New & Cool

Na de overnames van JBoss en Excadel heeft RedHat de RedHat Developer Studio uitgebracht. Dit pakket, gebouwd op Eclipse, stelt ontwikkelaars in staat op 4GL-achtige wijze webapplicaties te ontwikkelen op basis van Hibernate, Seam en RichFaces. Dit artikel introduceert kort de gebruikte technologieën (met name Seam) en geeft een overzicht van de belangrijkste kenmerken en voor- en nadelen van RedHat Developer Studio en het ontwikkelen ermee van geavanceerde webapplicaties.

Eenvoudige webontwikkeling met Hibernate, Seam en RichFaces

RedHat Developer Studio onder de loep genomen

Wouter van Reeven
AMIS Services B.V.

Aangezien JBoss overgenomen is door RedHat, zal het geen verrassing zijn dat in dit artikel JBoss-technologieën de boventoon voeren. De stack die door RedHat Developer Studio gebruikt wordt om webapplicaties te ontwikkelen bestaat uit Hibernate als ORM-laag, RichFaces als presentatielaag en Seam om deze twee lagen met elkaar samen te laten werken. Hibernate is het populairste ORM (Object Relational Mapping) framework en de meestgebruikte vorm van EJB3 en JPA (Java Persistence API). Zelfs vóór EJB3 de nieuwste Enterprise standaard werd, was Hibernate al zeer populair en betrouwbaar. De reden hiervoor is dat Hibernate ingezet kan worden om veel verschillende databases te benaderen, het ontwikkelaars via een heldere query-taal en aanverwante methodes in staat stelt om data in de database te beheren en Hibernate gratis, Open Source en goed gedocumenteerd is.

De standaard specificatie voor de presentatielaag van webapplicaties heet JSF (JavaServer Faces) en de JBoss-implementatie hiervan heet RichFaces. RichFaces is een rijke componentenbibliotheek en een geavanceerd framework om gemakkelijk Ajax-functionaliteit in webapplicaties te integreren. Met andere woorden, met RichFaces kun je snel en gemakkelijk moderne, interactieve webapplicaties creëren. Ondanks dat EJB3 en JSF standaarden zijn die samengevoegd zeer krachtige en schaalbare webapplicaties kunnen opzetten, kleven er ook nadelen aan deze combinatie. In de eerste plaats zijn EJB3 en JSF los van elkaar staande specificaties en is er geen standaard manier om deze twee technologieën met elkaar te laten samenwerken. Ten tweede is EJB3 sterk transactioneel (en daarmee kortstondig) opgezet, terwijl JSF vaak data over meerdere requests heen wil gebruiken. Ten derde worden JSF Backing Beans in de praktijk vaak gebruikt als doorgeefluik van data van en naar de onderliggende persistency-laag, waarmee deze beans in principe overbodig zijn. Om deze drie problemen op te lossen heeft JBoss het Seam-framework ontwikkeld.

Afbeelding 1. Het RedHat Developer Studio splash screen

Seam nader bekeken

Seam definieert een uniform componentenmodel voor alle businesslogica in (web)applicaties. Seam-componenten kunnen zowel stateless als stateful zijn. Als een component stateful is, kan deze state geassocieerd worden met een van de contexten die door Seam beschikbaar worden gesteld.

Naast de voor webapplicaties gangbare request-, session- en application-context, stelt Seam ook nog een langlopende, persistente, businessprocess-context en een conversation-context beschikbaar. De conversation-context kan ingezet worden om Seam-componenten en hun data over meerdere webrequests beschikbaar te stellen aan de gebruikers van een webapplicatie. Daarnaast heft Seam het onderscheid op tussen componenten in de presentatielaag en componenten in de businesslogicalaag. Hierdoor stelt Seam ontwikkelaars in staat om zelf te bepalen welke architectuur opgezet wordt voor een applicatie, in plaats van dat de keuze van technologieën een bepaalde architectuur opdringt. Seam maakt het mogelijk zijn componenten simultaan toegang te geven tot state die met een webrequest geassocieerd is en state die vastgehouden wordt in transactionele componenten, zonder dat het nodig is deze web-state toegankelijk te maken via method-parameters. Dit doet Seam door naast injection ook outjection te hanteren.

Injection is het mechanisme dat componenten in een applicatie door de gebruikte frameworks geïnstantieerd en beschikbaar gesteld worden. Outjection stelt ontwikkelaars in staat zelf componenten te instantiëren en daarna beschikbaar te maken in de hele applicatie. De combinatie van injection en outjection wordt bijection genoemd. Ten slotte voegt Seam ook uitbreidingen toe aan de JSF EL-taal. Deze uitbreidingen houden enerzijds in dat EL niet alleen in de webpagina's gebruikt kan worden, maar ook in Java-code. Anderzijds kunnen door deze uitbreidingen ook Java-methods aangeroepen worden die niet een setter of getter zijn.

Een eenvoudig Seam EL-voorbeeld

Stel dat we de volgende Entity hebben, inclusief getters en setters.

```
@Entity
@Table(name = "DEPARTMENTS")

public class Department implements java.io.
Serializable {

    private short departmentId;
    private String departmentName;

    // hieronder staan getters en setters
    // ....
}
```

En stel dat we de volgende Seam-class hebben.

```
@Name("departmentList")
public class DepartmentList {

         private Department department = new
Department();

    private List<Department> allDepartments;

    @In EntityManager entityManager;

    public Department getDepartment() {
        return department;
    }

    public List<Department> getAllDepartments() {
      return entityManager.createQuery("select d from
Department d").getResultList();
    }

}
```

Hierbij wordt een EntityManager door Seam geïnjecteerd door middel van de @In annotatie. We kunnen nu vanuit een JSF-pagina een department ophalen met de EL-expressie

```
#{departmentList.department}
```

en het aantal departments met

```
#{departmentList.allDepartments.size()}
```

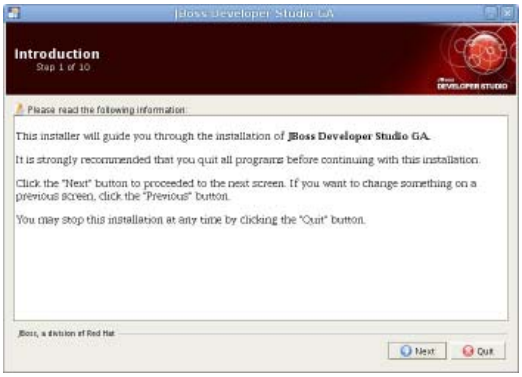
Merk op dat in de EL-expressie "#{departmentList.allDepartments.size()}" wordt gebruikt en niet "#{departmentList.allDepartments.size}". Het verschil is, dat "#{departmentList.allDepartments.size}" zou resulteren in een aanroep van getAllDepartments().getSize() in de class die de Seam-naam departmentList heeft. Maar de methode getSize() bestaat niet in een List-object en dit resulteert in een foutmelding. De uitbreiding van de EL-taal door Seam maakt het mogelijk om "#{departmentList.allDepartments.size()}" te gebruiken wat resulteert in een aanroep van getAllDepartments().size() en de size() methode bestaat wél in een List-object.

Installatie van RedHat Developer Studio

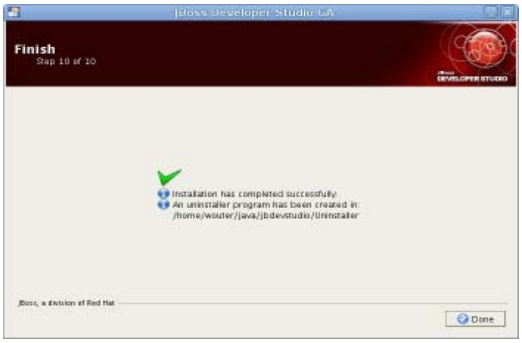
RedHat Developer Studio kan in verschillende vormen gedownload en geïnstalleerd worden. Via het RedHat Developer Network kan RedHat Developer Studio als één pakket gedownload en geïnstalleerd worden. Het downloaden en installeren van RedHat Developer Studio komt neer op het downloaden van een jar-file en deze met het volgende commando uit te voeren.

```
java -jar <jar file>
```

Na een eenvoudige wizard, waarin de mogelijkheid geboden wordt het installatieproces te configureren, wordt RedHat Developer Studio geïnstalleerd. Het is bij deze wizard belangrijk om een Java 5 JRE te selecteren en niet een Java 6 JRE. Na afloop van het installatieproces wordt aangegeven dat de installatie succesvol is verlopen. Hierna kan RedHat Developer Studio opgestart



Afbeelding 2. De installer (01-rhds-installatie-stap1.jpg)



Afbeelding 3. Succesvol geinstalleerd (02-rhds-installatie-stap10.jpg)

worden via een menu-item of via een snelkoppeling op het bureaublad.

Aan de slag met RedHat Developer Studio

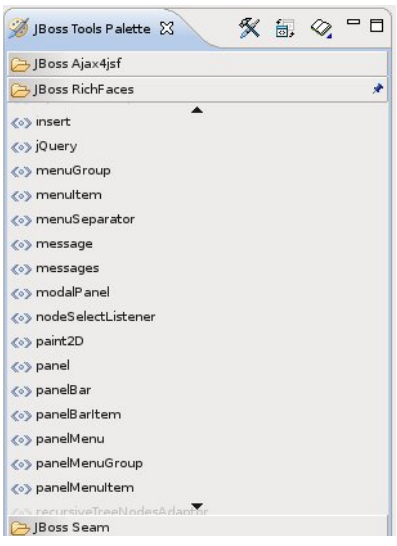
RedHat Developer Studio start Eclipse op met het Seam-perspectief geopend. Dit perspectief bevat aan Seam, Hibernate, RichFaces en JBoss Enterprise Application Server gerelateerde menu-items en panels. Zo springt rechts het JBoss Tools Palette meteen in het oog. Vanuit dit palet kunnen JSF-elementen van Seam en RichFaces gedragged en gedropt worden op de webpagina's van onze applicaties.

De eenvoudigste manier om een basisapplicatie op te zetten vanuit RedHat Developer Studio is deze een CRUD (create, update, delete) applicatie te laten genereren. Achter de schermen maakt RedHat Developer Studio gebruik van Hibernate Gen en Seam Gen, twee generatoren die op basis van een databaseschema een CRUD-applicatie kunnen genereren.

Om van deze functionaliteit gebruik te kunnen maken heeft RedHat Developer Studio een connectie met een database nodig. Via Window -> Open View -> Other -> Connectivity -> Data Source Explorer kunnen de databaseverbindingen die binnen RedHat Developer Studio bekend zijn, beheerd worden. Onderwater is RedHat Developer Studio hetzelfde als Eclipse, dus als je problemen ondervindt met het opzetten van een databaseverbinding kun je via internet zoeken naar oplossingen voor Eclipse. Ook voor het beschikbaar maken aan JBoss Enterprise Application Server van de JDBC-drivers voor de database van je keuze verwijs ik graag naar het internet. Ik maak hier gebruik van het standaard HR-schema van een Oracle XE database.

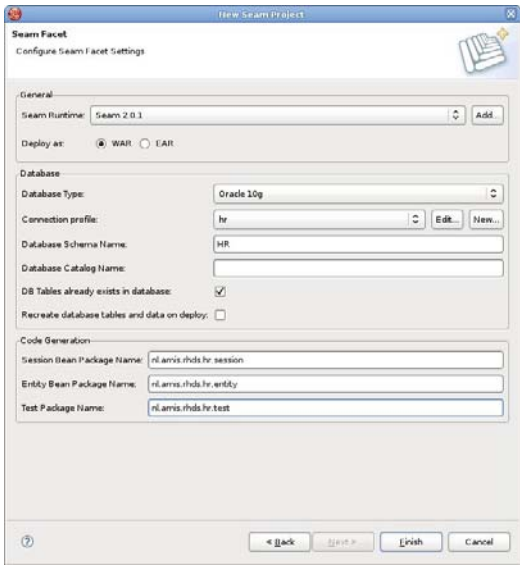
Een Seam CRUD-applicatie kan gegenereerd worden via File -> New -> Seam Web Project. Er volgt dan een wizard waarin het nieuwe project geconfigureerd kan worden. Het grootste deel van de standaardwaarden in de wizard kunnen gehandhaafd blijven, met uitzondering van de waarden in de eerste en in de laatste stap van de wizard. In de eerste stap van de wizard kan de naam van het nieuwe project opgegeven worden en kan aangegeven worden met welke versie van Seam ontwikkeld gaat worden. De keuze die hier gemaakt kan worden bestaat namelijk uit Seam 1.2.1 en Seam 2.0. Seam 2.0 heeft uitgebreidere functionaliteit dan 1.2.1 en een deel van de classes is gewijzigd ten opzichte van versie 1.2.1.
Ten tijde van het uitbrengen van RedHat Developer Studio was Seam 2.0 net uitgebracht, maar ergens is dit niet doorgedrongen tot de ontwikkelaars van RedHat Developer Studio. Seam 2.0 wordt namelijk overal in RedHat Developer Studio als Technology Preview aangemerkt. Met RedHat Developer Studio wordt dan ook Seam 1.2.1 meegeleverd en niet 2.0. Deze dient dus nog apart gedownload en geïnstalleerd te worden! De locatie van deze versie van Seam kan opgegeven worden in de laatste stap van de wizard.

In de laatste stap van de wizard kan naast de Seam-versie ook aangegeven worden welke databaseverbinding gebruikt moet worden en wat



Afbeelding 4. Het JBoss Tools-palet

Achter de schermen maakt RedHat Developer Studio gebruik van Hibernate Gen en Seam Gen



Afbeelding 5. Kiezen van de Seam-versie (06-configuratie-van0seam-laatset stap.jpg)

de namen van de packages zijn waarin de Java-classes aangemaakt gaan worden. Na het klikken op de Finish-knop, wordt de basis van het CRUD-project aangemaakt.

Wat is er aangemaakt?

In de eerste plaats zijn de bestanden die Eclipse nodig heeft voor het beheer van het project aangemaakt. Daarnaast zijn er configuratiebestanden voor Seam en Hibernate voor RedHat Developer Studio aangemaakt. Vervolgens zijn ook configuratiebestanden voor Seam en Hibernate voor JBoss Enterprise Application Server aangemaakt. Ten slotte is ook een basis voor de applicatie opgezet in de vorm van Java-classes, die Seam in de gedeployde applicatie kan gebruiken voor authenticatie en autorisatie, en enkele webpagina's waar de RichFaces-componenten in opgenomen zijn.

Onder src/action bevindt zich een Java-package zoals die aangegeven is in de laatste stap van de wizard. In mijn geval is dat nl.amis.rhds.hr.session en daarin bevindt zich onze eerste Seam class: Authenticator.java. De centrale method in deze class is authenticate() en hierin kan code opgenomen worden die de door de gebruiker ingevoerde username en password kan verifiëren tegen een database of directory service. Deze username en password worden ontsloten aan deze class door een instance van de class Identity te injecteren. Dit is aangegeven met de @In annotatie.

De username en password worden opgeslagen in de instance van de Indentity class via de login pagina. Hierin wordt via eenvoudige EL expressies de door de gebruiker ingevoerde username en password aan de Identity instance gekoppeld. Deze wordt automatisch door Seam in het leven geroepen en gehouden. Het bij deze pagina behorende bestand (login.page.xml) bevat aanvullende configuratie opties voor de JSF servlet. Standaard wordt je na succesvol inloggen naar de home pagina gerouteerd.

Ten slotte verdient ook de src/model-tak nog aandacht. Deze tak bevat op dit moment nog geen Java-code, maar wel configuratiebestanden voor JPA (persistence.xml in de META-INF directory) en Seam. Het components.properties-bestand bevat enkele configuratie-instellingen die voor de JBoss Enterprise Application Server belangrijk zijn. Indien je van plan bent jouw applicatie op een andere applicatieserver te deployen is dit de plek om wijzigingen in bijvoorbeeld de default namespace waarin JNDI look-ups voor EJB classes wordt gedaan, aan te passen.

Het is in dit stadium mogelijk de applicatie te laden in een webbrowser. Erg veel functionaliteit is er nog niet, maar het in- en uitloggen werkt, hoewel er nog geen regel zinnige code in de methode authenticate staat. De volgende stap is nu om een CRUD-applicatie te genereren.

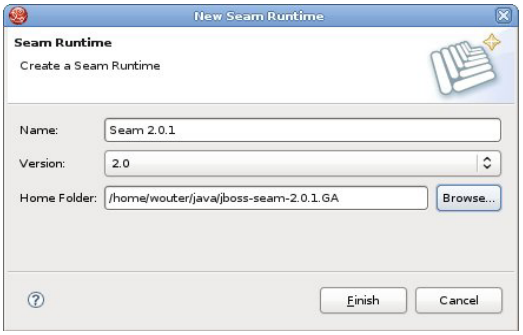
Het genereren van een CRUD-applicatie

Het genereren van een CRUD-applicatie kan gedaan worden via File -> New -> Seam Generate Entities. In tegenstelling tot wat de tekst van deze menuoptie suggereert, wordt veel meer gegenereerd dan alleen Entities. Er worden ook Seam Action-classes en webpagina's gegenereerd. Met andere woorden, een volledige CRUD-applicatie.

Aangezien Hibernate geen onderscheid maakt tussen databases en schema's, maar Oracle wel, is het hier van belang dat het hibernate-console.properties-bestand wordt aangepast vóór Seam Generate Entities wordt uitgevoerd. Hiermee zorg je er voor dat alleen de tabellen en views worden aangemaakt uit het schema dat door de Oracle-gebruiker wordt gebruikt, en niet voor de hele database! Om dit te voorkómen dien je aan het hibernate-console.properties-bestand de volgende regel toe te voegen.

```
hibernate.default_schema=HR
```

Na het uitvoeren van Seam Generate Entities is een groot aantal bestanden toegevoegd aan het



Afbeelding 6. Het zetten van de locatie van de juiste Seam-versie

Seam-project. Voor iedere tabel en view is een Entity-class aangemaakt. Indien een tabel of view een samengestelde primary key heeft wordt hier, zoals in de EJB3-specificatie staat, een aparte class voor gemaakt, en maakt de Entity gebruik van deze class voor de variabele die de @Id annotatie krijgt. Indien het schema foreign key-relaties bevat, worden deze gebruikt om OneToMany- en ManyToOne-associaties aan te brengen tussen de Entities waarvoor dit van toepassing is. In het geval dat ManyToMany-relaties opgenomen zijn in de database met een koppeltabel, zal deze met de hand opgenomen dienen te worden in de relevante Entity-classes.
Naast Entity-classes voor alle tabellen en views, worden ook EntityHome- en EntityList--subclasses aangemaakt. De EntityHome-class is een Seam-class die zorgt voor insert-, update- en delete-acties. Indien de applicatie vereist dat het standaard JPA-gedrag aangepast wordt, is dit de plek om dat te doen. De EntityList-class is een Seam-class die databaserijen beschikbaar maakt in een collectie van Entities. Om dynamische zoekqueries op te kunnen bouwen, bevatten de EntityList sub-classes array's van Strings waarin zoekrestricties in JPA Query Language-formaat opgenomen zijn. Als de applicatie vereist dat de zoekrestricties aangepast moeten worden, is dit de plek om dat te doen.

De grootste winst die de Seam Generator biedt is het genereren van webpagina's. Voor iedere Entity wordt een zoek-, een bekijk- en een edit-scherm gegenereerd. In het zoekscherm worden zoekcriteria opgegeven, waarna de resultaten getoond worden in een tabel. Via een link in de tabel kan het bekijkscherm worden geopend. In dit scherm worden standaard alle gegevens van een Entity getoond en wordt in een tabbed panel ook de child-relaties van de Entity getoond.
RedHat Developer Studio kan de gegenereerde webpagina's op drie manieren tonen. De eerste manier is alleen de source-code van de pagina. De tweede manier is een visuele representatie van de pagina. Deze lijkt heel erg op hoe de pagina uiteindelijk in een browser verschijnt. Het verschil is hier dat alle aanwezige elementen in het scherm getoond worden, dus bijvoorbeeld ook het element waar eventuele foutmeldingen getoond worden. In een browser wordt dit element alleen getoond als er ook daadwerkelijk foutmeldingen zijn. De derde manier is een combinatie van source en visuele presentatie.
De combinatie van source en visuele presentatie werkt in de praktijk heel prettig. In de visuele presentatie kun je schermelementen aanklikken waarna je deze in de source-presentatie kunt bewerken. Verder bevat de visuele presentatie een contextmenu (rechter muisknop) waarmee niet alleen het geselecteerde element bewerkt kan worden, maar ook het parent-element, diens parent-element enzovoort. Verder kunnen vanuit dit contextmenu nieuwe elementen voor, in of na het huidig geselecteerde element toegevoegd worden.

Het beïnvloeden van het generatie proces

De Entity-classes worden met behulp van de Hibernate Generator en Hibernate template-bestanden gegenereerd. Deze templates liggen opgesloten in de jar-files van Hibernate en het is daarom vrij lastig deze templates aan te passen. De enige manier is om de Hibernate-jars uit te pakken, de templates aan te passen en deze weer in jars in te pakken. Dit is erg omslachtig en foutgevoelig en daarom niet aan te bevelen. Alle overige bestanden worden met de Seam Generator en Seam template-bestanden gegenereerd. Deze templates bevinden zich gelukkig niet in jars, maar in de seam-gen directory; de directory waarin Seam geïnstalleerd is. De taal waarmee deze templates gegenereerd zijn is FreeMarker en er is uitgebreide documentatie beschikbaar voor deze template-taal. Op basis hiervan kunnen de Seam-templates eenvoudig aangepast worden, zodat de gegenereerde bestanden zo goed mogelijk overeenkomen met het gewenste eindresultaat. Hierna kun je de gegenereerde bestanden met de hand aanpassen om de applicatie helemaal volgens wens te maken.

Het verschil met andere IDE's

Het hele generatieproces dat in dit artikel beschreven is, kan ook vanaf de command-line uitgevoerd worden. De Seam-distributie bevat een seam.bat en seam-commando die vrijwel exact dezelfde output geven als via RedHat Developer Studio bereikt kan worden. Deze command-line tools genereren ook projectbestanden voor zowel Eclipse als NetBeans, zodat de projecten eenvoudig in deze IDE's geïmporteerd kunnen worden. De grote toegevoegde waarde van RedHat Developer Studio is de integratie van alle Hibernate en Seam tools in de IDE en de mogelijkheid om de gegenereerde pagina's via de visuele representatie te bekijken voor de applicatie gedeployed is. Voor zover mij bekend is er geen andere IDE die zo'n integratie van Hibernate en Seam heeft, of die dit visuele bewerken mogelijk maakt voor RichFaces-pagina's.

Een aanrader

RedHat Developer Studio is een krachtige, prettige IDE voor het ontwikkelen van webapplicaties die gebruik maken van de JBoss-stack die bestaat uit Hibernate, Seam en RichFaces. Afgezien van enkele kleine nukken die vooral de kop opsteken bij het gebruik van een Oracle-database, zou ik deze IDE willen aanraden aan iedereen die met deze JBoss-stack ontwikkelt. «