



Create an EJB 3.0 application

Contents

Create an EJB 3.0 application 1

Create an EJB 3.0 application 1

Lesson 1.1: Create an EJB 3.0 project 2

Lesson 1.2: Create required classes and interfaces for
the StatelessCounterBean.java class 3

Lesson 1.3: Create an Entity class and a data source
for data persistence 6

Lesson 1.4: Create a Web project to test your
application 7

Create an EJB 3.0 application summary 9

Create an EJB 3.0 application

This tutorial describes how to create an EJB 3.0 application that contains an EJB 3.0 project and a dynamic Web project to display a counter program.

Learning objectives

This tutorial enables you to create:

- An EJB 3.0 project, called EJBCounterSample, with an EJB 3.0 stateless session bean (with both interface and implementation classes) and a JPA 1.0 entity class
- A Web project (EJBCounterWeb), and with a Java ServerPages (JSP) page and a utility Java class
- A Java EE 5 application (EJBCounterSampleEAR) with the EJB and Web projects

Time required

30 minutes

Related information



View the PDF version

Counter EJB 3.0 sample

Create an EJB 3.0 application

This tutorial describes how to use the EJB 3.0 specification to create, deploy and run a simple EJB 3.0 application that increments a counter.

This tutorial is divided into several exercises that must be completed in sequence for the tutorial to work properly. This tutorial teaches you how to use the EJB 3.0 specification to create an EJB project that includes an EJB bean, remote and local interfaces, as well as a Servlet and a Faces JSP to deploy the application on the server. While completing the exercises, you will:

- Use the EJB 3.0 specification to create, deploy and run a simple EJB 3.0 application that increments a counter.
- Create a Java™ project, EJBCounterSample with an EJB 3.0 stateless session bean (with both interface and implementation classes), StatelessCounterBean.java, and a JPA 1.0 entity class JPACounterEntity.java.
- Create Web project (EJBCounterWeb), and with a Java ServerPages (JSP) page and a utility Java class
- Create EAR application, EJBCounterSampleEAR, containing the EJB and Web projects.

Time required

This tutorial should take approximately 30 minutes to finish. If you explore other concepts related to this tutorial, it could take longer to complete.

Skill level

Experienced

Audience

This tutorial is intended for users who are familiar with Enterprise JavaBeans, EJB 3.0 in particular, and Java EE 5 technology.

System requirements

To complete this tutorial, you need to have the following tools and components installed:

-
- **The WebSphere Application Server 6.1 Fixpack for EJB 3.0 installed..**
- **A clean workspace.**

Prerequisites

In order to complete this tutorial end to end, you should be familiar with:

- Java EE 5 and Java programming
- Basic Enterprise JavaBeans (EJB) and EJB 3.0 concepts

Lesson 1.1: Create an EJB 3.0 project


This lesson leads you through the detailed steps to create an EJB 3.0 project that you will use to contain your EJB session bean.

In order for you to create an EJB 3.0 project, you need to have the WebSphere® Application Server version 6.1 with the EJB 3.0 Feature Pack installed and to have a profile created for this server.

1. Create a profile for the WebSphere Application Server version 6.1 with the EJB 3.0 Feature Pack installed:

- a. Open the Servers view by selecting **Window → Show View → Servers**.
- b. Define a new EJB 3.0 server by right clicking the Servers view and selecting **New → Server**. Follow the instructions in the **New Server** wizard, ensuring that you select the WebSphere Application Server version 6.1 with the EJB 3.0 Feature Pack.

2. Create an EJB 3.0 project:

- a. If the Java EE icon, , does not appear in the top right tab of the workspace, you need to switch to the Java EE perspective. From the menu bar, select **Window > Open Perspective > Other**. The Select Perspective window opens.
- b. Select **Java EE**. Click **OK**. The Java EE perspective opens.
- c. In the Java EE perspective, select **File → New → Other → EJB → EJB project**.
- d. In the **Project name** field, type `EJBCounterSample`. Select **Add project to EAR**, and **Next**.
- e. On the Project Facets page, use these values:
 - EJB Module 3.0
 - Java 5.0
 - Java Persistence
- f. Accept the other default values and click **Finish**.
- g. On the EJB Module page, clear **Create an EJB Client JAR module to hold the client interfaces and classes**.
- h. On the JPA Facet page, accept the defaults, and click **Finish**.

3. Add a Java class, @Stateless annotation:

- a. In the Project Explorer view, right click the `EJBCounterSample` project and select **New → Class**.
- b. Accept the default **Source folder** (`EJBCounterSample/ejbModule`). In the **Package** field, type `com.ibm.websphere.ejb3sample.counter`, and in the **Name** field, type `StatelessCounterBean`.
- c. Click **Finish**.

- d. Your StatelessCounterBean class opens in the Java Editor. Add the EJB 3.0 annotation to generate a session bean by adding @Stateless:
- e. When you press CTRL+S to save, you can see a quick fix icon beside the @Stateless line.
- f. Right click the quick fix icon and select **Quick Fix**:
- g. Select **Import 'Stateless' (javax.ejb)** and press CTRL+S to save:

The required dependencies are automatically added to the source code.

Tip: A shortcut to using the Quick Fix is to type @Sta, and press CTRL+Spacebar. This will present you with two options, and you simply select **@Stateless - javax.ejb**.

The required dependencies are automatically added to the source code.

You now are ready to move on to Exercise 1.2, Create required classes and interfaces for the StatelessCounterBean.

Lesson 1.2: Create required classes and interfaces for the StatelessCounterBean.java class

Lesson 1.2 steps you through the creation of required classes and interfaces for the StatelessCounterBean.java class.

Before you begin, you must complete Lesson 1.1. In this lesson you will

- Add @Interceptors annotation and required dependencies.
 - Create an Audit.java Java class.
 - Add code to your StatelessCounterBean.java class
 - Create a LocalCounter.java interface and a RemoteCounter.java interface
1. Open your StatelessCounterBean.java class in the Java Editor.
 2. Under the @Stateless annotation, type @Interceptors.
 3. When you press CTRL+S to save, you can see a quick-fix icon beside the @Interceptors line. Right click the quick-fix icon and select **Quick Fix**:
 4. Select **import javax.interceptor.Interceptors**, and press CTRL+S to save.
 5. Right click the quick-fix icon and select **Quick Fix**. Select **Add missing attributes** and replace (**value={null}**) with (**Audit.class**), press CTRL+S to save.
 6. Right click the quick-fix icon and select **Quick Fix**. Select **Create class 'Audit'**. In the Create a new Java class page, accept the defaults and click **Finish**.
 7. In the Java editor for Audit.java, replace the default code with this code, and press CTRL+S to save:

```
// This program may be used, executed, copied, modified and distributed
// without royalty for the purpose of developing, using, marketing, or distributing.

package com.ibm.websphere.ejb3sample.counter;

import java.io.Serializable;

import javax.interceptor.AroundInvoke;
import javax.interceptor.InvocationContext;

public class Audit implements Serializable {
```

```

private static final long serialVersionUID = 4267181799103606230L;

@AroundInvoke

public Object methodChecker (InvocationContext ic)
throws Exception
{
    System.out.println("Audit:methodChecker - About to execute method: " + ic.getMethod());
    Object result = ic.proceed();
    return result;
}
}

```

8. Open your StatelessCounterBean class in the Java Editor and **replace this code**, and press CTRL+S to save:

```

public class StatelessCounterBean {

}

```

with this code:

```

public class StatelessCounterBean implements LocalCounter, RemoteCounter {

    private static final String CounterDBKey = "PRIMARYKEY";

    // Use container managed persistence - inject the EntityManager
    @PersistenceContext (unitName="Counter")
    private EntityManager em;

    public int increment()
    {
        int result = 0;

        try {

            JPACounterEntity counter = em.find(JPACounterEntity.class, CounterDBKey);

            if ( counter == null ) {
                counter = new JPACounterEntity();
                counter.setPrimaryKey(CounterDBKey);
                em.persist( counter );
            }

            counter.setValue( counter.getValue() + 1 );
            em.flush();
            em.clear();

            result = counter.getValue();

        } catch (Throwable t) {
            System.out.println("StatelessCounterBean:increment - caught unexpected exception: " + t);
            t.printStackTrace();
        }

        return result;
    }

    public int getTheValue()
    {
        int result = 0;

        try {
            JPACounterEntity counter = em.find(JPACounterEntity.class, CounterDBKey);

```



```

        if ( counter == null ) {
            counter = new JPACounterEntity();
            em.persist( counter );
            em.flush();
        }

        em.clear();

        result = counter.getValue();
    } catch (Throwable t) {
        System.out.println("StatelessCounterBean:increment - caught unexpected exception: " + t);
        t.printStackTrace();
    }

    return result;
}
}

```

9. Right click the quick-fix icon beside the line `public class StatelessCounterBean`, and select **Create Interface 'LocalCounter'**:

10. Your `LocalCounter.java` class opens in the Java Editor. Replace all the code with this code, and press CTRL+S to save:

```

// This program may be used, executed, copied, modified and distributed
// without royalty for the purpose of developing, using, marketing, or distributing.

package com.ibm.websphere.ejb3sample.counter;

import javax.ejb.Local;

@Local
public interface LocalCounter {
    public int increment();
    public int getTheValue();
}

```

11. Return to the `StatelessCounterBean.java` class in the Java editor.
12. Follow the same steps for the Remote Counter error, and replace all the code in `RemoteCounter.java` with this code, and press CTRL+S to save:

```

// This program may be used, executed, copied, modified and distributed
// without royalty for the purpose of developing, using, marketing, or distributing.

package com.ibm.websphere.ejb3sample.counter;

import javax.ejb.Remote;

@Remote
public interface RemoteCounter {
    public int increment();
    public int getTheValue();
}

```

13. Return to the `StatelessCounterBean.java` class in the Java editor.
14. Right click the quick fix icon beside `@PersistenceContext`, and select **Quick Fix**. Select **Import 'PersistenceContext' (javax.persistence)**, and press CTRL+S to save.
15. Right click the quick-fix icon beside `@EntityManager`, and select **quick-fix**. Select **Import 'EntityManager' (javax.persistence)**, and press CTRL+S to save.
16. Right click the quick-fix icon beside `JPACounterEntity`, and select **Quick Fix**. Select **Create class 'JPACounterEntity.java'**.

You now are ready to move on to Exercise 1.3, Create an Entity class and a database for data persistence.

Lesson 1.3: Create an Entity class and a data source for data persistence

Lesson 1.3 leads you through the creation of an Entity class and a database for data persistence.

Before you begin, you must complete Lesson 1.2. In this lesson you will

- Add code to the entity class, JPACounterEntity.java.
- Create a database, EJB3SampleDB, to persist the counter data.

1. Add code to the entity class:

- a. Open JPACounterEntity.java in the Java editor, replace all the code with this code, and press CTRL+S to save:

```
// This program may be used, executed, copied, modified and distributed
// without royalty for the purpose of developing, using, marketing, or distributing.

package com.ibm.websphere.ejb3sample.counter;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="EJB3COUNTERTABLE")

public class JPACounterEntity {

    @Id
    private String primarykey = "PRIMARYKEY";

    private int value = 0;

    public void setValue( int newValue )
    {
        System.out.println ("JPACounterEntity:setValue = " + newValue);
        value = newValue;
    }

    public int getValue()
    {
        System.out.println ("JPACounterEntity:getValue = " + value);
        return value;
    }

    public void setPrimaryKey( String newKey )
    {
        System.out.println ("JPACounterEntity:setPrimaryKey = '" + newKey + "'");
        primarykey = newKey;
    }

    public String getPrimaryKey()
    {
        System.out.println ("JPACounterEntity:getPrimaryKey = '" + primarykey + "'");
        return primarykey;
    }

}
```

- b. In the Project Explorer view, navigate to *EJBCounterSample/ejbModule/META-INF/persistence.xml* and double click the file in the Editor. Select source, and replace all the code with this code:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
    <persistence-unit name="Counter">
```

```

        <jta-data-source>jdbc/EJB3SampleDatasource</jta-data-source>
        <class>com.ibm.websphere.ejb3sample.counter.JPACounterEntity</class>
        <exclude-unlisted-classes>true</exclude-unlisted-classes>
    </persistence-unit>
</persistence>

```

2. Define a data source

- a. In the Project Explorer view, right click EJBSampleEAR, and select **Open WebSphere Application Server Deployment**:
- b. In the Websphere Deployment editor, select **Derby JDBC Provider (XA)**, and in the **Data source defined in the JDBC provider selected above** field, click **Add**:
- c. In the **Create Data Source** page, select Derby JDBC Provider (XA), and click **Next**:
- d. On the Select type of JDBC provider to create page, in the **Name** field, type EJBCounterSample Data source. In the **JNDI name** field, type jdbc/EJB3SampleDatasource, and click **Next**:
- e. In the **Create Resource Properties** page, highlight **databaseName** property field, and in the **Value** field, type databases/EJB3SampleDB, and click **Finish**:

You now are ready to move on to Exercise 1.4, Create a Web project to test your application.

Lesson 1.4: Create a Web project to test your application

Lesson 1.4 leads you through the creation of a Web project to test your application.

Before you begin, you must complete Lesson 1.3. In this lesson you will

- Create a Web project, EJBCounterWeb.
 - Create a webpage, EJBCount.jsp.
 - Create a Servlet, EJBCount.java
 - Run the Servlet to test your application.
1. In the Java EE perspective, select **File** → **New** → **Other** → **Dynamic Web Project**.
 2. In the Dynamic Web Project page, in the **Project name** field, type EJBCounterWeb.
 3. Accept the other defaults and click **Finish**. If asked to **Open associated perspective?**, click **No**.
 4. Right click the EJBCounterWeb project, and select **New** → **Web page**.
 5. On the **New Web page**, in the **File name** field, type EJBCount.jsp.
 6. In the Source view of the Web page editor, replace all the existing code with this code, and press CTRL+S to save:

```

<%@page session="false"%>
<HTML>
<HEAD>
<TITLE>IBM WebSphere EJB3 and JPA1 Counter Sample</TITLE>
<BODY bgcolor="cornsilk">
<H1>EJB 3.0 and JPA 1.0 Counter Sample</H1>
<P>
<B>
This application communicates with the WebSphere Application Server using http requests to increment a stateless EJB
</B>
<FORM METHOD=POST ACTION="counter">
<BR/>
<%
    response.setHeader("Pragma", "No-cache");
    response.setHeader("Cache-Control", "no-cache");

```

```

        response.setDateHeader("Expires",0);
        String msg = (String) request.getAttribute("msg");
        if (msg == null) msg = "";
    %>
    <B>Click on the Increment button to increment the count</B>
    <BR/><BR/>
    <INPUT TYPE=SUBMIT VALUE="Increment">
</FORM>
<H3><%=msg%></H3>
</BODY>
</HTML>

```

7. Right click the EJBCounterWeb project, and select **New** → **servlet**.
8. On the **New Servlet** page, in the **Java package** field, type `com.ibm.websphere.ejb3sample.counter`.
9. In the **Class name** field, type `EJBCount`, and Click **Next**:
10. On the Enter Servlet deployment descriptor specific information page, in the **Name** field, type `EJB Count Servlet`. In the **URL mappings** field, edit the existing mapping, and replace it with `/counter`, and click **Finish**:
11. Right click the EJBCounterWeb project, and select **Properties**.
12. Select **J2EE Module Dependencies**, and click the `EJBCounterSample.jar` file, and click **Okay**:
13. Expand **EJBCounterWeb** → **Java Resources: src** → **com.ibm.websphere.ejb3sample.counter**, and double click the `EJBCount.java` file. It opens in the Java Editor.
14. Replace the existing code with the following code, and press CTRL+S to save:

```

package com.ibm.websphere.ejb3sample.counter;

// This program may be used, executed, copied, modified and distributed
// without royalty for the purpose of developing, using, marketing, or distributing.

import java.io.IOException;

import javax.ejb.EJB;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * This servlet demonstrates an EJB3 counter bean with JPA.
 */

public class EJBCount extends HttpServlet {

    private static final long serialVersionUID = -5983708570653958619L;

    // Use injection to get the ejb
    @EJB private LocalCounter statelessCounter;

    public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
        String msg = null;
        int ejbCount = 0;

        try {
            ejbCount = statelessCounter.getValue();
        }
        catch (RuntimeException e) {
            msg = "Error - getValue() method on EJB failed!";
            e.printStackTrace();
        }
        msg = "EJB Count value for Stateless Bean with JPA: " + ejbCount;
    }
}

```

```

// Set attributes and dispatch the JSP.
req.setAttribute("msg", msg);
RequestDispatcher rd = getServletContext().getRequestDispatcher("/EJBCount.jsp");
rd.forward(req, res);
}

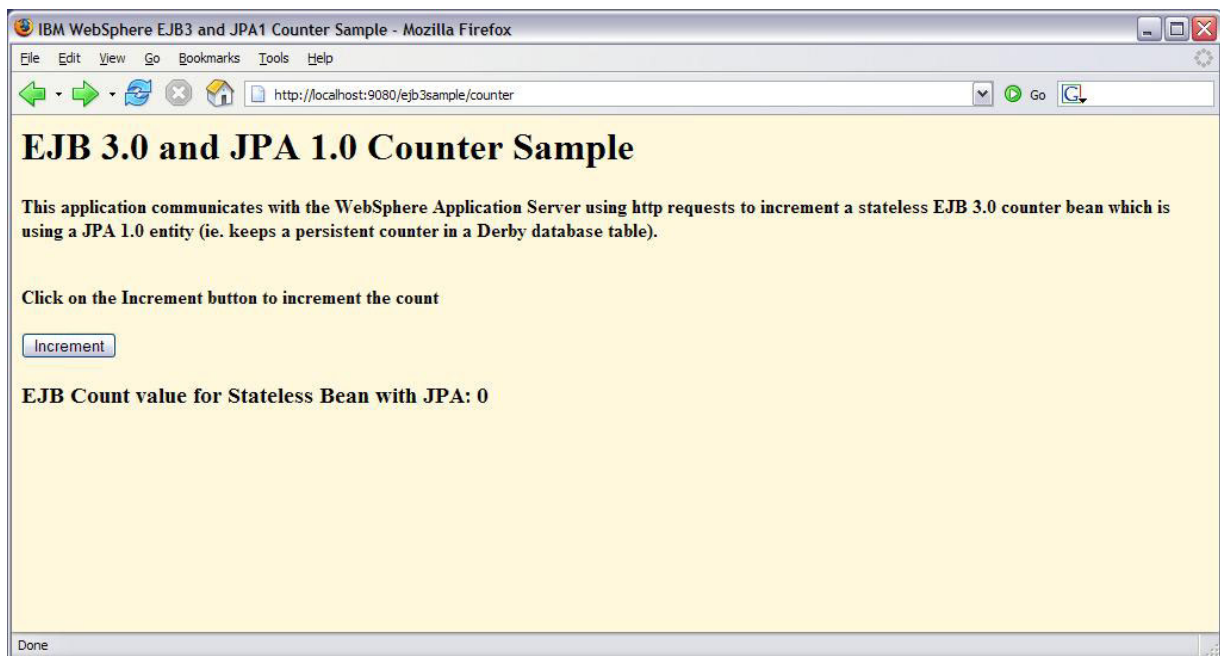
public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
String msg = null;
int ejbCount = 0;

try {
    ejbCount = statelessCounter.increment();
}
catch (RuntimeException e) {
    msg = "Error - increment() method on EJB failed!";
    e.printStackTrace();
}
msg = "EJB Count value for Stateless Bean with JPA: " + ejbCount;

// Set attributes and dispatch the JSP.
req.setAttribute("msg", msg);
RequestDispatcher rd = getServletContext().getRequestDispatcher("/EJBCount.jsp");
rd.forward(req, res);
}
}

```

15. In the Project Explorer view, expand the **EJBCounterWeb** → **Deployment Descriptor** → **Servlets**, and right-click the **EJBCount.jsp** file, and Select **Run** → **Run on Server**
16. The counter application opens in a Web browser:



Congratulations! You have completed the EJB 3.0 Counter application.

Create an EJB 3.0 application summary

This tutorial guided you through the detailed steps to create an EJB 3.0 application that contains an EJB 3.0 project and a dynamic Web project to display a counter program..

From this tutorial, you have learned how to create an EJB project and to use the EJB 3.0 annotation support for creating EJB 3.0 session beans. You also created an Enterprise Application project and a Dynamic Web project to deploy your application on a WebSphere application server.

Lessons learned

While completing the exercises, you learned how to:

- Create an EJB 3.0 project, called `EJBCounterSample`, with an EJB 3.0 stateless session bean (with both interface and implementation classes) and a JPA 1.0 entity class
- Create a Web project (`EJBCounterWeb`), and with a Java ServerPages (JSP) page and a utility Java class
- Create a Java EE 5 application (`EJBCounterSampleEAR`) with the EJB and Web projects