EJB3 Zugriff auf DB2 Entwicklung einer EJB3 Anwendung für WebSphere 6.1 mit RAD 7.5

© Abteilung Technische Informatik, Institut für Informatik, Universität Leipzig © Abteilung Technische Informatik, Wilhelm Schickard Institut für Informatik, Universität Tübingen

Einführung

Dieses Tutorial zeigt, wie eine einfache EJB 3 Anwendung mit Rational Application Developver 7.5 (kurz RAD7.5) erstellt und auf einem WebSphere Application Server 6.1 (kurz WAS6.1) deployed werden kann. In diesem Tutorial verwenden wir die Installation aus Tutorial 1: EJB3 Zugriff auf DB2, Installation und Konfiguration. Hier wurde bereits WAS6.1 mit dem EJB3 Feature Pack und RAD7.5 installiert.

Dieses Tutorial präsentiert eine primitive EJB 3.0 Internet Shop Anwendung. Die Anwendung besteht aus einer Session Bean, die einen Einkaufswagen (Shopping Cart) darstellt. Eine Entity Bean stellt die zum Verkauf stehenden Artikel (Items) dar. Die Artikel sind in einer z/OS DB2 Datenbank gespeichert. Die Entity Bean kann mit Hilfe der EJB 3.0 Java Persistence Architektur (JPA) auf die z/OS Architektur zugreifen. Ein Servlet stellt die Verbindung mit einem Browser her.

Das Session Bean Interface für das Servlet hat vier public methods

- Add an item to the cart
- Remove an item from a cart
- List the available items in the store
- Check out with the selected items

Einfaches Internet-Shop Modell



Websphere Application Server

Implementierungsübersicht

Die gesamte Anwendung besteht aus einem Enterprise Application Project "Shop" und zwei Unterprojekten

- JPA Project (ShopJPA)
- Dynamic Web Project (ShopWAR)

Das JPA-Projekt implementiert u.A. die Schnittstelle zu der DB2 Datenbank; es enthält ein Mapping der Entity Bean Variablen auf eine DB2 Tabelle. Außerdem implementiert es

- eine Entity Bean (ShopEJB)
- eine Session Bean (ShopEJBClient)
- eine Schnittstelle zu dem Servlet Container (ShopCart)

Das Dynamic Web Project implementiert ein Servlet mit dem Namen ShopServlet.

Zu jedem Projekt gehört ein Deployment Deskriptor in Form einer XML-Datei.

Übersicht zur Namensgebung:

Projekt	Implementierung	Name	Deployment Deskriptor	Archivtyp
EAR Project		Shop	application.xml	EAR
JPA Project	Entity Bean Entity Bean Interface	ShopJPA ShopEJB ShppEJBClient ShopCart	persistence.xml	JAR
Dyn. Web Project	Servlet	ShopWAR ShopServlet	xml	WAR

Die Entity Bean, Session Bean und deren Schnittstelle werden in ein Java Archive (JAR) verpackt.

Die Session Bean wird in ein Web Archive (WAR) verpackt.

Beide zusammen werden in ein Enterprise Archive (EAR) verpackt.

Die Entwicklung dieser Komponenten erfolgt unter dem Rational Application Developer (RAD7.5). Das Enterprise Archive (EAR) wird dann auf einem WebSphere Application Server deployed und dort ausgeführt.

Gliederung

Dieses Tutorial ist in 2 Teile gegliedert

- Teil 1 Konfigurieren von RAD7.5, des WebSphere Application Servers und der DB2 Datenbank
- Teil 2 Entwicklung einer Enterprise Anwendung

1. Konfiguration RAD7.5

- 1.1 Die Perspektiven für Java EE ändern
- 1.2 Erstellen eines Servers
- 1.3 Erstellen einer Datenbank-Verbindung
- 1.4 Erstellen einer DB2 Tabelle und Einfügen von Daten

2. Entwicklung einer Enterprise Anwendung unter RAD7.5

2.1 Erstellen des Enterprise Application Projectes.

- 2.2 Erstellen eines JPA-Projectes, erstellt auch die Entity Bean
- 2.3 Mapping der Laufzeitumgebung, bearbeitet Deployment descriptor

2.4 Erstellen eines EJB 3.0 Projektes für die Session Bean

2.5 Hinzufügen des Persistenz-Moduls zu dem Session EJB Modul.

2.6 Erstellen eines Business-Interface zwischen Servlet und Session Bean

2.7 Erstellen einer stateful Session Bean

2.8 Erstellen eines Dynamic Web Projektes und eines Servlets

2.9 Mapping der EJB Logical Reference. Erstellen des ShopWAR Deployment Descriptors

2.10 Das Projekt auf dem Server ausführen

1. Konfiguration von Rational Application Developer 7.5

1.1 Die Perspektive auf Java EE ändern

Der RAD ist ein Entwicklungstool was auf Eclipse basiert. Es bietet ein Workspace Management mit seinen Perspektiven, Editoren, Views und Plug-Ins. Wenn Sie bereits mit Eclipse gearbeitet haben, sollte es für Sie einfach sein, Enterprise-Anwendungen mit RAD zu entwerfen und zu entwickeln. Starten Sie es mit:

Start \rightarrow Programme \rightarrow IBM Software Delivery Platform \rightarrow IBM Rational Application Developer 7.5 \rightarrow IBM Rational Application Developer.



Sie können die Perspektive ändern über Window \rightarrow Open Perspective oder indem Sie auf das Open Perspective Icon klicken (rot umrandet).

1.2 Erstellen eines Servers

🔝 Problems 🧟 Tasks 🔲 Properties	👫 Servers 🛛 🙀 Data Source Ex	plorer 📔 Snippets 🗔 Annotations	🌣 🔘 🖉 🗏 🔁
Server 🔺	State	Status	
	New 🕨 🎬 Şerver		

Einer der wichtigsten Views der mit der JEE Perspektive verknüpft ist, ist der Servers View. Dort haben sie eine Übersicht Ihrer konfigurierten Server. 1kr irgendwo in dem Server View und dann 1k new \rightarrow Server.

O New Server		
Define a New Server		
Choose the type of server to	create	
Server's host name: localhos	;t	
Select the corver type:	Down	load additional server adapters
type filter text		
Apache Basic Basic BM WebSphere App WebSphere App WebSphere App WebSphere Port WebSphere Port WebSphere Port	ication Server v6.0 ication Server v6.1 ication Server v7.0 al v6.0 Server al v6.1 Server ebSphere Application Server v6.1.	
Server name:	WebSphere Application Server v6.1	stub at localhost
Server runtime environment:	Create a new runtime environment	💌 <u>Add</u>
	Cor	nfigure runtime environments
0	< Back Next >	Finish Cancel

Wählen Sie WebSphere Application v6.1 Server und klicken auf Add um die Server-Laufzeitumgebung (Runtime Environment) zu erstellen.

10	O New Serv	er Runtime Environment	XX
Def Chi	WebSphere Specify the W	Runtime ebSphere Application Server installation directory.	-
Ser	Name:		
	WebSphere A	pplication Server v6.1	ters
Sele	Installation dire	ectory:	
typ	C:\IBM\WebS	phere\AppServer Browse	
E	(For example,	Ordner suchen 🔹 💽 🔀	^
		NS Select the location of the runtime.	=
Rur		□ ③ Arbeitsplatz ▲ □ ③ 3½-Diskette (A:) ■ □ ④ Lokaler Datenträger (C:) ■ □ ② Abschlussarbeit ■ □ book ■ ■	~
Ser Serv	(?)	 ☑ Dokumente und Einstellungen ☑ ☐ fDisk ☑ ☐ IBM ☑ ☐ WebSphere 	Add
		AppServer Ordner: AppServer	nments
?		Neuen Ordner erstellen OK Abbrechen	ancel

Navigieren Sie zu dem WAS 6.1 Installationsverzeichnis c:\IBM\WebSphere\AppServer und bestätigen dies.

Aufgrund der Sicherheitseinstellungen des Servers, müssen wir die Benutzer-ID und Passwort eingeben. In diesem Beispiel sind diese beide "unilp". Klicken Sie auf Finish.

Jetzt existiert ein neuer Server-Eintrag in dem Servers-View.

1.3 Erstellen der Datenbank-Verbindung



Ein weiterer wichtiger View der Java EE Perspektive ist der Data Source Explorer. Es sollte in der gleichen Leiste zu finden sein, wie der Server View.

Wenn der View nicht vorhanden ist, 1k Window->Show View-> Data Source Explorer um ihn zu öffnen.

Window Help		
New Window New Editor	🔍 🕐 i 🏇 • 🚺 •	8 · 9 ·
Open Perspective		
Show View 🔹 🕨	🛄 Bookmarks	
Customize Perspective	📃 Console	Alt+Shift+Q, C
Save Perspective As	🗱 Data Source Explorer	
Reset Perspective	🔁 Navigator 🕏	
Close Perspective	📲 Outline	Alt+Shift+Q, O
Close All Perspectives	🖹 Problems	Alt+Shift+Q, X
Navigation 🕨	Project Explorer	
Web Browser 🔹 🕨	Properties	
Preferences	🔗 Search	Alt+Shift+Q, S
	🖧 Servers	
	🔚 Snippets	
	⁄ Tasks	
	Other	Alt+Shift+Q, Q



1kr auf Databases und New wählen

O Properties for Leia		
type filter text	Driver Properties	s 🔶 🔹 🗢 👻
Common Default Schema Filter Default Stored Procedure Filter Default Table Filter Driver Properties Version	Drivers: IBM Data Se Properties General Tracing Location: Host: Port number: Retrieve object	errer Driver for JDBC and SQLJ Default
	User name:	prak435
	Password:	••••
	Save passwor	d
	Connection URL:	jdbc:db2://leia.informatik.uni-leipzig.de:4019/51D931:retrieveMess agesFromServerOnGetMessage=true;emulateParameterMetaDataF orZCalls=1;
<		Test Connection
0		OK Cancel

Wählen Sie nun den DB2 for z/OS-Datenbank-Manager.

In diesem Tutorial wird also DB2 unter z/OS verwendet. Der Name der Datenbank ist S1D931. S1D931 als Location, leia.informatik.uni-leipzig.de als Host, 4019 als Port eingeben.

Geben Sie nun noch Ihren Benutzernamen und Passwort ein. 1k auf Finish.

Jetzt wurde eine neue Datenbank-Verbindung mit dem Namen S1D931 erstellt und erscheint in dem View. Unter der Verbindung finden Sie viele Schemata, da diese Datenbank von vielen Benutzern verwendet wird. 1kr auf diese Verbindung, und klicken Sie auf Properties.

O Properties for Leia	
type filter text	Schema Filter Properties 🔶 🔹 🚽
Common Default Schema Filter Default Stored Procedure Filter Default Table Filter Driver Properties Version	Specify a filter by selecting a predicate and entering a value. • Expression Name Starts with the characters • PRAK435 • Disable filter • Expression • Expression
0	OK Cancel

In dem nächsten Fenster wählen Sie Default Schema Filter auf der linken Seite, deaktivieren Sie Disable Filter und geben Sie Ihren Benutzernamen (case sensitive) ein. Klicken Sie auf OK. Eventuell müssen Sie den Connection View aktualisieren, dazu F5. Es gibt nur das Schema links.

1.4 Erstellen einer DB2 Tabelle und Einfügen von Daten

1kr auf S1D931, wählen Sie Open SQL Scrapbook. In den Textbereich geben folgenden Code ein, um eine neue Tabelle zu erstellen. Fügen Sie diese Daten ein:



1kr irgendwo im Text, Wählen Sie Execute all. Wenn die Tabelle erfolgreich erstellt wurde, wird der SQL Results View die folgende Meldung anzeigen:



Nun geben wir einige Daten in die neu angelegte Tabelle ein. Bitte den folgenden Code in den Text Bereich eingeben und ihn dann ausführen.

INSERT INTO ITEM VALUES	(1001 , 'Book', 20);
INSERT INTO ITEM VALUES	(1002, 'CD', 20);
INSERT INTO ITEM VALUES	(1003, 'DVD', 20)

🖹 Problems 🖉 Tasks 🔲 Properties 👫 Servers 🙀 Data Source E	Explorer 🔂 Snippets 🗔 Ann	otations 📃 Console 🔲 SQL Results 🛛 🛛 🔳 🗱 🎽 📄 🗎	
Type query expression here		Status	
Status Operation	Date Connectio	INSERT INTO ITEM VALUES (3, 'DVD', 20)	~
Succeed INSERT INTO ITEM VALUES (1, 'Book', 20)	5/11/10 2:0 51D931		
Succeed INSERT INTO ITEM VALUES (2, 'CD', 20)	5/11/10 2:0 51D931	(1 row affected)	
Succeed INSERT INTO ITEM VALUES (3, 'DVD', 20)	5/11/10 2:0 51D931		
			~
	>		>

Der SQL Results View wird Ihnen anzeigen, ob der SQL Code erfolgreich ausgeführt wurde.

2. Entwicklung einer Enterprise Anwendung

EJB3.0 verbessert die EJB2.x Spezifikation. In EJB2.x sind alle Ressourcen miteinander durch JNDI verbunden. EJB3 vereinfacht das Modell. Die meisten Ressourcen können mit einem tag @ definiert werden.

Ein typisches EJB3 Projekt besteht aus 4 Teilen:

- JPA Projekt, behält die in einer Datenbank gespeicherten Daten
- Client-Projekt, definiert wie ein Remote-Benutzer auf die Geschäftslogik zugreifen kann
- EJB-Projekt, implementiert die Schnittstellen, die in dem Client-Projekt definiert wurden
- Dynamic Web Project

2.1 Erstellen des Enterprise Application-Projektes.

O New Project	
Select a wizard Create a Java EE EAR project	
Wizards:	
type filter text	
EJB Project	^
→ ② Java Project ※ Java Project from Existing Ant Buildfile ⊃ ② Java EE	
Connector Project	
Utility Project	~
Show All Wizards.	
? < Back Next > Finish	Cancel

Das Enterprise Application-Projekt ist im Prinzip eine EAR (Enterprise Application Resource). Sie müssen später ein Dynamic Web Project dem EAR Project hinzufügen, und das Ganze dann exportieren.

Wählen Sie File \rightarrow New \rightarrow Project... In dem New Project Wizard, wählen Sie Java EE \rightarrow Enterprise Application Project

🖸 New EAR Application Project
EAR Application Project
Create a EAR application.
Project name: Shop
Project contents:
Use default
Directory: C:\wdz\workspace\Shop Browse
Target Runtime
WebSphere Application Server v6.1
EAR version
5.0
Configuration
Default Configuration for WebSphere Application Server v6.1 Modify
A good starting point for working with WebSphere Application Server v6.1 runtime. Additional facets can later be installed to add new functionality to the project.
Cancel

In dem EAR Application Project Panel, *Shop* als Projektnamen eingeben. Für Target Runtime *WebSphere Application Server v6.1* auswählen. Für EAR version *5.*0 auswählen. Wähle "Default Configuration for WebSphere Application Server v6.1" als Configuration.

1k auf Next.



Die hier gezeigte Abbildung ist eine Kopie aus Band 2, "z/OS Internet Integration", Abb. 17.2.7

Die EAR-Datei (application.xml) enthält nur den Namen der Enterprise-Anwendung und deren Bestandteile.

DD = Deployment Descriptor

.

New EAR Application Project	
Enterprise Application	—
Configure enterprise application settings.	
J2EE Module Dependencies:	
Select All Deselect All New Module	
Content Directory:	
Generate Deployment Descriptor	
? <back< th=""><th>Next > Finish Cancel</th></back<>	Next > Finish Cancel

Wählen Sie kein existierendes Modul (falls welche vorhanden sind). Generate Deployment Descriptor auswählen. 1k auf Finish.

Wenn Sie nicht bereits in der Java EE Perspektive sind, werden Sie aufgefordert, zu wechseln. Klicken Sie auf Yes.

Wundern Sie sich nicht über die Fehler (das rote Kreuz-Symbol im Shop-Projekt). Es ist sichtbar, weil die Deployment-Deskriptor-Datei (application.xml) keine Module enthält.



Die hier gezeigte Abbildung ist eine Kopie aus Band 2, "z/OS Internet Integration", Abb. 17.2.8.Die moderne Bezeichnung für Web-Container ist "Dynamic Web Project". Der Web-Container enthält auch statische HTML Seiten, welche Bestandteil der Anwendung sind.

JSP Tag Libraries bieten die Möglichkeit, JSPs frei von Java-Code zu halten. Dabei übernehmen Tags Aufgaben der Darstellungslogik wie bspw. Formatierungen, Iterationen über Arrays und Listen oder ähnliche Dinge. Durch die Einführung der Tag Libraries entstanden an vielen Stellen immer wieder die gleichen Tags. Daher lag es nahe, diese Bemühungen zu bündeln und eine einheitliche Bibliothek für die am häufigsten wiederkehrenden Aufgaben zu schaffen. Hierbei entstand der "JSP Standard Tag Library" (JSTL)-Standard.

2.2 Erstellen eines JPA-Projektes

Entity Beans des EJB2.x Standards wurden in EJB 3.0 komplett überholt. Entity Beans (auch Entities/Entitäten genannt) sind reine Java-Objekte, die mit "new" alloziiert werden können. Sie werden auf persistenten Speicher aufgesetzt (attached/detached/reattached). Entities sind nicht per remote abrufbar. (Die Definition eines Remote RMI-Objektes beinhaltet die Angabe einer Remote-Schnittstelle für das Objekt und die Erstellung einer Klasse, die diese Schnittstelle implementiert.) Auf Entities muss durch den neuen javax.persistence.EntityManager-Service zugegriffen werden. Der javax.persistence.EntityManager hat seine Wurzeln in dem populären OpenSource-Projekt "Hibernate".

Bei den Entities handelt es sich um ganz normale Java Klassen (POJOs), die kein spezielles Interface implementieren oder von einer vorgegebenen Oberklasse erben müssen. Lediglich ein Default-Konstruktor muss vorhanden sein. Dank JPA-Annotationen kann man erkennen, dass die Objekte persistiert werden.

Möchte man EJB3-konform arbeiten, so sollten die Annotationen dort angebracht werden, wo auf die Properties zugegriffen wird, spezifisch an den Attributen für *field-access* und an den Getter-Methoden für *property-access*. Dies sollte konsistent gemacht werden und die beiden Formen sollten nicht gemischt werden. (Zwischen den beiden Formen wird über die Position der @ld-Annotation unterschieden.)

Die Annotation der Attribute hat dabei den Vorteil, dass schnell erkennbar ist, was wie persistiert wird. Dies kann jedoch bei einigen JPA-Providern zu Performance-Problemen führen. Auf der anderen Seite führen Getter-Methoden, die nicht zu persistierende Werte liefern und nicht explizit von der Persistierung ausgenommen werden zu möglicherweise schwer zu findenden Programmfehlern.

JPA, die Java Persistence API bietet im Vergleich zu JDBC ein erhöhtes Abstraktionsniveau und ermöglicht damit in kurzer Zeit Anwendungen zu entwickeln, in denen Daten dauerhaft gespeichert werden müssen.

Bei dem Umgang mit Datenbanken gibt es schon sehr lange JDBC (Java Database Connectivity) als einheitliche API um die Verbindung zu Datenbanken herzustellen. Auch hier handelt es sich bereits um eine einheitliche Schnittstelle, jedoch wird darüber lediglich festgelegt, wie Verbindungen aufgebaut werden, wie eine Abfrage eingeleitet wird und wie Transaktionen manuell gesteuert werden können.

Damit bleiben zwei Probleme: Der SQL-Code muss von Hand erstellt werden, was gerade bei CRUD (Create, Update, Delete)-Anwendungen viel Produktivität kostet, und zudem ist der SQL-Code in der Regel herstellerabhängig. Ein Wechsel des Herstellers bedeutet damit, das Programm an vielen Stellen ändern zu müssen. Auch sind von Hand erstellte Abfragen nicht immer optimal: Lazy Loading von Entitäten, Caching Strategien und Transaktionssteuerung können sehr aufwendig werden. (Lazy Loading ist ein Design-Muster welches die Initialisierung eines Objekts bis zu dem Punkt verzögert, an dem die Initialisierung benötigt wird.) Dazu kommen der relativ hohe Wartungsaufwand und die Gefahr von Fehlern, die erst zur Laufzeit entdeckt werden.

JPA selber ist lediglich eine API – diese wird durch einen sogenannten JPA Provider implementiert. Hibernate, WebSphere und EclipseLink sind Beispiele für JPA Provider, die teilweise als OpenSource zur Verfügung stehen und auch kommerziell verwendet werden können.

http://www.jug-muenster.de/jpa-tutorial-jpa-einrichtung-829

Wählen Sie File \rightarrow New \rightarrow Project, dann Wähle JPA \rightarrow JPA Project. Klick Next.

O New JPA Project	
JPA Project	IDA
Configure JPA project settings.	
Project name: Shop IPA	
- Project contents:	
Use default	
Directory: C:\wdz\workspace\ShopJPA	Browse
Target Runtime	
WebSphere Application Server v6.1	New
Configuration	
Utility JPA project with Java 5.0	Modify
EAR Membership	
Add project to an EAR	
EAR Project Name: Shop	✓ New
O < Back Next > Finish	Cancel

ShopJPA als Projektnamen eingeben. Wähle Add project to an EAR, und wähle Shop als EAR Project Name. Beachten Sie die Konfiguration: Utility JPA project with Java 5.0. 1k auf Next.

O New JPA Project	
JPA Facet	IDA
Configure JPA settings.	JPA
Platform	
RAD JPA Platform	~
Connection	
51D931	✓
h.	Add connection
	Connect
Override default schema from	connection
Schema: PRAK224	►
- JPA implementation	
 Use implementation provided b 	y server runtime
OUse implementation library:	✓
	Configure default JPA implementation library
	Configure user libraries
Persistent class management	
 Discover annotated classes au 	tomatically
O Annotated classes must be liste	ed in persistence.xml
Create orm.xml	
0	< Back Next > Finish Cancel

Das JPA Facet-Panel ist das wichtigste Panel in dem JPA-Projekt-Assistenten. Wir geben an, wie ein JPA-Provider die Entitäten verwaltet und ob sie in der Datei persistence.xml aufgeführt werden oder nicht.

Woher weiß der Server, welche Datenbank für das save/update/query der Entity-Objekte benutzt werden soll? Wie konfigurieren wir das zugrunde liegende Objekt-relationale Mapping? In der Datei "persistence.xml" (normalerweise im META-INF Verzeichnis) befindet sich die Datenbankkonfiguration für den JPA-Provider. Die persistence.xml-Datei gibt Ihnen völlige Flexibilität, um den JPA-EntityManager zu konfigurieren.

Die persistence.xml-Datei ist eine Standard JPA Konfigurationsdatei. Sie muss in dem META-INF Verzeichnis, und dort innerhalb der JAR-Datei, welche die Entity Beans enthält, vorhanden sein. Die persistence.xml-Datei muss eine Persistenz-Einheit mit einem eindeutigen Namen in dem aktuellen Classloader definieren. Das Provider-Attribut spezifiziert die zugrunde liegende Implementierung des JPA-EntityManagers.

Für Connection wählen Sie "S1D931". Dies ist die DB2 Verbindung, die wir in Teil 1 erstellt haben. 1k auf Finish.

Java EE		• <u> </u>
JPA Tools		Generate DDL
Services		Generate Entities
Source		Configure Project for JDBC Deployment
Properties	Alt+Enter	Add JPA Manager beans
· · · - F - · · ·		

Erstellen einer JPA Entity (Entity Bean):

1kr auf ShopJPA-Project. Wählen Sie JPA Tools \rightarrow Generate Entities...

S1D931	~
PRAK435	~
have active connection to select schema)	
	Add connections
	Reconnect
	51D931 PRAK435 nave active connection to select schema)

Selektiere "S1D931" als Connection und ihre PRAK[xxx] als Schema. 1k auf Next.

Gene	rate Entities from Tables		
🗥 TI	ne use of the default package is dis	scouraged.	
Course			Duquia
Sound	:e folder:	ShopJPAjsrc	Browse
Pack	age:	shop (default)	Browse
			L
N	Inchronize classes in persiscence.x	(m) N	
Tab	iles:	м	
Т	able	Entity Name	Select All
E] CITY	City	
	CONTINENT	Continent	Deselect All
] COORDS	Coords	
	COUNTRY	Country	
	COUNTRY_RELIGION	CountryReligion	
	CUSTOMER	Customer	
] DEPT	Dept	
] EMP	Emp	
	IDGENERATOR	Idgenerator	
] INVENTORY	Inventory	
~] ITEM	Item	
	MEMBERSHIP	Membership	
	ORDER1	Order1	
	ORDERITEM	Orderitem	
	ORGANISATION	Organisation	
	PRODUCT	Product	
	RELIGION	Religion	
	STATISTIK	Statistik	
	SUPPLIER	Supplier	
	TEST	Test	
	XMLDOK	Xmldok.	

Shop als Package eingeben. Item auswählen in dem Generate Entities Panel. 1k auf Finish.

Wir erstellen hier ganz automatisch eine Entity Bean, ohne eine spezifische Eingabe. Der Code ist in der nächsten Abbildung wiedergegeben. Wir könnten hier die vorgefertigte Bean "Item" unverändert übernehmen. In der Praxis würde vermutlich zusätzlicher Code erforderlich sein. Dies ist der Quellcode (automatisch generiert):

```
package shop;
import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.Id;
@Entity /* define an Entity Bean */
public class Item implements Serializable {
   @Id /* define the key for the bean */
   private int id;
                               Die drei Variablen
   private String name;
                                der Entity Bean
   private int quantity;
   private static final long serialVersionUID = 1L;
   public Item() { super();
                                 }
   public int getId() { return this.id;
                                           }
   public void setId(int id) { this.id = id;
                                                }
   public String getName() { return this.name; }
   public void setName(String name) { this.name = name;
                                                          }
   public int getQuantity() { return this.quantity; }
   public void setQuantity(int quantity) { this.quantity = quantity; }
}
```

Wir fügen 2 Queries hinzu. Eine hat keine Parameter. Die andere hat den Parameter "id".

1kr auf Item dargestellt in Project Explorer. Selektiere JPA Tools->Configure JPA Entities. Item aus der Table-Liste auswählen und 1k auf Next.

O Configure JPA En	tities	
Tasks Select a JPA Entity from	the top table and a task from the left column.	
🖡 Item		
 Select a task from the le Primary Key Named Queries Relationships Concurrency C Other 	aft column to modify the associated properties. This displays queries used to retrieve JPA data. You can ad new queries or edit existing queries to return results tailored to your needs.	d Add Edit Remove
		~
0	<pre></pre>	Cancel

Named Queries auswählen und 1k auf Add.

O Add Named Query		×
الالا Add Named Query		
Add filters, order results, and select the attributes r	eturned by the query.	
Result Attributes Filter Results Order Results		
\odot Select which attributes should be returned by the	query:	
Attribute	Туре	
🗹 🐉 id	int	
	String	
v e quantity	INC	
		Û
		2
Select All Deselect All		
OUse an existing java bean to contain the results	Select Java Bean	
Query Statement:		
Named Query Name: getItem		
SELECT i FROM Item i		
		\checkmark
(?)		1

1k auf OK und dann nochmal 1k auf Add.

Ausgewählt sind wieder alle drei Attribute. "getItemByID" als Named Query Name und "Select i FROM Item i where i.id =:id " in den Text Bereich des Query Statement Panel eingeben.

1k auf OK und Finish.

Öffnen Sie den Quellcode von Item.java. Er wurde geändert. Einige neuen Zeilen Code sind in grün hier dargestellt:



2.3 Mapping der Laufzeitumgebung

2k auf persistence.xml im ShopJPA-Project (unter META-INF) um es zu editieren.

Perine the main components of the entity in this se	ection.	Set the properties for the	e selected item. Required fields are denoted b	у т.
type filter text		Name*:	ShopJPA	
Gersistence	Add	Description:		
Persistence Unit (ShopJPA)	Remove	Provider:		1000
Versistence Class (shop.Item)		Transaction Type:		~
	Up	JTA Data Source:	jdbc/leia	
	Down	Non JTA data source:	jdbc/leia	
		Exclude unlisted classes:		~
		Jar File: 🔓		Add
				Remove Up
			< >	Down
		Mapping File:		Add
				Remove
				Up
			< > >	Down

Die Java Implementierung des X/Open XA-Standards wird als "Java Transaction Architecture" (JTA) bezeichnet siehe Band 2, Abschnitt 19.1.

JTA als Transaction Type auswählen, sowohl für JTA Data Source als auch Non JTA Data Source "jdbc/leia" eingeben. (JTA ist die Java Transaction Architecture. "jdbc/leia" ist die Data Source, die wir in WAS6.1 bereits definiert haben.)

Tipp: Überprüfen Sie in der *Administrative console* den *JDNI name.* Ggf. müssen sie "jdbc/leia" abändern.

) *persistence.xml 🗙			- [
Persistence XML Editor - Sho	pJPA		
Overview Define the main components of the entity in this se	ection.	Details Set the properties for the selected item. Required fiel	ds are denoted by '*'.
type filter text		Name*:	
🖃 💮 Persistence	Add	Value*; PRAK435	
Persistence Unit (ShopJPA) Persistence Class (shop Item)	Remove		
	Up		
Property (open.jdbc.Schem	Down		
<			
sign Source			

Expandiere Persistence Unit (\rightarrow Properties, falls vorhanden, sonst: Add \rightarrow Properties) und füge eine neue Property mit dem Namen openjpa.jdbc.Schema für den Wert (Value) "PRAK[xxx]" hinzu.

1k auf den Source-Tab um den Inhalt von persistence.xml anzusehen. Dieser sollte nun so aussehen:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"</pre>
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence 1 0.xsd">
   <persistence-unit name="ShopJPA">
         <jta-data-source>jdbc/leia</jta-data-source>
         <non-jta-data-source>jdbc/leia</non-jta-data-source>
         <class>
         shop.Item
         </class>
         <properties>
              <property name="openjpa.jdbc.Schema" value="PRAK435" />
         </properties>
   </persistence-unit>
</persistence>
```

Die Änderungen abspeichern (Strg+S).

2.4 Erstellen eines EJB 3.0 Projekes für die Session Bean

In der JavaEE-Perspektive, File \rightarrow New \rightarrow Project auswählen und dann EJB \rightarrow EJB Project.

1k auf Next.

New EJB Projec	t	_ 🗆 🔀
JB Project Create an EJB Project	and add it to a new or existing Enterprise Application.	
Project name: Shop	EJB	
Project contents:		
✓ Use default		
Directory: E:\Works	spaces\RAD75ejb3new\ShopEJB1	Browse
Target Runtime		
WebSphere Applicat	ion Server v6.1	✓ New
EJB Module version -		
3.0		~
Configuration		
Default Configuratio	n for WebSphere Application Server v6.1	Modify
A good starting point Additional facets can	for working with WebSphere Application Server v6.1 later be installed to add new functionality to the project	runtime. ct.
EAR Membership		
Add project to an	EAR	
EAR Project Name:	Shop	New
0	< Back Next > Finish	Cancel

Eingabe von ShopEJB als Project Name.

Default Configuration for WebSphere Application Server v6.1 auswählen und das Projekt zu der Shop Enterprise Application hinzufügen (schon vorausgewählt). 1k auf Next.

O New EJB Project	
EJB Module	
Configure ejb module settings.	
Source Folder:	
ejbModule	
Create an EJB Client JAR module to hold the client interfaces and classes.	
ShopEJBClient	
Client JAR URI:	
ShopEJBClient.jar	
Generate deployment descriptor	
(?) < Back Next > Finish	Cancel

Auf das EJB3.0-Projekt wird mit Web-Clients zugegriffen, die mit dem Application Developer als Dynamic Web Projects entwickelt wurden. Das ShopEJBClient.jar enthält public (Business) Schnittstellen der EJBs. Es ist die einzige JAR-Datei, die vom Client (z. B. Web-Client) benötigt wird. Das EJB-Client-Projekt ist daher abhängig von den Client-Projekten, welche auf die EJB3.0-Bean zugreifen.

Falls noch nicht automatisch ausgewählt/-füllt:

Wählen Sie "Create an EJB Client JAR module to hold the client interfaces and classes", und übernehmen Sie die Default Namen.

Es ist immer eine guter Design-Ansatz, das Public Interface (das Business Interface der EJB3.0-Bean) von ihrer Implementierung (der Implementierungsklasse der EJB3.0-Bean) zu trennen. Zwei Projekte mit gut definierten Zielen macht ihre Wartung einfacher.

1k auf Finish.

2.5 Hinzufügen des Persistenz-Moduls zu dem Session EJB Modul.

Properties for ShopEJBClie	ent		_ 🗆 🔀	
type filter text	Java EE Module Dependen	cies	(=	
Resource BeanInfo Path Builders Code Coverage	This property page lets you add N or JARs contained within the sam class folders whose contents will	Manifest classpath dependencies to other J le Enterprise Application. It also supports the be added to the root of this module.	lava EE modules ne selection of	
Java Build Path	Enterprise application project nam	e: Shop	~	
Java Code Style	Available dependent JARs:			
Java Compiler Javadoc Location	References to EJB JARs with E	JB Client JARs		
⊕ Java Editor	Use EJB JARs () Use EJB dient JARs () Allow both			
Java EE Module Dependencies JDBC Connections	JAR/Module	Project	Up	
Project Facets	ShopEJB.jar	ShopEJB		
Project References	Shop JPA.jar	ShopJPA	Down	
Target Device Settings			Select All	
Targeted Runtimes			Deselect All	
TPTP JUnit Test	Manifest Class-Path (does not ind	ude contributions from classnath entries):		
Validation The VD octet	ShopJPA.jar	uu contratorio non casopaa, enereoji	~	
. Aboset			~	
<		Restore Defaults	Apply	
0		ОК	Cancel	
			Cancer	

Die Session-Bean für den Warenkorb hat Zugriff auf die Artikel der Entity-Bean. Wir fügen das JPA-Modul als eine Dependency zu dem EJB-Client-Modul hinzu.

1kr auf ShopEJBClient-Projekt und 1k auf Properties. Java EE Module Dependencies und ShopJPA.jar auswählen. 1k auf OK.

2.6 Erstellen eines Business-Interface

Eine EJB 3.0 Session Bean implementiert ein Business Interface. Wir erstellen diese Schnittstelle, ehe wir die Session-Bean erstellen.

O New Java Inter	face				
Java Interface Create a new Java in	terface.				I
Source fol <u>d</u> er: Pac <u>k</u> age: Enclosing type:	ShopEJBCliv	ent/ejbModule			Br <u>o</u> wse Bro <u>w</u> se Bro <u>w</u> se
	Cart	O defa <u>u</u> lt	⊖ pri <u>v</u> ate) pro <u>t</u> ected	<u>A</u> dd <u>R</u> emove
Do you want to add comments? (Configure templates and default value <u>here</u>)					
0				Einish	Cancel

Erstellen Sie einen Business-Interface shop.Cart in dem ShopEJBClient-Projekt (unter ejbModule).

Das Interface hat vier public methods

- public void add(int id) Item zum Einkaufswagen hinzufügen.
- public void remove(int id) Item aus dem Einkaufswagen löschen.
- public List<Item> getItems() List Vorgandene Items auflisten.
- public List<Item> checkout() Mit den Items auschecken.

Der Code ist hier dargestellt:

```
package shop;
import java.util.List;
public interface Cart {
    public List<Item> getItems();
    public List<Item> checkout();
    public void add(int id);
    public void remove( int id);
}
```

2.7 Erstellen einer Stateful Session Bean

Nach Erstellen der Interface shop.Cart erstellen wir die dazugehörige Implementation als eine Stateful Session Bean in dem ShopEJB-Projekt. Die Stateful Session Bean kann Entities in Synchronisation mit der Datenbank handhaben.

→ Fügen sie *ShopEJBClient* und *ShopJPA* projects als Dependencies zu dem ShopEJB-Projekt zu. (über die Java EE Module Dependencies in dem Properties Dialog für ShopEJB)

In dem ShopEJB-Projekt, erstellen Sie die Bean Klasse shop.CartBean. Diese implementiert das shop.Cart-Interface. Der Code ist hier dargestellt:

```
package shop;
import java.util.ArrayList;
import java.util.List;
import javax.ejb.Remove;
import javax.ejb.Stateful;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.PersistenceContextType;
import javax.persistence.Query;
@Stateful
public class CartBean implements Cart {
@PersistenceContext(type = PersistenceContextType.EXTENDED)
EntityManager em;
private List<Item> selectedItems = new ArrayList<Item>();
public void add(int id) {
      Query query = em.createNamedQuery("getItemByID");
      query.setParameter("id", id);
      Item item = (Item) query.getSingleResult();
      item.setQuantity(item.getQuantity()-1);
      selectedItems.add(item);
}
@Remove
public List<Item> checkout() {
      System.out.println("Checkout:");
      Item[] items = selectedItems.toArray(new Item[
selectedItems.size()]);
      for (int i=0; i< items.length; i++){</pre>
            Item item = items[i];
      System.out.println(item.getId() + " " + item.getName());
      }
      em.flush();
      return selectedItems;
}
public List<Item> getItems() {
      return em.createNamedQuery("getItem").getResultList();
}
public void remove(int id) {
      Query query = em.createNamedQuery("getItemByID");
      query.setParameter("id", id);
      Item item = (Item) query.getSingleResult();
      item.setQuantity(item.getQuantity()+1);
      selectedItems.remove(item); }
```

Die Stateful Session Bean verwendet einen erweiterten Persistenzkontext um die Entitäten am EntityManager angehangen zu halten. Der Persistenz-Kontext (die Menge der Entities) wird zwischen Transaktionen bewahrt, anstatt sie am Ende jeder Transaktion zu verwerfen. Die Änderungen werden in die Datenbank am Ende jeder Transaktion geschrieben. In unserem Fall geschieht dies am Ende jeder add- oder remove-Methode. Bei der Ausführung der Methode, die mit @Remove annotiert ist, wird die Stateful Bean entfernt.

In den Methoden add und remove benutzen wir die Named Query "getltemByID". Diese haben wir in der Item Entity Bean definiert.

2.8 Erstellen eines Dynamic Web-Projektes

Project name: ShopWAR	
Directory: C:\wdz\workspace\ShopWA	Browse
Target Runtime	
WebSphere Application Server v6.1	New
Dynamic Web Module version 2.4	✓
Configuration	
Default Configuration for WebSphere Application Server v6.1	🗸 (Modify)
A good starting point for working with WebSphere Application Server v6.1 later be installed to add new functionality to the project.	runtime. Additional facets can
EAR Membership	
Add project to an EAR	
EAR Project Name: Shop	✓ New

Ein Dynamic Web Project implementiert einen Web Container, der u.a. Servlets, Java Server Pages, Tag Libraries für die JSPs sowie statische HTML-Seiten enthalten kann.

Klicken Sie auf Projekt \rightarrow Neu \rightarrow Dynamic Web Project. Schreiben Sie "ShopWAR" als Project name. Wählen Sie "Default Configurateion für WebSphere Application Server v.6.1" für Configuration, und wählen Sie Add project to an EAR.

Klicken Sie auf Finish.

Öffnen Sie die Properties und wählen Sie ShopJPA und ShopEJBClient Projects als Java EE Module Dependencies.

Erstellen Sie ein Servlet shop. Shop
Servlet, indem Sie rechts klicken auf Deployment Descriptor und New
 \rightarrow Servlet

O Create Ser	vlet	
Create Servi Specify class fil	et e destination.	S
Web project: Source folder:	ShopWAR 🗸	Browse
Java package: Class name:	shop ShopServlet	Browse
Superclass:	javax.servlet.http.HttpServlet	Browse
🔲 Use an exist	ing Servlet class or JSP	
Class name;	ShopServlet	Browse
?	< Back Next > Finish	Cancel

1k auf Finish. Das ShopServlet wird nun im Editor geöffnet.

Geben Sie den folgenden Code für die Klasse ein.

```
package shop;
import java.io.IOException;
import java.util.List;
import javax.ejb.EJB;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
public class ShopServlet extends HttpServlet {
private static final long serialVersionUID = 1L;
public static final String CART_REF_NAME = "shop.cart";
public ShopServlet() {
super();
}
protected void doGet(HttpServletReguest reguest,
   HttpServletResponse response) throws ServletException, IOException {
   HttpSession session = request.getSession();
   Cart cart = (Cart) session.getAttribute(CART REF NAME);
   if (cart == null) {
          try {
                System.err.println("runs here");
                Context ctx = new InitialContext();
                cart = (Cart) ctx.lookup("java:comp/env/ejb/Cart");
                session.setAttribute(CART REF NAME, cart);
          } catch (NamingException ne) {
                throw new ServletException(ne);
         }
   /**** the business logic ****/
   java.io.PrintWriter out = response.getWriter();
   // display the available items
   out.println("<h2>ltems</h2>");
   List<Item> itemlist = cart.getItems();
   Item[] items = itemlist.toArray(new Item[itemlist.size()]);
   for (int i=0; i< items.length; i++){
          Item item = items[i]:
          out.println(item.getId() + "   " + item.getQuantity() + "   "
                + item.getName()+ "<br>");
   }
```

```
// fill the shopping cart
    cart.add(1001); // add a book to cart
    cart.add(1001); // add a book to cart
    cart.add(1003); // add a dvd to cart
    cart.add(1002); // add a cd to cart
    cart.add(1002); // add a cd to cart
    cart.add(1002); // add a cd to cart
    cart.remove(1001); // remove a book from cart
    cart.remove(1002); // remove a cd from cart
   // display the remaining items
    out.println("<h2>Remaining Items</h2>");
    itemlist = cart.getItems();
    items = itemlist.toArray(new Item[itemlist.size()]);
    for (int i=0; i< items.length; i++){</pre>
          Item item = items[i];
          out.println(item.getId() + "   " + item.getQuantity()
                + "   " + item.getName()+ "<br>");
   }
   // checkout and list the content of the shopping cart
    out.println("<h2>Checkout Cart</h2>");
    itemlist = cart.checkout();
    items = itemlist.toArray(new Item[itemlist.size()]);
    for (int i=0; i< items.length; i++){</pre>
          Item item = items[i];
          out.println(item.getId() + "   " + item.getName()+ "<br>");
    }
   // remove session data
   session.removeAttribute(CART REF NAME);
}
protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
   doGet(request, response);
}
}
```

Shop Servlet Code Teil 2

2.9 Mapping der EJB Logical Reference

Öffnen Sie den Deployment Descriptor-Editor für ShopWAR

Wechseln Sie zu Source des Web Deployment Descriptor-Editors .Fügen Sie die ejb//Cart Reference hinzu. Weil die Web Application und die Enterprise Bean Teil der gleichen Enterprise Application sind, können wir das ejb-local-ref Element benutzen, um die Referenz zu erstellen.

xml version="1.0" encoding="UTF-8"? <web-app id="WebApp_ID" version="2.4"> <display-name>ShopWAR</display-name> <servlet></servlet></web-app>			
<description></description>			
<display-name>ShopServlet</display-name>			
<servlet-name>ShopServlet</servlet-name>			
<servlet-class>shop.ShopServlet</servlet-class>			
<servlet-mapping></servlet-mapping>			
<servlet-name>ShopServlet</servlet-name>			
<url-pattern>/ShopServlet</url-pattern>			
<ejb-local-ref></ejb-local-ref>			
<description></description>			
<ejb-ref-name>ejb/Cart</ejb-ref-name>			
<ejb-ref-type>Session</ejb-ref-type>			
<local-home></local-home>			
<local>shop.Cart</local>			
<ejb-link>CartBean</ejb-link>			
<welcome-file-list></welcome-file-list>			

2.10 Das Projekt auf dem Server ausführen

1kr auf Shop-Projekt. Wählen Sie Run As \rightarrow Run on Server. wählen sie das Server Environment, das wir in Teil 1 erstellten aus.

O Run On Server	
Run On Server	
Select which server to use	-
How do you want to select the server?	
Choose an existing server	
O Manually define a new server	
Select the server that you want to use:	
type niter text	
😑 🗁 localhost	_
KI WebSphere Application Server v6.1 at localhost p. Started	
WebSphere Application Server v6.1	
Always use this server when running this project	
Reck Next > Finish	Cancel

1k auf Finish.



🖹 Problems 🤕 Tasks 🔲 Properties	🕺 🕅 Servers 🛛 🙀 Data Source E	xplorer 🛅 Snippets	Co Annotations	📮 Console			
					\$≯ () 🔊	ŧ¢
Server 🔺	State	Status					
WebSphere Application Server General Shop General ShopEJB General ShopEJBClient General ShopJPA	rt 🖡 Started	Synchronized Synchronized Synchronized Synchronized Synchronized					
👼 ShopWAR		Synchronized					

Warten Sie, bis alle Projekte gestartet wurden.

Öffnen sie einen Web Browser. Geben Sie "localhost:9080/ShopWAR/ShopServlet" ein. Sie sollten dieses Fenster sehen.

http://localhost:9080/Shop/ShopServlet - Opera	
File Edit View Bookmarks Widgets Tools Help	
E Speed Dial X http://localhost:9080/5h × +	3
[← ← → →] • ⑦ 👚 🕆 🗎 http://localhost:9080/ ▼	
Items	
1001 15 Book 1002 16 CD 1003 18 DVD	
Remaining Items	
1001 14 Book 1002 14 CD 1003 17 DVD	
Checkout Cart	
1001 Book 1003 DVD 1002 CD 1002 CD	
र्म्नर ध्रीर © र	▶∎∢ 🔤 ▼ 🔍 100% 🔻

Herzlichen Glückwunsch! Sie haben erfolgreich eine EJB 3.0 Anwendung erstellt.

Aufgaben:

- 1. Personalisieren Sie Ihr ShopServlet so, dass die Ergebnisseite Ihren Namen und Ihre PRAK[xxx]-Kennung enthält.
- 2. Erstellen Sie von dem Ergebnis im Browser ein Screenshot.
- 3. Machen Sie des Weiteren Screenshots von den RAD-Views:
 - Data Source Explorer (Schemas [Filtered] ausgeklappt)
 Servers (Shop augeklappt)
- 4. Laden Sie sie in den Abgabeordner hoch.

Viel Erfolg!