Lab 13: EJB Timers

In this lab, you will write an EJB that uses a periodic, repeating timer.

Objectives:

- To use the EJB timer API

Part 1: Writing an EJB with a Timer Calllback

Steps:

_1. Create an EJB project named **lab13EJB** associated with **lab13EAR**. Be sure to deselect the client JAR option.

_2. Create a stateless session bean in the new project. The bean's interface and implementation class should have two methods:

```
public String startMyTimer();
public String stopMyTimer();
```

_3. Complete the bean implementation class:

a. Inject a SessionContext reference into a field.

b. In the startMyTimer() method, retrieve a TimerService from the SessionContext, then call the getTimers() method to retrieve a Collection of Timers.

If the collection's size is zero, start a timer. The timer should have an initial delay of three seconds, and then fire every five seconds. Then return a string that indicates that you started the timer successfully.

If the collection's size is not zero, return a string indicating that the timer is already running.

c. In the stopMyTimer() method, retrieve a TimerService from the SessionContext, then call the getTimers() method to retrieve a Collection of Timers.

If the collection's size is not zero, loop through the collection, calling *cancel()* on each Timer, then return a string indicating that you stopped the timer successfully.

If the collection's size is zero, return a string indicating that there's no timer to stop.

d. Write a method that returns *void* and accepts a *Timer* reference. Annotate this method with **@Timeout** so it acts as a timer callback.

In the method, simply output a message to the console indicating that the timer fired.

Part 2: Writing a Client

Steps:

_1. Create a Dynamic Web project named **lab13Web** associated with *lab13EAR*. Add the EJB project to the Web project's build path.

Copyright © Descriptor Systems 2014. All Rights Reserved. Materials may not be reproduced in any form without prior written consent of Joel Barnum.

_2. In the WebContent folder, create a simple HTML file with two forms, each of which has a submit button, one button labeled **Start**, the other **Stop**.

The "start" form's *action* should be **StartTimer** and the "stop" form's action **StopTimer**.

_3. Create a servlet named **StartTimer** that overrides *doGet*, then complete the servlet class:

a. Inject a reference to the EJB into a field.

b. In doGet, set the content type to **text/html** and retrieve a PrintWriter.

Then call the EJB's startMyTimer method and then output the returned string to the PrintWriter.

_4. Run the Web page, then press the Start button - you should see the timer messages in the console.

_5. Create another servlet named **StopTimer** that's similar to the first one except that it calls the EJB's stopMyTimer method.

_6. Run the Web page again, then press the Stop button - the timer messages in the console should stop.

Copyright © Descriptor Systems 2014. All Rights Reserved. Materials may not be reproduced in any form without prior written consent of Joel Barnum.