Session EJBs

- Implement business logic that runs on an EJB server
- Are not shared between clients - logical extension of a single client
- Do not maintain persistent state
- Two types
  - Stateless - no client specific state is maintained between client method calls
  - Stateful - can maintain conversational state
Count Stateless Session EJB

<<Interface>>
EJBHome
(from ejb)

<<Interface>>
CountHome
(from statelessSession)

<<Interface>>
Count
(from statelessSession)

increment()

countHome

create()

main()

<<Interface>>
EJBObject
(from ejb)

<<Interface>>
SessionBean
(from ejb)

CountEJB
(from statelessSession)

CountEJB()
ejbCreate()
ejbActivate()
ejbPassivate()
ejbRemove()
setSessionContext()
increment()

Copyright 2001-2002, Dan Harkey. All rights reserved. Page 7-8
Home Interface for Stateless Session EJBs

Extends the EJBHHome interface. It can also extend other interfaces but must follow the RMI-over-IIOP rules for interface definition.

Declares one method named create which has no parameters. The create method returns a reference to a remote interface for the bean and throws CreateException and RemoteException.
The interface must extend, either directly or indirectly, the interface java.rmi.Remote. This happens automatically for enterprise beans as the EnterpriseBean interface extends java.rmi.Remote.

Each method declaration must satisfy the requirements of a remote method declaration. These requirements are:

- A remote object declared as a parameter or return value must be declared as a remote interface, not the implementation class of that interface.

Each method must throw RemoteException.

Method parameters or return types must be a Java object that is serializable. This includes Java primitive types, remote Java objects, and non-remote Java objects that implement the java.io.Serializable interface.

A remote interface may also extend another non-remote interfaces, as long as all of the methods (if any) of the interface being extended satisfies the requirements of a remote method declaration.

Rules for Writing RMI-IIOP Interfaces

(continued)
// CountHome.java
package count.statelessSession;

import javax.ejb.EJBHome;
import java.rmi.RemoteException;
import javax.ejb.CreateException;

public interface CountHome extends EJBHome
{
    public Count create() throws RemoteException, CreateException;
}
// Count.java
package count.statelessSession;

import javax.ejb.EJBObject;
import java.rmi.RemoteException;

public interface Count extends EJBObject {
    public int increment(int sum) throws RemoteException;
}

May extend other classes or interfaces but bean class or one of it’s parent classes must implement the SessionBean interface.

The class must be public and must not be abstract or final.

It must provide a default, no-argument constructor and it must not have a finalize method.
The bean class or one of its parent classes must implement the ejbCreate method (note that ejbCreate returns a void) and any business methods declared in the remote interface.

The bean class can implement helper and private methods that are not exposed to clients of the enterprise bean.

---

### Count Stateless Session EJB Class

```java
// CountBean.java
package count.statelessSession;

import javax.ejb.*;
import java.rmi.RemoteException;

public class CountEJB implements SessionBean
{
    // no arg constructor
    public CountEJB() {} 

    public void ejbCreate() {} 
    public void ejbActivate() {} 
    public void ejbPassivate() {} 
    public void ejbRemove() {} 
    public void setSessionContext(SessionContext ctx ) {} 

    public int increment(int sum) 
    { 
        return ++sum; 
    } 
}
```
Count Stateless Session EJB: EJB Deployment Descriptor

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 1.1//EN' 'http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd'>
<ejb-jar>
  <description>no description</description>
  <display-name>Ejb1</display-name>
  <enterprise-beans>
    <session>
      <description>no description</description>
      <display-name>CountStateless</display-name>
      <ejb-name>CountStateless</ejb-name>
      <home>count.statelessSession.CountHome</home>
      <remote>count.statelessSession.Count</remote>
      <ejb-class>count.statelessSession.CountEJB</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Bean</transaction-type>
    </session>
  </enterprise-beans>
</ejb-jar>

Count Stateless Session EJB: Application Deployment Descriptor

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE application PUBLIC '-//Sun Microsystems, Inc.//DTD J2EE Application 1.2//EN' 'http://java.sun.com/j2ee/dtds/application_1_2.dtd'>
<application>
  <display-name>CountStateless</display-name>
  <description>Application description</description>
  <module>
    <ejb>ejb-jar27688.jar</ejb>
  </module>
</application>
**Count Stateless Session EJB:**

**J2EE RI Specific Deployment Descriptor**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<j2ee-ri-specific-information>
  <server-name/>
  <rolemapping />
  <enterprise-beans>
    <ejb>
      <ejb-name>CountStateless</ejb-name>
      <jndi-name>statelessSessionCountHome</jndi-name>
    </ejb>
  </enterprise-beans>
</j2ee-ri-specific-information>
```

**Steps Performed by an EJB Client**

- Look up the enterprise bean’s home using JNDI
- Create an instance of the bean
- Call all business methods on the bean
- Remove the bean when it is no longer needed
Steps Performed to Look Up an EJB's Home Interface Using JNDI

✓ Creating an instance of the JNDI InitialContext object
✓ Call the lookup method of the InitialContext object to find the bean's home. A string name is passed to the `lookup` method to identify the bean's home
✓ The object that is discovered using the lookup method is cast to the correct type using the `narrow` method of the PortableRemote object

Classes used by a JNDI Client

---

```
<<Interface>>
Context
(from naming)
```

---

```
InitialContext
(from naming)
```

---

```
-InitialContext()
-addToEnvironment()
-bind()
-close()
-composeName()
-createSubcontext()
-destroySubcontext()
-getEnvironment()
-getNameInNamespace()
-getNameParser()
-list()
-listBindings()
-lookup()
-lookupLink()
-rebind()
-removeFromEnvironment()
-rename()
-unbind()
```
Count Stateless Session EJB: Client (1 of 3)

// CountClient.java
package count.statelessSession;

import count.statelessSession.Count; // Count EJB Remote
import count.statelessSession.CountHome; // Count EJB Home

import java.rmi.*;
import java.util.*;
import java.io.*;
import javax.rmi.*;
import javax.naming.*;
import javax.ejb.*;

public class CountClient
{
    public static void main ( String args[] )
    {
        int sum; // current sum
        CountHome countHome;

        try
        {
            // Get the initial JNDI context
            System.out.println("Getting initial context");
            InitialContext context = new InitialContext();

            // Find the home interface and narrow to CountHome
            Object object = context.lookup("statelessSessionCountHome");
            countHome =
                (CountHome)PortableRemoteObject.narrow(object, CountHome.class);

            // Create EJB
            System.out.println("Creating EJB");
            Count count = countHome.create();

            // Set sum to initial value of 0
            System.out.println("Setting sum to 0");
            sum = 0;
        }
    }
}
// Calculate Start time
long startTime = System.currentTimeMillis();

// Increment 1000 times
System.out.println("Incrementing");
for (int i = 0 ; i < 1000 ; i++ )
{ sum = count.increment(sum);
}

// Calculate stop time; print out statistics
long stopTime = System.currentTimeMillis();
System.out.println("Avg Ping = "+((stopTime - startTime)/1000f) + " msecs");
System.out.println("Sum = " + sum);
} catch(Exception e)
{ System.err.println("Exception");
 System.err.println(e);
}