Framework

Het Oracle Application Development Framework (ADF) bestaat al jaren en is inmiddels geëvolueerd tot een volwassen framework waarmee enterprise applicaties kunnen worden ontwikkeld. Desondanks is het framework bij veel ontwikkelaars nog onbekend. Aino Andriessen en Luc Bors laten je kennis maken met ADF en helpen vooroordelen over ADF de wereld uit.

Oracle ADF: onbekend maakt onbemind

Fact or Fiction: ADF is Hot !

Aino Andriessen
Principal development consultant bij AMIS.

Luc Bors
Principal development consultant bij AMIS.

Oracle JDeveloper bestaat al sinds eind jaren negentig. Destijds heeft Oracle een licentie genomen op JBuilder (van Borland), maar die oorsprong is al lang verdwenen. Het ADF Framework heeft in de vorm van BC4J (Business Components for Java) in 1999 zijn intrede gedaan . Het is begonnen als een ORM framework maar werd al snel uitgebreid met JSP web ontwikkeling. De eerste versies van het tool en het framework waren verre van volwassen.

Dit heeft er toe geleid dat er tot op de dag van vandaag nog steeds mensen zijn die op basis van die ervaring JDeveloper en ADF tot 'verboden gebied' hebben verklaard. Sinds 'those early days' is er echter heel veel veranderd en in dit geval staat deze verandering gelijk aan verbetering. Inmiddels zijn we aangekomen bij ADF 11g Release 2 (11.1.2) die in juni 2011 werd gereleased.

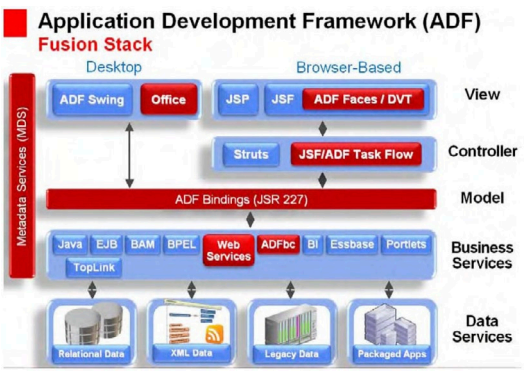
ADF is het strategisch framework van Oracle en wordt gebruikt om Oracle 'Fusion Applications' mee te ontwikkelen. In 'Fusion Applications' worden onder andere Oracle E-Business Suite, PeopleSoft, Siebel, Retek en JDEdwards samengevoegd in één nieuwe totaal oplossing. Maar dat is niet alles.

- ADF wordt gebruikt voor het maken van schermen voor de ondersteuning van Human Tasks uit BPEL processen in Oracle SOA Suite;
- ADF wordt gebruikt voor het ontwikkelen van Oracle Webcenter componenten. Webcenter is de Oracle oplossing voor enterprise 2.0 en bevat features zoals een social network, wiki, discussieforum, portals en meer.
- ADF wordt gebruikt voor het ontwikkelen van de beheermodules van Oracle voor de Oracle Database en Oracle Weblogic Server.

De kosten voor het gebruik van ADF hangen af van de situatie. Op een Oracle Applicatie Server zijn ADF runtime libraries gratis. Bij een 'non' Oracle Application Server kosten liggen de kosten rond 5000 euro per CPU. Niet veel, maar het draagt niet direct bij aan de populariteit van ADF. Een actieve lobby in de community (zie verderop) probeert het licentie model achter ADF te veranderen. Het best-case scenario is dat ADF volledig gratis wordt, maar zover is het nog niet.

Architectuur in het kort

ADF is een 'meta' framework. Dat wil zeggen dat het uit meerdere 'sub' frameworks (de rode blokken in onderstaande figuur) bestaat. ADF volgt de bewezen, industrie breed aanvaarde Model-View-Controller (MVC) architectuur en breidt deze MVC



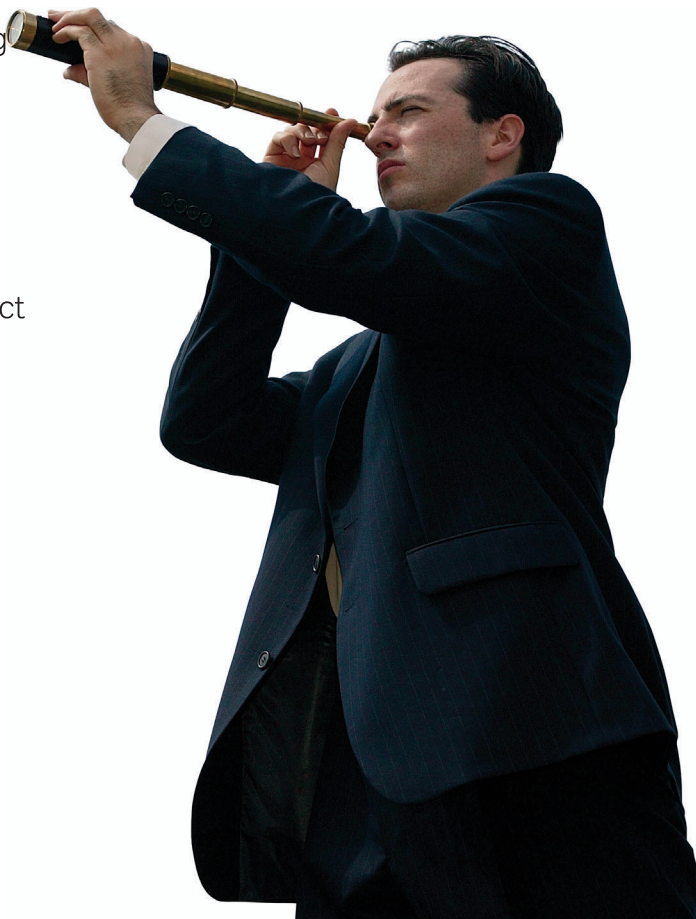
Elke laag in ADF heeft een verschillende rol en een specifieke verantwoordelijkheid.

logica
be brilliant together

So you've seen IT all before...

Jouw expertise. Daar zijn wij naar op zoek. Jij kent Java en kan onze klanten helpen om hun doel te bereiken. Heeft de klant een keten van informatiesystemen, jij bedenkt en bouwt een website waarin alles overzichtelijk samenkomt. En ook voor nieuwe technieken met Google Android of de iPhone kunnen we je inschakelen. Of voor elektrisch rijden. Bedenk jij veilige software voor betaling aan de 'pomp'? Jouw professionaliteit is de toekomst.

We vragen nogal wat van je. Kennis van het vak, enthousiasme naar de klant, doorzettingsvermogen. Logica heeft namelijk alleen de beste mensen in dienst. Onze klanten weten dat ook. Daarom kunnen wij jou de interessantste opdrachten aanbieden. Is een opdracht innovatief, erg technisch of complex? Met jouw ervaring kun je het aan.



Past dit bij je ambitie, kijk dan op www.werkenbijlogica.nl of bel naar 088-564 4000 voor een vrijblijvend kennismakinggesprek. Of solliciteer direct op één van onze vacatures.

architectuur uit met een extra laag - de Business Service Laag. Elke laag in ADF heeft een verschillende rol en een specifieke verantwoordelijkheid.
De top laag ADF view (ADFv) bevat de gebruikersinterface pagina's. De view laag wordt ontwikkeld met behulp van ADF Faces Rich Client componenten (meer dan 150). Deze zijn op standaard JSF componenten gebouwd en bevatten ingebouwde AJAX, Active Data Service (ADS) voor server push en Web 2.0-achtige functies. Naast voor de hand liggende componenten als buttons, input fields, calenders, tabellen en dergelijke zijn er ook componenten voor bijvoorbeeld menu's, toolbars, file upload, carrousel, drag and drop, spinners, sliders, voor layout en voor allerlei soorten gedrag als converters, listeners en validators. Eén van de (visueel) mooiste onderdelen van ADF Faces zijn de Data Visualisatie Componenten (DVT) zoals grafieken, kaarten, 'pivot tables' en gant charts. Naast de ADF Faces Rich Client Components, ondersteunt ADFv, standaard JSF view technologie, JSP en Swing en Microsoft Office en zijn er ADF Mobile componenten die specifiek zijn gebouwd voor mobiele toepassingen.

De ADF controller (ADFc) laag verwerkt input van de gebruiker en handelt de pagina navigatie af. Ten opzichte van plain standaard JSF biedt ADFc een verbeterde navigatie en een beter state management-model. In JDeveloper kun je grafisch en declaratief de applicatie flow uitwerken.

De ADF model (ADFm) is de centrale laag in het framework. Deze laag zorgt middels JSR-227 voor uniforme (ont-) koppeling tussen business service en user interface.

De Business Service laag regelt de toegang tot de gegevens in de Database of andere dataopslag. Deze laag bevat business logica en business rules. ADF heeft zijn eigen technologie: ADF Business Components (ADFbc) waarbij je als ontwikkelaar veel declaratieve ondersteuning krijgt voor het opzetten van de Business Service Laag.
Leuk al die lagen, maar wat doen ze nou echt, en hoe worden ze gebruikt in de praktijk ?

De Rol van de Lagen in Applicatie ontwikkeling

Eén van de misverstanden met betrekking tot ADF is dat je alleen maar een applicatie kunt bouwen door deze in elkaar te klikken. Dit komt in de praktijk zelden voor. Afhankelijk van de functionele eisen aan de applicatie komt er wel degelijk Java code om de hoek kijken, en kun je als 'hardcore' Java EE ontwikkelaar helemaal los.

De ADF Business Components Laag

Over het algemeen begint het ontwikkelen van een ADF Applicatie in de Business Service Laag. Je hoeft hier geen gebruik te maken van ADF Business Components (zie 'ADF en Anderen'), maar aangezien dit een artikel over ADF is passeren ze toch kort de revue. De korte algemeen bekende definitie: met ADF Business Components wordt de database ontsloten (views, SQL en PL/SQL program units) naar de middle tier. Maar ADFbc is meer dan dat. ADFbc bestaat uit 3 delen:

- Entity Objects; Object representatie van database tabellen, data cache en verantwoordelijk voor DML
- View Objects; Definieren de data access layer op basis van SQL, of Java, of Stored Procedures of static lists.
- Application module; de business service of service façade, verantwoordelijk voor transactie management.

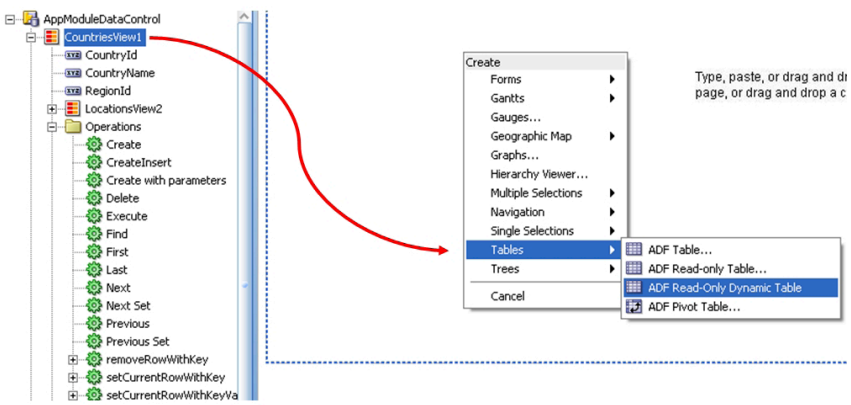
In ADFbc definiëer je validaties, default values, calculaties en andere business rules. Dit kan volledig declaratief, maar ook op basis van Java code. Sinds ADF 11g wordt ook Groovy ondersteund, waarmee het (nog) eenvoudiger is om je business rules te definiëren. Verder is het mogelijk om veel UI gerelateerde eigenschappen vast te leggen zoals labels, veldlengte en List of Values. Ik ben de eerste om toe te geven (separation of concerns) dat dit misschien niet de juiste plek is, maar met name de List of Values is wel heel erg handig.

ADFbc kun je ook prima in een Service Architectuur gebruiken. Behalve op database objecten kun je de entity objects ook baseren op webservices. En andersom kan natuurlijk ook, namelijk business componenten publiceren als webservices, waaraan je dan bijvoorbeeld Service Data Objecten (SDO) in je BPEL proces kan 'koppelen'. Je hergebruikt dus dezelfde functionaliteit en business logica voor de service toegang tot je applicatie.

De ADF Model Laag

Om gebruik te kunnen maken van ADF Model voor het koppelen van de Business Service Laag en de ViewController laag maakt ADF gebruik van een datacontrol. Een datacontrol is feitelijk een repre-

ADFbc kun je ook prima in een Service Architectuur gebruiken.

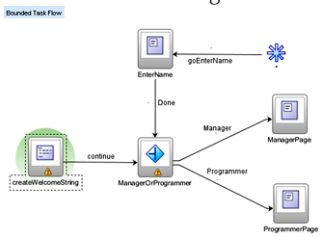


Bij gebruik van ADFbc wordt de datacontrol automatisch aangemaakt.

Voor een consistente look and feel maak je gebruik van skinning.

sentatie van business service en is gedefinieerd in een aantal XML-metadata-bestanden waarin is vastgelegd welke eigenschappen van de business service door de applicatie gebruikt kunnen worden. Bij gebruik van ADFbc wordt deze datacontrol automatisch aangemaakt. Gebruik je een andere technologie, dan kun je zelf een datacontrol creëren.
De datacontrol kun je vervolgens gebruiken om via drag and drop ADF Faces componenten te koppelen aan collecties en operaties van je business service. Standaard biedt dit bijvoorbeeld ondersteuning voor CRUD acties en paginering. Voor deze koppeling, ook wel ADF Binding genoemd, is een java API zodat operaties ook in code kunnen worden uitgevoerd. Oracle heeft geprobeerd om via JSR-227 deze werkwijze tot standaard te verheffen. Deze JSR heeft het niet gered maar deze manier van werken blijft voor een ADF applicatie zeer krachtig.

De ADF Controller Laag
Net als ieder ander JSF Framework bevat ADF een controller laag. In ADF 11g zijn de ADF Taskflows geïntroduceerd ADF Taskflows zijn het fundament van een ADF Applicatie. Kort gezegd kun je een ADF Taskflow het beste zien als een 'mini applicatie'. Zo'n 'mini applicatie' heeft zijn eigen in- en uitvoer parameters, zijn eigen 'initializers en finalizers' en een eigen memory scope (zie verderop). De taksflows kunnen hergebruikt worden en vormen de kern van ADF applicatie ontwikkeling. In vergelijking met andere JSF frameworks biedt ADF de mogelijkheid om niet alleen flows te definiëren van de ene naar de andere pagina, maar ook naar andere activiteiten zoals Java methoden. Een ADF Taskflow bevat meestal één of meerdere pagina's, method calls, navigation cases en routers waarvan er eentje als default wordt aangemerkt. JDeveloper beidt de mogelijkheid om de Taskflow 'grafisch' te ontwikkelen weer te geven met behulp van de Task-



flow diagrammer (zie figuur).

Taskflows maken het ook mogelijk om pagina's te bookmarken en om deeplinking te gebruiken.

De controller laag van ADF heeft in aanvulling op de standaard JSF memory scopes de volgende extra scopes:

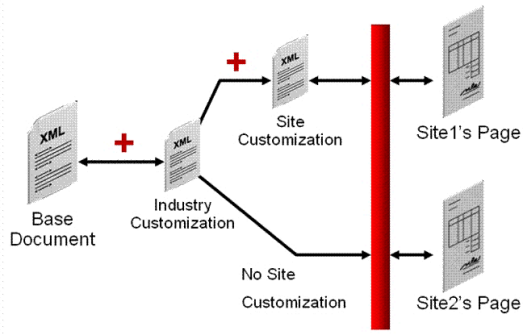
- pageFlowScope: Het object is beschikbaar tijdens de gehele taskflow en is geldig voor alle pagina's, methode's e.d. in die taskflow. Het is een beetje te vergelijken met de conversation scope van Seam.
- backingBeanScope: Wordt gebruikt voor managed beans van page fragments en declaratieve componenten. Het object is beschikbaar voor de duur tussen de HTTP-requests totdat een antwoord wordt teruggestuurd naar de client. Dit is nodig omdat er meer dan één pagina fragment of declaratieve component op een pagina kan voorkomen.
- viewScope: Het object is beschikbaar tot de ID voor de huidige view verandert. Gebruik viewScope om waarden vast te houden voor een bepaalde pagina. Terwijl je standaard de requestScope kan gebruiken om een waarde door te geven van de ene pagina naar de volgende, zal alles in viewScope verloren gaan zodra de viewID veranderd.

Ook heeft ADF de standaard JSF Request lifecycle uitgebreid met extra fases als prepareModel en validateModel die betrekking hebben op het bijwerken van het model en de onderliggende business service laag met de wijzigingen vanuit de request.

De ADF View Laag
De View laag wordt ontwikkeld met ADF full pages of ADF Page Fragments, beiden op basis van Facelets. De gangbare praktijk is om de taskflows (lees 'mini applicaties') te baseren op pagina fragments in plaats van 'full pages'. Dit biedt meer flexibiliteit voor de uiteindelijke pagina's die je daardoor dynamisch kan samenstellen aan de hand van één of meerdere taskflows, zoals bijvoorbeeld voor de single page applicatie, oftewel in ADF termen de UIShell.
Om je applicatie een consistente look and feel te geven maak je gebruik van skinning. Sinds ADF 11g Release 2 is er een visuele skin editor beschikbaar waarmee het stylen van de ADF componenten een stuk gemakkelijker wordt.
Verder kun je gebruik maken van (eventueel zelf te ontwikkelen) page templates die je kunt inzetten voor iedere pagina die je in je applicatie bouwt. Wijzigingen in de template worden direct doorgevoerd in alle pagina's die van deze template gebruik maken.

Een ander krachtig feature van ADF Faces is de Javascript API. Met de Javascript API kun je een eenvoudige manier je ADF applicatie verrijken met Javascript, waardoor interactie met bijvoorbeeld Google Maps en JQuery snel te realiseren is. Daarnaast kun je vanuit Javascript ook server code uitvoeren. Dat wil zeggen, ADF biedt de mogelijkheid om een server event te Queuën vanuit Javascript, waarna deze event in de gewone request lifecycle meegaat.

Het laatste belangrijk feature dat we de revue laten passeren is 'Customization & Personalization' en 'Designtime at Runtime'. Faciliteiten om een applicatie aan te passen aan een organisatie, rollen en individuele gebruikers. ADF gebruikt hiervoor Meta-Data Services(MDS): een generiek mechanisme om op run-time aanpassingen door te voeren op de ADF applicatie en deze te persisteren. Een voorbeeld hiervan is bijvoorbeeld de kolomvolgorde in een tabel, de positie van een splitter op het scherm, maar ook door gebruikers gedefinieerde zoekcriteria. Al deze wijzigingen zijn sowieso gedurende de sessie van de gebruiker van kracht, maar ze kunnen middels MDS worden gepersisteerd. Daarnaast is ook mogelijk om zogenoemde 'seeded customizations' te gebruiken. Zo is het mogelijk om bijvoorbeeld specifieke aanpassingen te plegen voor een bepaalde bedrijfstak, of voor één of meerdere specifieke rollen binnen een organisatie. Het is op deze manier zelfs mogelijk om run-time het gedrag van de business laag aan te passen.



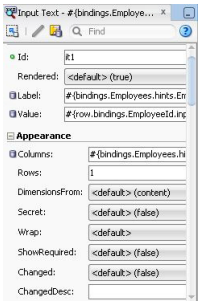
Verder kunnen customizations 'gelaagd' worden. Erg krachtig voor SAAS oplossingen. Deze customizations worden design time vastgelegd in XML bestanden. Let wel, de daadwerkelijke broncode wijzigt niet. Je beheert één code base, met daarnaast eventueel meerdere XML based customization layers.
En waar leidt dit allemaal toe ....

Een vaak gestelde vraag met betrekking tot ADF is 'Is ADF wel Sexy? Laat eens wat moois zien'. In dit artikel is dat lastig. Niet omdat ADF niet sexy is, maar omdat features zoals drag and drop, data visualisatie en de carrousel component animaties bevatten die zich niet op papier laat vangen. Je mag er zonder meer vanuit gaan dat ADF Applicaties voldoen aan de eisen die eindgebruikers stellen aan Web 2.0 en Enterprise 2.0 achtige applicaties. Mocht je behoefte hebben aan meer animatie dan kun je vrij eenvoudig gebruik maken van bijvoorbeeld jQuery om een slideshow aan je applicatie toe te voegen of om effecten zoals fading te gebruiken.

ADF ontwikkeling

Het moet gezegd worden dat JDeveloper primair gericht is op productieve en toegankelijke ontwikkeling van Fusion Middleware applicaties, oftewel ADF Faces (JSF) en Services (SCA en BPEL) en wat minder op andere Enterprise Java omgevingen. Sinds enige tijd is het ook mogelijk om ADF faces in Eclipse te ontwikkelen middels het Oracle Enterprise Pack for Eclipse. De mogelijkheden zijn nog niet zo uitgebreid als met JDeveloper, maar belangrijke features zijn al wel beschikbaar.

Aan de ene kant is het ontwikkelen van een ADF applicatie in JDeveloper heel vergelijkbaar met het ontwikkelen in een andere omgeving. Net zoals andere IDE's kent JDeveloper natuurlijk code completion, syntax highlighting, code folding, code templates, inline JavaDoc, code navigatie, refactoring, unittesten etc. En ja, het is een andere IDE, dus je zal moeten wennen als je overstapt. Aan de andere kant is ook een duidelijk andere benadering gekozen door de meer visuele en declaratieve manier van ontwikkelen.



De declaratieve manier uit zich in de vele wizards, property sheets, drag and drop opties en andere tools waarmee je veelal niet zelf code hoeft te kloppen maar die voor jou wordt 'gegenereerd'. Deze tools zijn beschikbaar in de verschillende views waarin je werkt zoals de source en de design view. Enkele voorbeelden:

- JSF componenten sleep je, eventueel in combinatie met een kleine wizard, vanuit de component palette op de pagina. Hierdoor wordt de juiste JSF code in de pagina opgenomen
- In een property sheet zijn bijvoorbeeld van die component alle attributen beschikbaar zodat je niet eindeloos hoeft te zoeken of afhankelijk bent

Voor meer animatie kun je eenvoudig gebruik maken van jQuery.

ADF ontwikkelaars zijn niet alleen maar 'dozenschuivers'.

van code completion. Deze worden synchroon gehouden met de JSF code.
- Deze property sheets zijn context aware zodat je alleen de relevante attributen voor dat component tot je beschikking hebt en bijvoorbeeld de relevante navigation rules voor een commandButton action die voor die pagina van toepassing zijn.
- EL expression builders zijn beschikbaar, zowel in de code als in de property sheets, en bieden toegang tot alle relevante objecten als bijvoorbeeld managed beans.

De pagina die je op deze manier maakt is gewone JSF code en niet een één of ander esoterisch construct dat niet meer te herkennen en te onderhouden is. Bovendien werk je in de praktijk regelmatig in die source files, waar je nog steeds de beschikking hebt over al die tools. Als ontwikkelaar kun je je hierdoor meer richten op de functionele aspecten en hoef je je minder bezig te houden met 'plumbing' en het zoeken naar de juiste opties, deze worden je immers aangereikt.

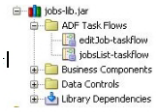
De misschien wel extreme kant van het declaratieve ontwikkelen vindt je bij de Business components. Deze worden namelijk in XML gedefiniëerd en niet direct in Java. Een belangrijke reden daarvoor is dat run-time customization op XML makkelijker is door te voeren dan op Java. Waar je in de JSF pagina's meestal in de source code werkt, doe je dat met Business Componenten eigenlijk helemaal niet en gebeurt dus vrijwel alles via wizards en property sheets, zoals:
- Het definiëren van de attribuut eigenschappen als dataType, lengte, key, verplichtheid, updatable.
- Validatie rules voor attributen en objecten, met behulp van Groovy.
- Tuning aspecten

En als je daaraan niet genoeg hebt dan gebruik je een implementatie class om de meer complexere logica in Java te coderen; iets wat regelmatig zal voorkomen.

In de praktijk is direct Java code kloppen enigszins ondergeschikt aan visuele en declaratieve ontwikkeling. Maar het is nog steeds een substantieel en noodzakelijk onderdeel van ADF ontwikkeling. Het is een misvatting dat ADF ontwikkelaars alleen maar 'dozenschuivers' zouden zijn. ADF/JDeveloper biedt voor iedere soort ontwikkelaar de mogelijkheden die hij of zij zelf zoekt of je nu hardcore Java wilt doen of liever high level pagina design. Inderdaad, als je graag zelf JDBC code schrijft dan heb je waarschijnlijk weinig te zoeken bij ADF, maar in alle andere gevallen is het meer dan de moeite waard.
Natuurlijk, het is frusterend dat het framework zich niet altijd gedraagt zoals jij dat zou willen of dat bepaalde functionaliteit niet beschikbaar is of op een andere manier dan je gewend bent. En af en toe zul je frustraties weg moeten slikken. En ja, je zal er tijd en moeite in moeten steken om het te leren en om de leercurve door te gaan. Maar wat dat betreft bevindt ADF zich in goed gezelschap, want er is geen enkel framework waar dit niet voor geldt. Wat ons betreft is het zeker de moeite waard, omdat ADF alle mogelijkheden biedt voor een productief, succesvol en uitdagend ontwikkeltraject.

ADF deployment

Tijdens ontwikkeling is de deployment cycle heel direct. Run je pagina's direct vanuit JDeveloper in de embedded Weblogic. De eerste keer opstarten kost eventjes, maar daarna is het secondenwerk. Codewijzigingen worden direct opgepakt zonder een nieuwe deploy uit te voeren (hot reload), ook configuratie files en Business Components. Debuggen gaat net zo eenvoudig en biedt bovendien toegang tot het onderliggende ADF model en ook de ADF source code is bij Oracle op te vragen. Met de Business Components tester kun je handmatig je Business Components testen en controleren op het verwachte gedrag. Natuurlijk ontslaat deze tester je niet om geen unittesten te schrijven.



In het delivery proces wordt een centrale rol ingenomen door de ADF library. In deze jar file worden meestal één of meerdere ADF taskflows gepackaged (zie figuur) die zo geheel zelfstandig herbuikbaar zijn, bijvoorbeeld in andere ADF applicaties. Maar ze kunnen ook stand-alone gedeployed worden, bijvoorbeeld voor Webcenter om on-the-fly pagina's te assembleren.

Het build process is gebaseerd op deploy profiles waarmee het artifact (jar, war, ear etc.) worden geconfigureerd en waarmee je de applicatie kan packagen en deployen, zowel vanuit JDeveloper alsook vanaf de command-line met de ojdeploy utility.

Deze methode is in de Java oertijd ontwikkeld in een tijd dat er nog geen standaard build methodiek beschikbaar was en is in de loop van de jaren doorontwikkeld en ondersteund naast alle gangbare formaten als ear, war, rar ook de Oracle specifieke formaten als mar en de ADF Library. De combinatie van ant en/of maven met ojdeploy is prima geschikt om de build te automatiseren en in bijvoorbeeld Hudson op te nemen. Sinds 11gR2 wordt ook eindelijk Maven ondersteund.

Voor code analyse is de audit tool ojaudit beschik-

baar, die is gericht op specifieke ADF aspecten, zoals ADF Business Componenten in XML, en kwaliteitseisen, maar ook de JSF en Java code kan analyseren. Het blijft echter zinnig om voor Java de bewezen tools als FindBugs en Checkstyle te gebruiken.

ADF en Anderen

Natuurlijk is ADF gebaseerd op JEE standaarden, net zoals alle andere frameworks. Sinds 11gR2 wordt JSF 2.0 ondersteund, zijn facelets de aanbevolen view technologie en worden de laatste Enterprise Java standaarden ondersteund. ADF is gecertificeerd voor Weblogic en WebSphere en het is van oudsher de mogelijkheid om naar JBoss en Tomcat te deployen, maar die worden momenteel helaas niet meer officiëel ondersteund.
Zoals eerder genoemd bestaat ADF uit verschillende onderdelen en ondersteunt het verschillende technologiën. De combinatie ADF Faces met de ADF Business Components is wel de natural fit, maar wat ook vaak gedaan wordt is om Spring, Hibernate of EJB's toe te passen in plaats van ADF Business Components. Dat wordt met name gedaan wanneer men al een ruime ervaring heeft in die technologieen. Business Components worden vaker toegepast vanuit een Oracle (Forms) achtergrond.

We kunnen ADF Faces en andere JSF component libraries wel uitgebreid met elkaar vergelijken maar op veel punten zijn ze erg vergelijkbaar, o.a. wat betreft het aantal en de veelzijdigheid van de componenten, AJAX functionaliteit, JEE standaarden, aanpassingen aan de JSF lifecycle, custom scopes etc. Maar eigenlijk is dat niet zo interessant omdat het échte onderscheidende vermogen van ADF de taskflows en customization zijn. De taskflow als zelfstandige, functionele, herbruikbare, UI component (mini applicatie) en customization waarmee de applicatie zich run-time vergaand aan de gebruiker kan aanpassen zijn uniek voor ADF en zijn wat ons betreft dé killer features!

De ADF Community

ADF wordt meer en meer gebruikt. Dit is onder andere te zien aan een toename van het aantal aanvragen voor ADF ontwikkelaars, zeker nadat Oracle ADF 11g lanceerde in 2008. ADF wordt gebruikt bij landelijke en lokale overheden, grote multinationals, banken, verzekeraars en ook ISV's. De ADF community is in opbouw, maar groeit exponentieel, zowel binnen als buiten Nederland. De toegang tot alle informatie in deze community is het makkelijkst via de ADF Code Corner op Oracle Technet (OTN). Hier vind je niet alleen informatie van Oracle, maar deze site bevat links en referenties naar de ADF Blogs, de ADF Enterprise Methodology Group (ADF-EMG) , het JDeveloper/ADF Forum en meer. Het Oracle JDeveloper en ADF Forum is een goede informatiebron. Vragen worden snel beantwoord en het is de meest bezochte van alle Oracle fora. Er is ook een actieve blogosphere waar je voor een hoop problemen en uitdagingen een oplossing kunt vinden. Een aantal van deze blogs is van Oracle medewerkers maar het merendeel is opgezet door ervaren ADF ontwikkelaars en architecten. De ADF Enterprise Methodology Group is een onafhankelijke discussiegroep over ADF Development. De ADF-EMG is gestart in 2008. Sindsdien heeft de EMG een groei doorgemaakt van 100 in 2009 naar 500 in 2010. Naar verwachting bereikt de groep dit jaar 2000 leden. De leden van deze groep zijn over het algemeen technisch architecten en senior ADF ontwikkelaars die met elkaar best-practices bespreken en opstellen. De ADF-EMG is een serieuze gesprekspartner van Oracle, en werkt samen met de productmanagers van Oracle aan de continue verbetering van ADF. Zo is er momenteel een thread waarin je een 'future wishlist' kunt voorstellen.

Samenvatting

Dé killer features van ADF zijn taskflows en customization. Samen met de zeer ruime JSF componenten library en de declaratieve aanpak in JDeveloper biedt dat een productieve omgeving voor de ontwikkeling van enterprise applicaties. De volwassenheid en toepasbaarheid van ADF staat niet meer ter discussie: 'als het geschikt is voor Oracle's Fusion Applications dan is het ook geschikt voor onze applicatie'. De declaratieve manier van ontwikkelen zal voor velen wel even wennen zijn maar je krijgt er een rijke, functionele en productieve omgeving voor terug. Al met al is ADF een meer dan valide keuze als ontwikkelplatform voor moderne enterprise applicaties. «

Resources

- ADF Enterprise Methodology Group: https://groups.google.com/forum/#!forum/adf-methodology
- ADF en JDeveloper Forum: http://forums.oracle.com/forums/forum.jspa?forumID=83
- ADF Code Corner en meer: http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index.html
- ADF Reuse: http://blogs.oracle.com/jdevotnharvest/entry/about_jsf_fragments_adf_regions_declarative_components
- AMIS technology blog: http://technology.amis.nl/blog/?s=ADF+11g

Declaratieve manier van ontwikkelen is voor velen wel even wennen.