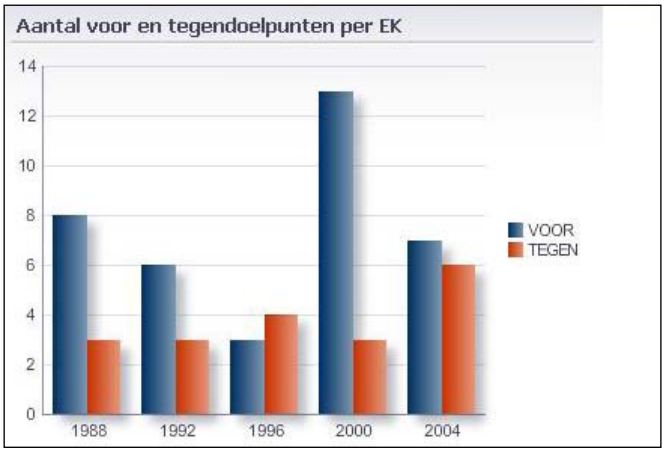
Configureer de class met deze methode als managed bean. Drop nu de Advanced Graph-tag vanuit het componentpalet op een JSF-pagina. In de data-page van de Property Inspector kies je in de Tabular Data dropdown-box voor Expression Builder en zoek je de juiste managed bean, met de methode die de lijst met data bevat. In de Expression Builder, wijst het tabularData-attribuut nu naar de methode uit de managed bean. In ons voorbeeld heet de managed bean *voorgaandeJaren* en de methode *getVoorEnTegen*. Het tabularData-attribuut heeft dus nu de setting: #(voorgaandeJaren. voorEnTegen).

```
<dvt:graph tabularData="#{voorgaandeJaren.voorEnTegen}">
     <dvt:graphTitle text="Aantal voor en tegendoelpunten per EK"/>

</dvt:graph>
```

Configuratie van grafiek zonder ADF Data Binding maar met data uit managed bean

Als we de pagina nu runnen dan wordt de grafiek getoond. Deze toont een redelijk zorgwekkende oplopende trend in het aantal tegendoelpunten van Oranje op eindtoernooien!



Bar Chart op basis van managed bean data - met twee dataseries: Voor en Tegen

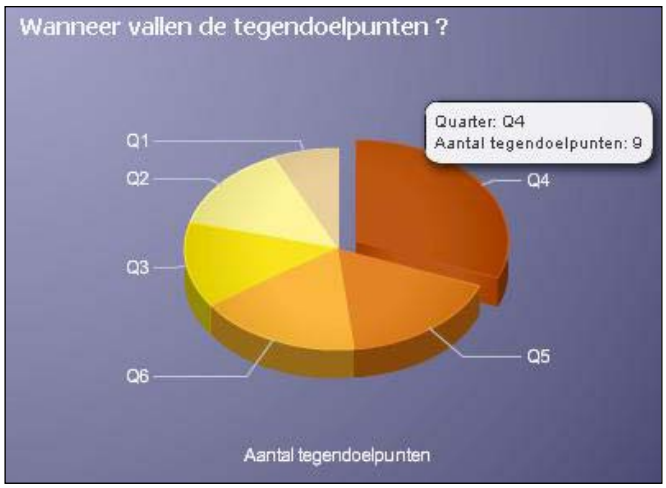
Maar zoals een bekende voetballer ooit zei: "Zolang je er zelf meer scoort dan de ander."

Taartgrafiek

In een taartgrafiek met animatie en 3D-effecten kunnen we zien hoe deze doelpunten zijn verdeeld over de verschillende periodes in een wedstrijd. We willen dat de taartpunt met het grootste aantal tegendoelpunten als het ware explodeert. Om dit te bereiken voegen we een dvt:seriesSet toe binnen de dvt:pieGraph-tag. Binnen de dvt:seriesSet is nog een dvt:series nodig, waarvan we de pieSliceExplode-property op 100 zetten. In de Property Inspector moet nog een aantal properties van de taartgrafiek gezet worden: het threeDEffect =true, imageFormat=FLASH en stylePath =Comet.

```
<dvt:pieGraph id="pieGraph3"
      value="#{bindings.goalsConcievedPerPeriod1.graphModel}"
      threeDEffect="true" imageFormat="FLASH" style="Comet">
      <dvt:graphTitle text="Wanneer vallen de tegendoelpunten ?"/>
      <dvt:seriesSet>
          <dvt:series id="10" pieSliceExplode="100"/>
      </dvt:seriesSet>
      <dvt:sliceLabel textType="LD_TEXT">
          <dvt:numberFormat decimalDigit="0"/>
      </dvt:sliceLabel>
      <dvt:legendArea rendered="false"/>
</dvt:pieGraph>
```

Configuratie in JSPX van 3D pie-chart - data komt uit bound rowset in de data-control



Taartdiagram met uitgelicht het kwartier met de meeste tegendoelpunten

Blijkbaar is het eerste kwartier na rust (kwartier 4) het gevaarlijkst voor het Nederlands elftal. Dan vallen de meeste tegendoelpunten.

Gauges - meters en wijzertjes

Gauges zijn als het ware grafische metertjes die kunnen worden gebruikt om snel visueel weer te geven wanneer een bepaalde waarde in een kritiek gebied valt. Gauges concentreren zich, in tegenstelling tot graphs volledig op één 'data point'. Een gauge toont bijvoorbeeld expliciet de voorraad van een bepaald magazijn. Wanneer een gauge-component gebaseerd wordt op een datacollectie die meerdere rijen bevat, zal er voor iedere rij een gauge verschijnen. Bij gauges kunnen minimum-, maximum- en drempelwaarden worden ingesteld die gevisualiseerd worden, bijvoorbeeld door onderscheidend kleurgebruik. Gauges vormen daardoor een uitstekend middel om te monitoren of kritische waarden worden bereikt. Ook gauges kunnen als Flash of SVG worden gerenderd. De volgende soorten gauges zijn beschikbaar: Dial, Status Meter en LED. Voor dials en statusmeters kunnen een oneindig aantal drempelwaarden worden ingesteld. Voor sommige LED's is dit beperkt tot twee.

Voorbeelden van verschillende Gauge-uitvoeringen

Ook bij gauges is sprake van zo'n twintig child-tags waarmee verschijningsvorm en gedrag van de gauge kon worden gestuurd. Het gaat hier onder meer om zaken als fonts, kleur- en schaduwgebruik, het soort wijzer bij een dial, te gebruiken eenheden, minimum-, maximum- en drempelwaarden en special effects. Met behulp van een tresholdSet kan worden aangegeven wanneer een waarde in het groene, gele of rode gebied valt.

```
<dvt:thresholdSet>
     <dvt:threshold thresholdMaxValue="1" id="t0"/>
     <dvt:threshold thresholdMaxValue="3" id="t1"/>
     <dvt:threshold thresholdMaxValue=""  id="t2"/>
</dvt:thresholdSet>
```

Definitie van Gauge-drempelwaarden

Zo kunnen we in ons voorbeeld snel zien of we te maken hebben met spelers die veel scoren, of juist minder.

ADV

Land	Speler	Positie	Goals
Duitsland	Klose	Voorhoede	4 (0 1 3 10)
Frankrijk	Henry	Voorhoede	4 (0 1 3 10)
Italie	Inzaghi	Voorhoede	6 (0 1 3 10)
Italie	Toni	Voorhoede	3 (0 1 3 10)
Nederland	Kuijt	Voorhoede	5 (0 1 3 10)
Nederland	Robben	Voorhoede	2 (0 1 3 10)
Nederland	van Nistelrooij	Voorhoede	3 (0 1 3 10)

Gauge-indicatoren voor spitsen - hoe meer doelpunten, hoe verder de uitslag van de gauge

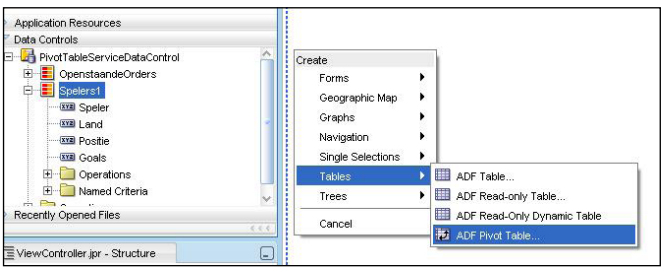
Pas vooral op voor de spits van Italië, Inzaghi, maar een beetje voetbalkenner heeft uiteraard voor deze waarschuwing de gauge niet nodig.

Pivot Table

De Pivot Table is een zeer krachtige component waarmee genormaliseerde relationele data kunnen worden weergegeven in een gebruikersvriendelijke matrixstructuur. Een klassiek voorbeeld hiervan is de urenstaat. In een relationeel datamodel is die veelal geïmplementeerd door middel van een tabel die per combinatie taak/medewerker/datum een rij bevat met het aantal bestede uren voor de betreffende combinatie. Heel efficiënt, maar niet voor de gebruiker die zijn/haar uren moet invoeren. Die wil een matrix, met de verschillende taken als rijen en de dagen als kolommen. Wat alle huidige oplossingen gemeen hebben, is dat er veel en ingewikkelde code aan te pas komt. En 'net zo handig en flexibel als in Excel' werd het tot onze grote frustratie bijna nooit.

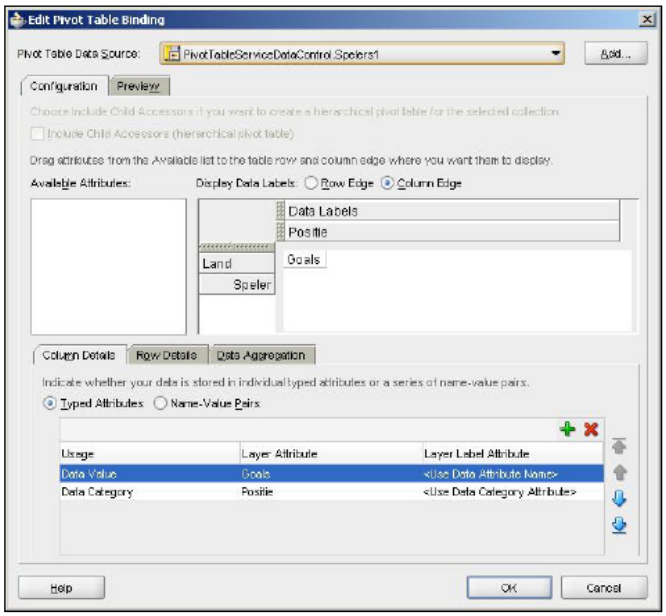
De Pivot Table-component maakt ons het leven op dit vlak opeens een stuk makkelijker. De pivot table vormt een grid waarbij zowel op de x-as als de y-as op meerdere niveaus 'data labels' kunnen worden opgenomen. Daarbij biedt de pivot table de functionaliteit om op de verschillende niveaus binnen de structuur automatisch (sub)totalen te genereren. Het is voor de eindgebruiker mogelijk om data labels van de ene naar de andere as te slepen, zodat gegevens in een andere structuur worden geordend en vanuit een andere invalshoek kunnen worden bekeken. De pivot table is sterk vergelijkbaar met de 'draaitabel' uit Excel en de 'crosstab' in Oracle Discoverer. Het is met de Gantt de enige component die niet alleen data read-only presenteert, maar ook voor manipulatie kan worden gebruikt.

Zoals voor vrijwel alle ADF Faces- en ADF Data Visualization-componenten geldt, is het creëren en configureren verbazingwekkend eenvoudig en volledig declaratief. Pivot tables kunnen gebaseerd worden op elke datacollectie met een rij-structuur. In de productierelease zal het ook mogelijk zijn om pivot tables te baseren op hiërarchische data-controls en zal drill-down functionaliteit aanwezig zijn.
In ons voorbeeld gebruiken we een view uit een database als datasource. Onze pivot table is bedoeld om inzicht te verschaffen in het aantal doelpunten dat gescoord is op een Europees kampioenschap. Per land. Of nee, per speler. Of toch liever per positie op het veld? Ach, waarom niet gewoon allemaal? Nadat we de view beschikbaar hebben gemaakt als data-control, bijvoorbeeld door middel van een ADF BC View Object, kunnen we die data-control met de attributen Land, Positie, Speler en Goals gebruiken als basis voor onze pivot table. Dit gaat het snelst door de data-control naar een JSF-pagina te slepen en dan in het getoonde contextmenu de optie ADF Pivot Table te kiezen.

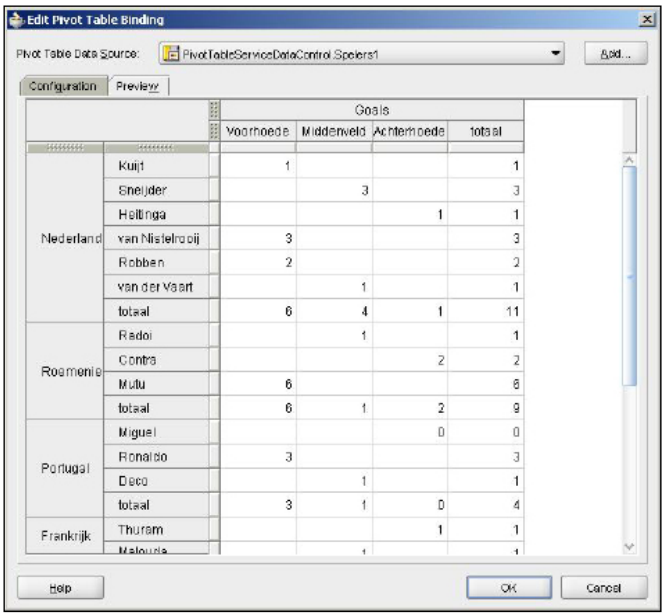


Voeg een pivot table toe aan een JSF-pagina door drag & drop van een datacollectie

Er verschijnt een Create Pivot Table Binding-editor waarmee we de data-binding kunnen regelen en de pivot table kunnen configureren. In dit voorbeeld tonen we de attributen Land en Speler initieel als rij ('van boven naar beneden'), de Positie als kolommen ('van links naar rechts') en Goals als celwaarde. We geven dit in de editor aan door de beschikbare attributen naar de gewenste sectoren van de pivot table te slepen.



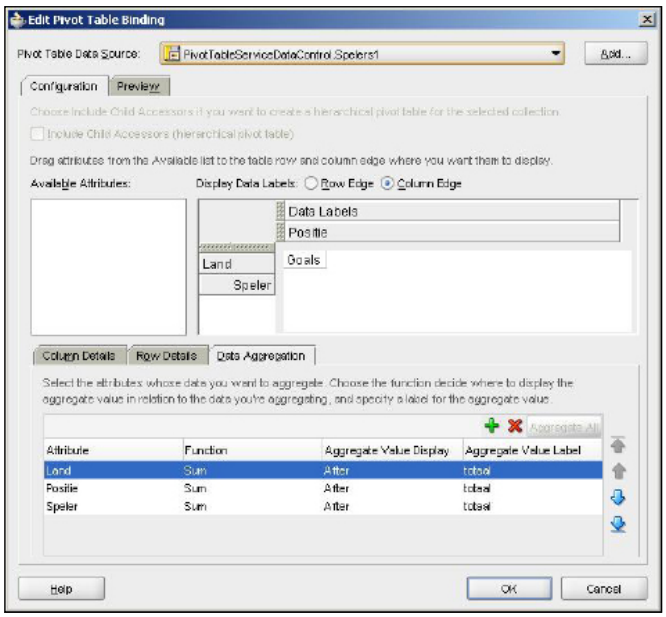
Configureer de data-binding van de pivot table



Design Time Preview van de pivot-table binnen JDeveloper

Op het tabblad Data Aggregation kunnen we aangeven hoe we om willen gaan met aggregaties als totalen, gemiddelden, enzovoort. Alsof we echte eindgebruikers zijn, geven we hier direct aan dat we 'alles' willen, dus (sub)totalen op ieder beschikbaar (sub)niveau (Land, Speler en Positie).

In drie eenvoudige stappen hebben we volledig declaratief een rijke en data-bound component aan onze pagina toegevoegd. De kracht ervan wordt duidelijk als we de pagina runnen. Door middel van drag & drop kan de rij/kolomstructuur door de gebruiker worden aangepast, zodat de gewenste presentatievorm ontstaat. Daarbij worden ook de diverse totalen automatisch opnieuw herschikt. Zo zien we bijvoorbeeld rechts dat de voorhoedespelers gezamenlijk 29 keer gescoord hebben, terwijl dit in de linkervariant niet een twee drie te zien is.



Stap 2 in configuratie pivot table - benoem rij- en kolom-labels en sub-aggregaten

Initiële aanblik Pivot Table

De previewer geeft op basis van echte data een voorschot op het eindresultaat. Dit is een erg handig hulpmiddel om vertrouwd te raken met de verschillende mogelijkheden.

Tijdens haar presentatie bij AMIS heeft Katarina Obradovic nog meer geavanceerde functionaliteiten van de Pivot Table en andere componenten laten zien. Denk aan zaken als cel-, rij- en kolomselectie, ordening, drill-down door hiërarchische gege-

vens, het muteren van celwaarden, het highlighten van cellen met specifieke waarden, enzovoort. Toch blijft er natuurlijk nog altijd iets te wensen over. Zo lijkt het in Technical Preview 4 (nog?) niet mogelijk om verschillende aggregatiefuncties naast elkaar te gebruiken. Terwijl daar best use-cases voor bestaan. Stel je voor dat we in ons voorbeeld in de body niet het attribuut Goals hadden getoond, maar een attribuut aantal kansen en een attribuut rendement, tegenwoordig niet weg te denken uit de nabeschouwingen. Dan hadden we waarschijnlijk het attribuut aantal kansen als SUM geaggregeerd willen zien, en het attribuut rendement als AVERAGE.

Geographic Map

De geographic map maakt het mogelijk gegevens te projecteren op een geografische kaart. Denk bijvoorbeeld aan een kaart waarop iedere provincie een bepaalde kleur krijgt, afhankelijk van de verkoopresultaten in de betreffende provincie. Of een kaart waarop per provincie een pie-graph wordt getoond die de verdeling van het resultaat over verschillende productgroepen aangeeft. Er kunnen meerdere van dergelijke lagen ('themes') over een kaart gelegd worden.
De kaartdata worden ontleend aan Oracle Spatial. Het is (nog) niet mogelijk om andere kaarten, zoals Google Maps, als 'ondergrond' te gebruiken.

De volgende soorten themes zijn beschikbaar:

- Color theme. Bedoeld om regio's op een kaart van elkaar te onderscheiden. Bijvoorbeeld door iedere regio een kleur te geven die correspondeert met het aantal klanten.
- Point theme. Toont bepaalde plaatsen op de kaart, bijvoorbeeld de locaties van te bezoeken klanten. Er kunnen daarbij verschillende soorten plaatjes als aanwijzer worden gebruikt, bijvoorbeeld verschillend voor grote en kleine accounts.
- Graph theme. Hiermee kunnen (statistische) gegevens met betrekking tot een bepaalde locatie worden getoond in de vorm van een pie-graph of een bar-graph.

Voor iedere map-theme is een data-control vereist die locatie-informatie bevat die gelinkt kan worden aan de base-map. Dit kunnen coördinaten zijn, maar bijvoorbeeld ook adresgegevens die vervolgens door een geocoder naar coördinaten worden vertaald. Ook geographic maps kunnen in grote mate worden gecustomized. Zaken als afmeting van de kaart, zoomstrategie, enzovoort kunnen worden ingesteld.

Gantt

De Gantt is een component die bedoeld is voor planning- en tracking-doeleinden. De component bestaat feitelijk uit twee naast elkaar geplaatste tree-tables, met daartussen een splitter. De linker-table toont een lijst met taken, de rechter (per taak) de voortgang in de vorm van een balk die de begin- en eindtijd markeert. De x-as representeert hierbij de tijd. Binnen deze structuur kan voor iedere taak de begin- en eindtijden worden weergegeven, evenals allerlei afhankelijkheden tussen taken (en subtaken). Begin- en eindtijden kunnen worden gemuteerd door middel van verslepen en er kunnen verschillende vormen van afhankelijkheid tussen taken worden vastgelegd. De beide tables delen hetzelfde data- and selectiemodel en worden (dus) automatisch gesynchroniseerd. Dat gebeurt ook bij scrollen of het open-/dichtklappen van rijen.
Er wordt onderscheid gemaakt tussen Project Gantt en Scheduling Gantt. In het geval van een Project Gantt bevat de linkertabel een lijst met taken, in het geval van een Scheduling Gantt gaat het daar om resources in de vorm van personen, machines, vergaderzalen, enzovoort. De Gantt gebruikt partial page-rendering wanneer gegevens gemuteerd/versleept worden. Dit betekent dat niet de hele pagina opnieuw wordt geladen, maar slechts de Gantt en eventueel daarmee te synchroniseren pagina-elementen. De Gantt kan worden gebaseerd op twee soorten datamodellen: een tree of een collection met een set van rijen. Er bestaat een helper-class (GanttPrinter) die printfaciliteiten voor de Gantt biedt. Deze class genereert een Formatted Object (FO) dat in Oracle's XML Publisher (maar wellicht ook met behulp van een product als Apache FOP?) gebruikt kan worden om PDF's te produceren.

Toepassen DVT-componenten

Een van de vragen die Katrina Obradovic ons voorlegde was: hoe kunnen we ADF-ontwikkelaars overtuigen van de toegevoegde waarde van de DVT-componenten, en hoe krijgen we ze zover dat ze er ook daadwerkelijk mee aan de slag gaan? Het lijkt erop dat de belangrijkste stap - naast weten dat de componenten er zijn natuurlijk - er uit bestaat uit dat we gevoel krijgen voor het inzetten van visuele datapresentatie als alternatief voor de klassieke mechanismen als master-detail met form en tabel lay-out.

In afbeelding 15 zien we een voorbeeld een afbeelding van zo'n read-only ADF-tabelcomponent met namen van werknemers

showDetailItem 1

		Goals						
		Nederland	Roemenie	Portugal	Frankrijk	Duitsland	Italie	totaal
Voorhoede	Kuijt	1						1
	van Nistelrooij	3						3
	Mutu		6					6
	Ronaldo			3				3
	Toni						3	3
	Robben	2						2
	Henry				1			1
	Inzaghi						6	6
	Klose					4		4
	totaal	6	6	3	1	4	9	29
Middenveld	Sneijder	3						3
	Radoi		1					1
	Schweinsteiger					3		3
	Malouda				1			1
	Deco			1				1

Pivot Table na drag. Land van rij naar kolom door de gebruiker

ADV

met hun bijbehorende salaris, bonus en commissie. In afbeelding 16 zien we een DVT-component, een stacked bar-graph op basis van dezelfde dataset. Welke afbeelding toont meer informatie? In de graph zijn de hoogste en de laagste waarde, onderlinge verschillen en verhoudingen in een oogopslag te overzien. Er wordt dus eigenlijk zelfs meer informatie getoond dan in de tabel: meer overzicht en inzicht. Ook de exacte getallen zijn beschikbaar in de graph - als de gebruiker met de muis over een werknemer beweegt, verschijnt een klein pop-up window dat de data toont (niet in afbeelding).

Een bekend spreekwoord luidt: 'een plaatje zegt meer dan duizend woorden'. Graphs, gauges, charts, gantts en geografische kaarten zeggen dan ook veel meer dan getallen in een tabel. De DVT-componenten zijn vooral erg waardevol in situaties waar de gebruiker (snel) inzicht in of een overzicht over data nodig heeft, zodat hij de data in een oogopslag kan analyseren. Visuele informatie is veel sneller en efficiënter op te nemen voor onze hersenen dan getallen, en blijft langer in ons geheugen. De DVT-componenten zijn als het ware een soort visuele rapportjes. ADF-ontwikkelaars bouwen veel master-detail-schermen, en zijn gewend om veel ADF-tabelcomponenten te gebruiken om (read-only) data te presenteren. We zouden nu eens moeten nagaan wat mogelijke use-cases zijn waar de DVT-componenten een waardevolle aanvulling op tabellen en formulieren kunnen vormen.

Zomaar een paar ideetjes

Gauges kunnen overal worden toegepast om een waarde te tonen in verhouding tot een schaal. Bijvoorbeeld: Een salarisadministratieapplicatie, met een scherm met een 'medewerkers'-tabel, heeft een kolom waar zich per rij een horizontale gauge bevindt. Deze geeft aan op welk punt een werknemer zich in zijn salarisschaal bevindt, relatief ten opzichte van het minimum en maximum van zijn schaal.
Gantts zijn een fraai alternatief voor de manipulatie van bijvoorbeeld data voor de planning van de presentaties op een conferentie of de tafelreserveringen in een restaurant. Met drag & drop kunnen 'blokjes' op verschillende plaatsen in het schema worden gezet.
Op veel schermen kan een grafiek extra, snel toegankelijke detailinformatie verschaffen, bijvoorbeeld een taartgrafiek die voor het geselecteerde (master-)record in een tabel snel een overzicht biedt tussen de verhoudingen van de details. Een voorbeeld hiervan is de verhouding tussen de taken waarop de geselecteerde werknemer uren heeft geschreven in de tijdschrijfapplicatie, of de indeling van spelers in leeftijdscategorieën voor het geselecteerde elftal.
Lijngrafieken kunnen goed inzicht geven in trends. Met behulp van bijvoorbeeld Flashback Query kan goed de ontwikkeling van het gemiddelde salaris in een afdeling of functiecategorie opgevraagd worden. In een grafiek met een lijn per afdeling ontstaat snel inzicht in de verschillen en overeenkomsten in de trends.

Conclusies

Met de Data Visualization-componenten hebben ontwikkelaars binnen ADF Faces 11g de mogelijkheid om faciliteiten - die we tot nu toe vooral kennen uit BI-toepassingen, van Excel en Oracle Discoverer tot Oracle BI Enterprise Edition en Cognos - ook in gewone, operationele applicaties te integreren. Dit past bij de trends rond applicatieontwikkeling die naast Web 2.0-aspecten als grafisch rijke, interactieve en op social networking gerichte mash-up's ook veelal geïntegreerde business intelligence omvatten. Het gevolg is dat een nieuwe groep eindgebruikers met data-analyse aan de slag kan gaan om gerichter operationele taken uit te voeren. En dat ADF-applicatieontwerpers en ontwikkelaars werk gaan doen dat tot nu toe hoofdzakelijk door BI-consultants wordt gedaan.
Nieuwe kansen dus en zeker ook nieuwe uitdagingen.
De waarde van de statistieken in dit artikel rondom 'onze' jongens moet op het moment van schrijven nog blijken, maar de nieuwe DVT's hebben in ieder geval een grote toegevoegde waarde. Dit artikel beschrijft slechts een deel van de mogelijkheden die DVT biedt. De auteurs zijn overtuigd van de kracht van DVT's, en hopen de nieuwsgierigheid van de lezer te hebben gewekt.
Met name de flitsende animatie van grafieken en de Partial Page Refresh komen op papier niet goed tot hun recht, en verdienen een 'real time' ervaring. Kortom, nu downloaden op de JDeveloper 11g homepage, en direct uitproberen! In de tweede helft van 2008 is de productierelease waarschijnlijk beschikbaar.

Links

JDev eloper 11g Homepage en DVT tutorial:
http://www.oracle.com/technology/products/jdev/11/index.html
http://www.oracle.com/technology/obe/obe11jdev/11/dvt/gant_chart_pivot_table.htm

AMIS weblog met diverse artikelen over JDeveloper en ADF 11g en diverse DVT-componenten:
http://technology.amis.nl/blog/

Luc Bors (luc.bors@amis.nl), **Frank Houweling** (frank.houweling@amis.nl), **Eric van Mourik** (eric.van.mourik@gmail.com)