Building Java EE applications with IBM Rational Application Developer V7.5 and WebSphere Application Server V7.0

A step-by-step guide to creating Java EE V5.0 applications

Neeraj Agrawal

November 04, 2008

IBM® Rational® Application Developer for WebSphere Software Version 7.5 contains tools to create Java™ EE applications utilizing JPA and EJB) technology Version 3.0. Rational Application Developer V7.5 introduces new wizards, editors, and perspectives to create powerful applications easily. This article, targeted for intermediate users, walks you through various new features and shows you how to create an end-to-end Java EE application.

Introduction

This article introduces you to IBM® Rational® Application Developer for WebSphere Software Version 7.5, and walks you through building a JPA entity, session bean, and servlet, and then deploying them to IBM® WebSphere® Version 7.0.

Java™ Platform, Enterprise Edition Version 5.0 (Java™ EE) makes significant changes in creating enterprise applications due to support for the Java™ Persistence API (JPA), and for Enterprise Java™Beans technology (EJB) V3.0 programming model, annotation, and dependency injection.

Prerequisites

This article assumes that you have Rational Application Developer V7.5 installed with at least the WebSphere V7.0 test server.

Sample application

The sample application you are going to create has a JPA entity, a Session bean that calls the JPA, a Servlet which invokes the Session bean and a web page which calls the servlet. The database used in this application is embedded Derby database which comes along with RAD 7.5. The following diagram illustrates the sequence of interactions in the components of the application.
The sequence diagram

Follow these steps to develop the application:

1. Create a JPA project and entity, and specify mapping
2. Create an EJB V3.0 Project, and a session bean that uses a JPA Entity
3. Create a Web V2.5 project, and a servlet that invokes the session bean
4. Set a data source and run a Web page on WebSphere V7.0

Create a JPA project and entity, and specify mapping

In this example, you create a JPA entity in a separate archive (.jar). Note that the Java EE specification allows you to package JPA entities in a Web or EJB project, too. In Rational Application Developer, you can add JPA facets to a Web or EJB project to enable them for JPA entities.

Create a JPA project

1. In the Java EE perspective, select File > New > Other > JPA Project, as shown in Figure 1.
Figure 1. New project

2. Specify the Project name, as shown in Figure 2.
3. You are prompted for switching to JPA perspective, choose No. JPA perspective has views such as JPA Structure and JPA Details, since Annotations view in the Java EE perspective allows similar functionality, switching to JPA perspective is not necessary.

Figure 3. Switch perspective

Note that Rational Application Developer creates a project with these files:
The persistence unit name is set to `EmployeeJPA`.

**Create a JPA entity**

4. Right click the project and choose **New > Entity**, as shown in Figure 4.

**Figure 4. New entity**

5. Specify the package name and class name, as shown in Figure 5, and then click **Next**.

**Figure 5. JPA Entity details**
6. Create the entity fields by clicking the **Add** button in the following page, as shown in Figure 6.

**Figure 6. JPA Entity properties**

![New JPA Entity](image)

7. In the resulting Entity Fields dialog, select `java.lang.String` as the **Type** from the drop-down list, as shown in Figure 7.

**Figure 7. Entity Fields**

![Entity Fields](image)

8. Click **Finish** on the wizard, and you will see a class created with getters and setters for the fields, annotation `@Entity`, and the annotation `@Id` for the key field, as shown in Listing 1.

**Listing 1.JPA Entity Source**

```java
import java.io.Serializable;
import java.lang.String;
import javax.persistence.*;
```
/**
 * Entity implementation class for Entity: Employee
 */
@Entity
public class Employee implements Serializable {

    @Id
    private String EMP_ID;
    private String First_Name;
    private String Last_Name;
    private static final long serialVersionUID = 1L;

    public Employee() {
        super();
    }

    public String getEMP_ID() {
        return this.EMP_ID;
    }

    public void setEMP_ID(String EMP_ID) {
        this.EMP_ID = EMP_ID;
    }

    public String getFirst_Name() {
        return this.First_Name;
    }

    public void setFirst_Name(String First_Name) {
        this.First_Name = First_Name;
    }

    public String getLast_Name() {
        return this.Last_Name;
    }

    public void setLast_Name(String Last_Name) {
        this.Last_Name = Last_Name;
    }
}

Map a database

To make sure that the entity class can properly communicate with the database Table, a mapping between the two is required. JPA provides two mechanisms for mapping. One, by annotation, the other by XML (specified in orm.xml). This example uses the annotation mechanism for mapping.

9. From the Data Source Explorer view in the Java EE perspective, choose Derby Sample Connection > Connect, as shown in Figure 8.
10. Examine the Employee table, as shown in Figure 9.

Figure 9. Database schema

11. Because the schema does not match, you need to fix mappings.

12. In the Annotation view, click the Table node of the entity, Employee, and specify the following, as shown in Figure 10.
   - Table Name: EMPLOYEE
   - Schema: SAMP
13. Click the column node under **EMP_ID** and map it to the EMPNO column, as shown in Figure 11.

**Figure 11. ID field mapping**

14. Similarly, map **First_Name** and **Last_Name** to the appropriate columns.

15. Now, the Java file will look like that shown in Listing 2.

**Listing 2. Mapped Entity Source**

```java
@Entity
@Table(schema="SAMP", name = "EMPLOYEE")
public class Employee implements Serializable {

@Id
@Column(name="EMPNO")
private String EMP_ID;

@Column(name="FIRSTNME")
private String First_Name;

@Column(name="LASTNAME")
private String Last_Name;
```

16. The Java Database Connectivity (JDBC) API data source in the persistence.xml needs to be set. Open the file in Enterprise explorer, as shown in Figure 12.
17. Next, specify the data source, as shown in Figure 13 (an actual data source needs to be created: the steps are shown in the last section).

Create an EJB V3.0 Project, and a session bean that uses a JPA entity

In this section, you will create an EJB project and a session bean.

Create an EJB Project


Figure 14. Open a new EJB project
2. Specify the **Project name**, set the **EJB Module version** as 3.0, and select the **EAR Project Name**, as shown in Figure 15. For this example, choose the EAR project that was created when you created the JPA project.

**Figure 15. New EJB Project**

3. Right click the EJB project and select **Java EE Module Dependencies**, and then select **EmployeeJPA.jar**, as shown in Figure 16. This creates the manifest entry in the EJB project, and also enables the JPA entity classes visible during development.
Create a session bean

4. Right-click the **Session Beans** node in the Enterprise Explorer and select **New > Session Bean**, as shown in Figure 17.
Figure 17. Create a new session bean

5. Specify the Java package and the class name, as shown in Figure 18. In this example, the Local business interface is selected.

Figure 18. New EJB 3.0 session Bean

6. Click Finish. The EmployeeSession class and EmployeeSessionLocal business interface are created.
Invoking JPA entity

7. Add a method called `findEmployee` to the session bean class.

Listing 3. Add a method

```java
EntityManager em;

public Employee findEmployee(String empNo)
{
    Employee emp = (Employee) em.find(Employee.class, empNo);
    return emp;
}
```

8. You need to define a persistence unit to the entity manager variable. Remember that, when JPA project was created, the `EmployeeJPA` persistent unit was set in the persistence.xml. You can directly annotate it: `@PersistenceContext(name="EmployeeJPA")` and import `javax.persistence.PersistenceContext`, or you can select **Add annotation** in the Annotation view.

9. Select the bean, as shown in Figure 19.

Figure 19. Session bean in annotation view

![Image of annotation view](image)

10. Choose the annotation, as shown in Figure 20.
11. Name the persistence unit EmployeeJPA, as shown in Figure 21.

**Figure 20. Add annotation to the session bean**

![Image of Add annotation dialog]

**Figure 21. Name the persistence unit**

![Image of persistence unit configuration]

The action not only adds the annotation, but also sets the import correctly.

12. The final class looks like that shown in Listing 4.

**Listing 4. Session bean source**

```java
package com.ibm.ejb;

import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import com.ibm.entities.Employee;
import javax.persistence.PersistenceContext;

/**
 * Session Bean implementation class EmployeeSession
 */
@Stateless
public class EmployeeSession implements EmployeeSessionLocal {
```
/**
 * Default constructor.
 */
public EmployeeSession() {
    // TODO Auto-generated constructor stub
}

@PersistenceContext(unitName="EmployeeJPA")
EntityManager em;

public Employee findEmployee(String empNo){
    Employee emp = (Employee) em.find(Employee.class, empNo);
    return emp;
}

13. The business interface of the session bean needs to have the signature of the method you just added. You can do this quickly: select the method in the outline view, and then choose **Java EE > Promote Methods**, as shown in Figure 22.
14. This opens the Promote Methods dialog. Select the `findEmployee(string)` method, as shown in Figure 23.
Figure 23. Promote Methods

![Promote Methods](image)

The session bean is now ready to be consumed by the client.

Create a Web V2.5 Project, and a servlet that invokes the session bean

In this section, you will create a Web project and a servlet.

Create a Web Project

1. Choose File > New > Dynamic Web Project, as shown in Figure 24.

Figure 24. Open a dynamic Web project

![Open a dynamic Web project](image)

2. Specify the Web Project name, select the Dynamic Web Module version as 2.5, and choose the same EAR Project Name as for the JPA and EJB Project, as shown in Figure 25.
3. Select No when prompted to open the Perspective, as shown in Figure 26, Web perspective has tools for advanced web editing which we do not need in this example.

4. Right-click the Web project and select Java EE Module Dependencies from the list on the left, and then select EmployeeEJBClient.jar and EmployeeJPA.jar, as shown in Figure 27.
Create a servlet

5. Choose **New > Servlet** from the **Servlets** node in the Enterprise explorer, as shown in Figure 28.

**Figure 28. New servlet in Web Project**
6. Specify the **Java package** (com.ibm.servlets) and the **Class name** (EmployeeSearch) for the servlet, as shown in Figure 29.

**Figure 29. Create servlet**

![Create Servlet](image)

7. Modify the `doGet` method, as shown in Listing 5. The method gets employee id from the request object and uses it to pass to the session bean to find the employee information.

**Listing 5. Servlet Source**

```java
@EJB(name="Employee")
private EmployeeSessionLocal employeeSession;

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String empNo = request.getParameter("empid");
    Employee emp = employeeSession.findEmployee(empNo);
    if(emp != null){
        response.getWriter().println(emp.getFirst_Name() + " " + emp.getLast_Name());
    }else{
        response.getWriter().println("Employee with id =" + empNo + "couldn't be found");
    }
}
```

8. Create an HTML page in the Web Project that has input field for employee id, and a Submit button, as shown in Listing 6.
Set data source and run Web page on WebSphere V7.0
In this section, you will set the data source and run a Web page.

Setting dataSource
1. Right-click the EAR project, EmployeeEAR, and select **JavaEE > Open WebSphere Application Server Deployment**, as shown in Figure 30.

Figure 30. WebSphere Deployment page

2. In the JDBC provider section, click **Add** and select **Database type** (Derby) and **JDBC provider type** (Derby JDBC Provider), as shown in 31.

Figure 31. Create JDBC Provider dialog
3. Name the provider (DerbyDB) and select the driver and the **Class path** (IBM\SDP75\runtimes \base_v7\derbylib\derby.jar), as shown in Figure 32.

**Figure 32. Provide JDBC details**

![Create JDBC Provider](image)

4. Now create the data source by clicking Add, as shown in Figure 33.

**Figure 33. Data source section**

![Data source defined in the JDBC provider selected above](image)

5. Select **Derby JDBC Provider** and **Version 5.0 data source**, and click the **Next** button, as shown in Figure 34.
6. Specify the data source name and the JNDI name, as shown in Figure 35. Remember to use the same name JNDI name as you used in the persistence.xml JPA.
7. Locate the sample Derby database in your workspace, and enter the path in the databasename property (workspace\EmployeeSample\metadata\plugins\com.ibm-datatools.db2.cloudscape.driver\SAMPLE), as shown in Figure 36.

8. Right-click in the EmployeeSearch.html and choose Run As > Run on Server, as shown in Figure 37.
Figure 37. Run on server

The EAR is added to the server, as shown in Figure 38.

Figure 38. EAR added to the Server

Enter the EmployeeID and press Submit, as shown in Figure 39.
Figure 39. Web page running

11. The result is displayed on the screen.

Figure 40. Web page results

What you have learned

Java EE V5.0 simplifies enterprise application development. This article showed you how to use Java Persistence API, EJB V3.0, and the tools provided in Rational Application Developer V7.5 to build these enterprise artifacts.
### Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample Project used in this article</td>
<td>EmployeeSample.zip</td>
<td>35KB</td>
</tr>
</tbody>
</table>
Related topics

- The developerWorks article Developing Web applications with the Java Persistence API and JavaServer Faces, by Tom Mutdosch, demonstrates JPA Web tools in Rational Application Developer v7.5.
- The developerWorks article Web 2.0 Application Development using JPA, AJAX, and Dojo tools in Rational Application Developer Version 7.5, by Jarett Stein, explains how to create a simple end-to-end Web application using the JPA, AJAX, and Dojo tools available in IBM Rational Application Developer for WebSphere Software Version 7.5.
- Understand more about JavaServer Faces tooling in Rational Application Developer Version 7.5 by reading this article What's new in JavaServer Faces tooling in Rational Application Developer Version 7.5?.
- Understand more about how Rational Application Developer Version 7.5 provides solutions to the many challenges that software teams face when tasked to deliver Web and service-oriented applications by reading this article Why Rational Application Developer for WebSphere Software Version 7.5?.
- Understand more about how the new functionality of the UML Modeler component common to both IBM Rational Software Architect Standard Edition Version 7.5 and IBM Rational Software Architect for WebSphere Software Version 7.5 by reading this article Using the new features of UML Modeler in IBM Rational Software Architect Version 7.5.
- Subscribe to an IBM developerWorks newsletter.
- Download a free IBM product evaluation.
- Try IBM Bluemix free for 90 days. Bluemix is a cloud platform as a service (PaaS). Bluemix supports several programming languages and services as well as integrated DevOps to build, run, deploy and manage applications on the cloud.

© Copyright IBM Corporation 2008
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)