EJB Timers

- Timer API
- Starting a Timer

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Scheduling Events

- It's a common enterprise requirement to have "jobs" execute periodically, especially batch processing
- JEE defines the Timer API and there are third-party solutions such as Quartz and Spring Batch
- EJB provides a Timer Service API that you can use to schedule the execution of session and message-driven EJBs

---

You can read more about Quartz at http://quartz-scheduler.org.

You can read more about Spring Batch at http://projects.spring.io/spring-batch/

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The TimerService Interface

- The **javax.ejb.TImerService** type provides basic support for creating and scheduling timers

javax.ejb.TimerService
createCalendarTimer(schedule:ScheduleExpression):Timer createCalendarTimer(schedule:ScheduleExpression, cfg:TimerConfig):Timer createIntervalTimer(initial:Date, duration:long, cfg:TimerConfig):Timer createIntervalTimer(initial:long, duration:long, cfg:TimerConfig):Timer createSingleActionTimer(expiry:Date, cfg:TimerConfig):Timer createSingleActionTimer(duration:long, cfg:TimerConfig):Timer createTimer(initial:Date, info:Serializable):Timer createTimer(expiry:Date, duration:long, info:Serializable):Timer createTimer(initial:long, info:Serializable):Timer createTimer(inital:long, duration:long, info:Serializable):Timer getTimers():Collection

The durations are in milliseconds, but note that they are not guaranteed to be precise to that level because of the inherent timing uncertainties of Java code running in a multithreaded virtual machine.

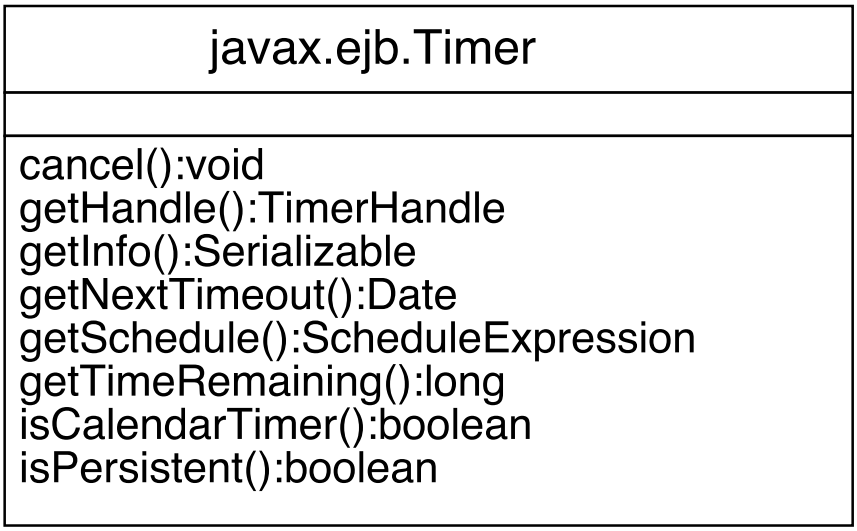
The "info" object is optional - you can pass a Serializable object and the timer API will initialize it with information about the newly created timer. If you don't care about that info, you can pass null for that parameter.

Note that EJB 3.1 adds a few more methods to this interface.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The Timer Interface

- The **javax.ejb.Timer** type represents a timer instance and provides methods to let you interact with the timer



Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The TimedObject Interface

- The **javax.ejb.TimedObject** interface is designed to be implemented by an EJB implementation class to define the **callback** method that runs when the timer fires

```
1    @Stateless
2    public class TimerBean implements TimedObject,
3          MyEJBInterface
4    {
5      public void ejbTimeout(Timer timer)
6      {
7         // runs when timer fires
8      }
9      . . .
10   }
```

This interface defines a single method, ejbTimeout().

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The @Timeout Annotation

- As an alternative to implementing the TimedObject interface, you can annotate a method to indicate it's a timer callback method

```
1    @Stateless
2    public class MyTimerBean2 implements MyEJBInterface
3    {
4      @Timeout
5      public void myCallback(Timer timer)
6      {
7         // runs when timer fires
8      }
9      . . .
10   }
```

You can also specify callback methods in the ejb-jar.xml configuration file.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Starting a Timer

- To start a timer, first retrieve a TimerService instance from the EJB context, then create the timer (automatically starts it)

```
1    @Stateless
2    public class MyTimerBean2 implements MyEJBInterface {
3      @Timeout
4      public void myCallback(Timer timer) {...}
5
6      @Resource
7      private SessionContext ctx;
8
9      public void startMyTimer() {
10       TimerService ts = ctx.getTimerService();
11       ts.createTimer(2000, 1000, null);
12     }
13   }
```

This example starts a timer that fires after an initial delay of two seconds, then every second thereafter.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Chapter Summary

In this chapter, you learned:

- How to use the EJB Timer API to schedule periodic events

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum