calculator- Using EJB 3.0 Functions

Sample of a Stateless Session Bean in EJB 3.0

This sample will demonstrate the following new features from EJB 3.0

1. Elimination of the requirement for EJB component interfaces for session beans. The required business interface for a session bean can be a plain Java interface rather than an EJBObject, EJBLocalObject, or java.rmi.Remote interface.
2. Elimination of the requirement for home interfaces for session beans.
3. Encapsulation of environmental dependencies and JNDI access through the use of annotations, dependency injection mechanisms, and simple lookup mechanisms.
4. Introduction of Java metadata annotations to be used as an alternative to deployment descriptors.

Calculator Implementation

Calculator.java: A stateless session bean that implements a simple java interface instead of an EJB component interface like EJBObject, EJBLocalObject or java.rmi.Remote. By annotating this class as a @Stateless session there is no need for a deployment descriptor to describe it separately. This class implements both a local and remote business interface, namely CalculatorLocal and CalculatorRemote.

Calculator.java

```
package org.apache.geronimo.samples.slsb.calculator;

import javax.ejb.Stateless;

@Stateless
public class Calculator implements CalculatorRemote, CalculatorLocal {

        public int sum(int add1, int add2) {
                return add1+add2;
        }

        public int multiply(int mul1, int mul2) {
                return mul1*mul2;
        }

}
```

CalculatorLocal.java: Since this is a local business interface, it is optional that the coder marks this class with a @Local annotation. A business interface which is not annotated with @Local or @Remote is assumed to be Local.

CalculatorLocal.java

```
package org.apache.geronimo.samples.slsb.calculator;

public interface CalculatorLocal {

        public int sum(int add1, int add2);

        public int multiply(int mul1, int mul2);
}
```

CalculatorRemote.java: Since this is a remote business interface, it must be annotated with the @Remote annotation.

CalculatorRemote.java

```
package org.apache.geronimo.samples.slsb.calculator;

import javax.ejb.Remote;

@Remote
public interface CalculatorRemote {

        public int sum(int add1, int add2);

        public int multiply(int mul1, int mul2);

}
```

CalculatorServlet.java: This is a servlet to process the form on the jsp page. It uses the stateless session bean Calculator to do some computation and returns the result. Note that CalculatorLocal is being annotated with the @EJB annotation. The ejb container will route every request to different bean instances. Note: a stateful session bean must be declared at the type level, whereas a stateless session bean may be declared at any level.

CalculatorServlet.java

```
package org.apache.geronimo.samples.calculator;

import java.io.IOException;
import javax.ejb.EJB;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.geronimo.samples.slsb.calculator.CalculatorLocal;

public class CalculatorServlet extends HttpServlet {

    @EJB
    private CalculatorLocal calc = null;

    public void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {

        try {
            String firstNumber = req.getParameter("firstNumber");
            String secondNumber = req.getParameter("secondNumber");
            String operation = req.getParameter("operation");

            int firstInt = (firstNumber == null) ? 0 : Integer.valueOf(firstNumber).intValue();
            int secondInt = (secondNumber == null) ? 0 : Integer.valueOf(secondNumber).intValue();

            if ( "multiply".equals(operation) ) {
                req.setAttribute("result", calc.multiply(firstInt, secondInt));
            }
            else if ( "add".equals(operation) ) {
                req.setAttribute("result", calc.sum(firstInt, secondInt));
            }

            System.out.println("Result is " + req.getAttribute("result"));

            getServletContext().getRequestDispatcher("/sample-docu.jsp").forward(req, resp);

        }
        catch ( Exception e ) {
            e.printStackTrace();
            throw new ServletException(e);
        }
    }
}
```

Deployment Plans

The structure of the deployable should look like the following:

```
|- calculator-stateless-ear-2.0-SNAPSHOT.ear
   |- META-INF
      |- application.xml
   |- calculator-stateless-ejb-2.0-SNAPSHOT.jar
   |- calculator-stateless-war-2.0-SNAPSHOT.war
```

application.xml: The JAR file is referenced to provide the functionality of this deployable. The WAR file is referenced in order to show the usage of this deployable through a web based interface. The context-root is set to be /calculator so that the URL for this application will be http://<hostname>:<port>/calculator. This is generated by maven during the build and may be found in the ear or at calculator-stateless-ear/target/application.xml. The versions will vary depending on the source code version.

application.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/application_5.xsd"
version="5">
  <description>Geronimo Sample EAR for Stateless Session</description>
  <display-name>Geronimo Sample EAR for Stateless Session</display-name>
  <module>
    <web>
      <web-uri>calculator-stateless-war-2.2-SNAPSHOT.war</web-uri>
      <context-root>/calculator-stateless</context-root>
    </web>
  </module>
  <module>
    <ejb>calculator-stateless-ejb-2.2-SNAPSHOT.jar</ejb>
  </module>
</application>
```

plan.xml: A Geronimo plan is not needed for standalone deployment of this application but is recommended to fix the module id. This example is completed by the car-maven-plugin in calculator-stateless-jetty. The dependencies are not actually needed for standalone deployment.

plan.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://geronimo.apache.org/xml/ns/j2ee/application-1.2">
  <dep:environment xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2">
    <dep:moduleId>
      <dep:groupId>org.apache.geronimo.samples</dep:groupId>
      <dep:artifactId>calculator-jetty</dep:artifactId>
      <dep:version>2.1.2</dep:version>
      <dep:type>car</dep:type>
    </dep:moduleId>
    <dep:dependencies>
      <dep:dependency>
        <dep:groupId>org.apache.geronimo.configs</dep:groupId>
        <dep:artifactId>jetty6</dep:artifactId>
        <dep:version>2.1.2</dep:version>
        <dep:type>car</dep:type>
      </dep:dependency>
      <dep:dependency>
        <dep:groupId>org.apache.geronimo.configs</dep:groupId>
        <dep:artifactId>jasper</dep:artifactId>
        <dep:version>2.1.2</dep:version>
        <dep:type>car</dep:type>
      </dep:dependency>
      <dep:dependency>
        <dep:groupId>org.apache.geronimo.configs</dep:groupId>
        <dep:artifactId>openejb</dep:artifactId>
        <dep:version>2.1.2</dep:version>
        <dep:type>car</dep:type>
      </dep:dependency>
      <dep:dependency>
        <dep:groupId>org.apache.geronimo.configs</dep:groupId>
        <dep:artifactId>openjpa</dep:artifactId>
        <dep:version>2.1.2</dep:version>
        <dep:type>car</dep:type>
      </dep:dependency>
    </dep:dependencies>
    <dep:hidden-classes/>
    <dep:non-overridable-classes/>
  </dep:environment>
</application>
```