EJB - TIMER SERVICE

Timer Service is a mechanism using which scheduled application can be build. For example, salary slip generation on 1st of every month. EJB 3.0 specification has specified @Timeout annotation which helps in programming the ejb service in a stateless or message driven bean. EJB Container calls the method which is annotated by @Timeout.

EJB Timer Service is a service provided by Ejb container which helps to create timer and to schedule callback when timer expires.

Steps to create Timer

Inject SessionContext in bean using @Resource annotation

```
@Stateless
public class TimerSessionBean {

   @Resource
   private SessionContext context;
   ...
}
```

Use SessionContext object to get TimerService and to create timer. Pass time in milliseconds and message.

```
public void createTimer(long duration) {
   context.getTimerService().createTimer(duration, "Hello World!");
}
```

Steps to Use Timer

Use @Timeout annotation to a method. Return type should be void and pass a parameter of type Timer. We are canceling the timer after first execution otherwise it will keep running after fix intervals.

```
@Timeout
public void timeOutHandler(Timer timer){
   System.out.println("timeoutHandler : " + timer.getInfo());
   timer.cancel();
}
```

Example Application

Let us create a test EJB application to test Timer Service in EJB.

Step	Description
1	Create a project with a name *EjbComponent* under a package *com.tutorialspoint.timer* as explained in the *EJB - Create Application* chapter.
2	Create *TimerSessionBean.java* and *TimerSessionBeanRemote* as explained in the *EJB - Create Application* chapter. Keep rest of the files unchanged.
3	Clean and Build the application to make sure business logic is working as per the requirements.
4	Finally, deploy the application in the form of jar file on JBoss Application Server. JBoss Application server will get started automatically if it is not started yet.
5	Now create the ejb client, a console based application in the same way as explained in

the *EJB - Create Application* chapter under topic **Create Client to access EJB**.

EJBComponent *EJBModule*

TimerSessionBean.java

```
package com.tutorialspoint.timer;

import javax.annotation.Resource;
import javax.ejb.SessionContext;
import javax.ejb.Timer;
import javax.ejb.Stateless;
import javax.ejb.Timeout;

@Stateless
public class TimerSessionBean implements TimerSessionBeanRemote {

   @Resource
   private SessionContext context;

   public void createTimer(long duration) {
      context.getTimerService().createTimer(duration, "Hello World!");
   }

   @Timeout
   public void timeOutHandler(Timer timer){
      System.out.println("timeoutHandler : " + timer.getInfo());
      timer.cancel();
   }
}
```

TimerSessionBeanRemote.java

```
package com.tutorialspoint.timer;

import javax.ejb.Remote;

@Remote
public interface TimerSessionBeanRemote {
   public void createTimer(long milliseconds);
}
```

- As soon as you deploy the EjbComponent project on JBOSS, notice the jboss log.
- JBoss has automatically created a JNDI entry for our session bean - **TimerSessionBean/remote**.
- We'll using this lookup string to get remote business object of type - **com.tutorialspoint.timer.TimerSessionBeanRemote**

JBoss Application server log output

```
...
16:30:01,401 INFO  [JndiSessionRegistrarBase] Binding the following Entries in Global
JNDI:
   TimerSessionBean/remote - EJB3.x Default Remote Business Interface
   TimerSessionBean/remote-com.tutorialspoint.timer.TimerSessionBeanRemote - EJB3.x
Remote Business Interface
16:30:02,723 INFO  [SessionSpecContainer] Starting
jboss.j2ee:jar=EjbComponent.jar,name=TimerSessionBean,service=EJB3
16:30:02,723 INFO  [EJBContainer] STARTED EJB:
com.tutorialspoint.timer.TimerSessionBeanRemote ejbName: TimerSessionBean
...
```

EJBTester *EJBClient*

jndi.properties

```
java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
java.naming.provider.url=localhost
```

- These properties are used to initialize the InitialContext object of java naming service
- InitialContext object will be used to lookup stateless session bean

EJBTester.java

```
package com.tutorialspoint.test;

import com.tutorialspoint.stateful.TimerSessionBeanRemote;
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.List;
import java.util.Properties;
import javax.naming.InitialContext;
import javax.naming.NamingException;

public class EJBTester {

   BufferedReader brConsoleReader = null;
   Properties props;
   InitialContext ctx;
   {
      props = new Properties();
      try {
         props.load(new FileInputStream("jndi.properties"));
      } catch (IOException ex) {
         ex.printStackTrace();
      }
      try {
         ctx = new InitialContext(props);
      } catch (NamingException ex) {
         ex.printStackTrace();
      }
      brConsoleReader =
      new BufferedReader(new InputStreamReader(System.in));
   }

   public static void main(String[] args) {

      EJBTester ejbTester = new EJBTester();

      ejbTester.testTimerService();
   }

   private void showGUI(){
      System.out.println("**********************");
      System.out.println("Welcome to Book Store");
      System.out.println("**********************");
      System.out.print("Options \n1. Add Book\n2. Exit \nEnter Choice: ");
   }

   private void testTimerService(){
      try {
         TimerSessionBeanRemote timerServiceBean =
(TimerSessionBeanRemote)ctx.lookup("TimerSessionBean/remote");

         System.out.println("["+(new Date()).toString()+ "]" + "timer created.");
         timerServiceBean.createTimer(2000);
```

```
        } catch (NamingException ex) {
            ex.printStackTrace();
        }
    }
}
```

EJBTester is doing the following tasks.

- Load properties from jndi.properties and initialize the InitialContext object.
- In testTimerService method, jndi lookup is done with name - "TimerSessionBean/remote" to obtain the remote business object *timerstatelessejb*.
- Then createTimer is invoked passing 2000 milliseconds as schedule time.
- EJB Container calls the timeoutHandler method after 2 seconds.

Run Client to access EJB

Locate EJBTester.java in project explorer. Right click on EJBTester class and select **run file**.

Verify the following output in Netbeans console.

```
run:
[Wed Jun 19 11:35:47 IST 2013]timer created.
BUILD SUCCESSFUL (total time: 0 seconds)
```

JBoss Application server log output

You can find the following callback entries in JBoss log

```
...
11:35:49,555 INFO  [STDOUT] timeoutHandler : Hello World!
...
```