Session Beans

• As its name implies, a session bean is an interactive bean and its lifetime is during the session with a specific client. It is non-persistent.
• When a client terminates the session, the bean is not longer associated with the client and is terminated as well.
• A server site session bean represents a particular client. It responses on behalf of a client and terminates when the client session is over.
• Session beans are often designed for major and complex business logic and flow control in front of entity beans.
• A session bean may control the dialogues with entity bean business objects. They may also make requests to another session bean or to other Web components such as JSP, Servlet, or HTML pages.
• There are two session bean types: stateless session beans and stateful session beans.
Stateless Session Bean

• The stateless session bean simply defines a set of independent operations that can be performed on behalf of clients.

• A stateless session bean plays a role of controller and perform some procedural operation on behalf of client during its session.
Life Cycle of a Stateless Session Bean

• The life cycle of a stateless session bean is very simple since it does not need to keep any state and lives only during the session. Its life cycle has only two stages: not-exist and method ready for the invocation of business methods.

• The not-exist stage basically is where the bean interface and class files are located. The method stage is where the instantiated bean instance is loaded into memory.

• The EJB container may instantiate session beans when the server starts.
Life Cycle of a Stateless Session Bean (cont.)

- The EJB container manages a bean instance pool to reduce the number of component instantiations so that expenses on the creations and removals of bean instances can be significantly reduced.
- There are two type methods in a enterprise bean: the business methods and the bean life cycle methods.
- The business methods are called by clients and life cycle methods (callback) methods are called back by the EJB container when the EJB container thinks it is necessary.
- The EJB callback methods are underlined in the diagram and others are notated in the boxes.
Life Cycle of a Stateless Session Bean (cont.)

• A client requests a new session bean instance by create() method of bean home interface, and the container calls the class’s newInstance() method to create a new bean object; and then the container calls the setSessionContext() method to pass in the context environment object; it calls back the ejbCreate() method to initialize the instance.

• The last two methods are the EJB container callback methods which programmers can define. At this time this session bean is in its method ready pool stage and ready to respond client method invocation. The ejbCreate() method is only called once during any stateless session bean life cycle.
When the remove() method is called the ejbRemove() is then called next; the bean may be pulled out from the ready stage and is back to not-exist stage.
Your first Stateless Session Bean

- In this section we demonstrate a simple stateless session bean which performs a temperature conversion from a Fahrenheit temperature to its Celsius temperature. First, two interfaces (Home interface and Remote interface) are specified in F2CHome.java and F2C.java files respectively.

//F2C.java specifies remote interface for this converter //session bean. It exposes the business method fToC()

package f2c;
import javax.ejb.EJBObject;
import java.rmi.RemoteException;
import java.rmi.RemoteException;
import java.math.*;
Your first Stateless Session Bean (cont.)

```
public interface F2C extends EJBObject {
    public double fToC(double f) throws RemoteException;
}

// The file F2CHome.java specifies the home interface
// for this EJB

package f2c;
import java.io.Serializable;
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;
public interface F2CHome extends EJBHome {
    Converter create() throws RemoteException,
        CreateException;
}
```
Your first Stateless Session Bean (cont.)

- Second, we define the implementation of this stateless session bean in the F2CBean.java file.
- The fToC() method implementation is specified in this file; the declaration of this method is listed in its remote interface.
- Notice that this bean class does not have its own state property. It simply takes client inputs and performs the conversion operations, and then returns the results. It specifies the implementations of the EJB interfaces listed above.
- After it completes its service it will not remember what happened in the past.
//The file F2CBean.java specifies the EJB implementation
//class for above interfaces of this EJB component.
package f2c;
import java.rmi.RemoteException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
import java.math.*;

public class F2CBean implements SessionBean {

    public double fToC(double f) {
        double temp=(f-32)*5./9;
        return temp;
    }
}
Your first Stateless Session Bean (cont.)

// It must have a default constructor; All EJB container call back methods are also listed

    public F2CBean() {}
    public void ejbCreate() {}
    public void ejbRemove() {}
    public void ejbActivate() {}
    public void ejbPassivate() {}
    public void setSessionContext(SessionContext sc) {}
Finally, we develop a Web JSP client for this stateless session bean EJB component in the index.jsp file.

```jsp
<%@ page import="f2c.TempConv,f2c.TempConvHome,javax.ejb.*, java.rmi.RemoteException, javax.naming.*,javax.rmi.*,java.text.DecimalFormat" %>

<%!  
    private TempConv conv = null;
    public void jspInit() {
        try {
            InitialContext ic = new InitialContext();
            Object objRef =
                ic.lookup("java:comp/env/ejb/myBean");
```
Your first Stateless Session Bean (cont.)

```java
TempConvHome home =
    (TempConvHome)PortableRemoteObject.narrow(objRef,
        TempConvHome.class);
    conv = home.create();
} catch (RemoteException ex) {
    System.out.println("Couldn't create bean."+
        ex.getMessage());
} catch (CreateException ex) {
    System.out.println("Couldn't create bean."+
        ex.getMessage());
} catch (NamingException ex) {
    System.out.println("Unable to lookup home: " + "myBean "+ ex.getMessage());
}
}
Your first Stateless Session Bean (cont.)

```java
public void jspDestroy() {
    conv = null;
}
```

```xml
<html>
<head>
    <title>Temperature Converter</title>
</head>
<body bgcolor="white">
    <center>
        <h4><b>Temperature Converter</b></h4>
        <p>Enter a temperature in Fahrenheit degree:</p>
        <form method="get">
            <input type="text" name="degree" size="25">
            <br>
            <p>
                <input type="submit" name="fToC" value="Fahrenheit to Celsius">
            </p>
        </form>
    </center>
</body>
</html>
```
Your first Stateless Session Bean (cont.)

```html
<form>
  <%
      DecimalFormat twoDigits = new DecimalFormat("0.00");
      String degree = request.getParameter("degree");
      if ( degree != null && degree.length() > 0 ) {
          double d = Double.parseDouble(degree);
  %>
      if (request.getParameter("fToC") != null ) {
  %>
          <p>
          <%= degree %> in Fahrenheit degree is equivalent to
          <%= twoDigits.format(conv.fToC(d)) %> in Celsius degree.
          </p>
      %>
  %>
</center></body>
</html>
```
Your first Stateless Session Bean (cont.)

- Web clients of this application locate the home object of this session bean by the Java Naming and Directory Interface (JNDI). The *InitialContext* class is the context for performing JNDI naming operations. The `lookup()` method takes the bean's JNDI name "myBean" (deployed name) as the argument:

```java
Context initialContext = new InitialContext();
F2CHome home =
    (F2CHome)PortableRemoteObject.narrow
    (initialContext.lookup(" java:comp/env.ejb/myBean"),
    F2CHome.class);
```
Your first Stateless Session Bean (cont.)

- The PortableRemoteObject.narrow() method must be used in order to access a remote bean object via JNDI lookup.
- This method converts the RMI-IIOP compatible remote home stub into a Java object.
- For a local clients, the client and EJB bean are in the same server, the return value of the InitialContext.lookup() method is not a stub and you can directly cast it to the local home interface just like the following statement.

```java
LocalF2CHome home =
    (LocalF2CHome)initialContext.lookup("java:comp/env.ejb/myBean");
```
Your first Stateless Session Bean (cont.)

• The detail procedures of the compilation, configuration, deployment of this session bean and its Web client can be found in the section 6.7 Examples and Lab Practice. The following screen shots illustrate this stateless session bean Web application which converts 32 Fahrenheit degrees to 0 Celsius degrees.

• Client can use any Web browsers to browse the index.jsp JSP page which is the default JSP page that you don’t even need to include it as your URL; the index.jsp gets the input from clients and locates this session EJB; it then gets the required services from the bean and display the converted temperature on the page.

• This is a simplest Web application of a stateless Java enterprise session bean.
Temperature Converter

Enter a temperature in Fahrenheit degree:

32

Fahrenheit to Celsius
Temperature Converter

Enter a temperature in Fahrenheit degree:

Fahrenheit to Celsius

32 in Fahrenheit degree is equivalent to 0.00 in Celsius degree.