Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte
Campus Natal-Central
Diretoria Acadêmica de Gestão e Tecnologia da Informação
Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Utilizando o Serviço de Temporização

Prof. Fellipe Aleixo (*fellipe.aleixo@ifrn.edu.br*)

Agendamento de Tarefas

- Comum em muitos modelos de negócio
  - Geração de folha de pagamento
  - Mudar status em empréstimos em atraso
  - Fechar balanço diário de loja
  - Etc.

- *Timer service* – o serviço de temporização permite o agendamento de notificações para todos os tipos de EJB, com exceção dos *stateful*

Agendamento de Tarefas

- O agendamento pode ser:
  - Em função de datas específicas
  - Em função da passagem de um período de tempo
  - Passados certos intervalos de tempo
- Exemplos:
  - Às 10:30 AM
  - No dia 23 de maio
  - Em 30 dias
  - A cada 12 horas

Temporizadores

- Temporizadores para EJBs pode ser programáticos ou automáticos
  - Programáticos – definidos por solicitação explícita
    - Invocação de métodos da interface ***TimerService***
  - Automáticos – criados após o *deploy* de um EJB
    - Anotações ***javax.ejb.Schedule***

Expressões de Agendamento

- Temporizadores pode ser definidos de acordo com agendamentos baseados no calendário
- Um agendamento é expresso por uma expressão similar ao utilitário CRON do UNIX
- Os tipos de temporizadores se utilizam dessas expressões para definição dos agendamentos

Atributos para uma Expressão de Agendamento

Atributo	Descrição	Padrão	Exemplos
second	Valor em segundos	0	0 a 59. p.ex.: second="30"
minute	Valor em minutos	0	0 a 59. p.ex.: minute="15"
hour	Valor em horas	0	0 a 23. p.ex.: hour="13"
dayOfWeek	Dia da semana	*	0 a 7 (0 e 7 = domingo). Também pode ser um dos seguintes valores: [Sun, Mon, Tue, Wed, Thu, Fri, Sat]
dayOfMonth	Dia do mês	*	1 a 31. p.ex.: dayOfMonth="15". Expressão com: [1st, 2nd, 3rd, 4th, 5th, Last] [Sun, Mon, Tue, Wed, Thu, Fri, Sat]. p.ex: "2nd Fri" ou "Last"
month	Mês em questão	*	1 a 12. p.ex.: month="7". Ou: [Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec]
year	Ano em questão	*	Convencional. p.ex.: year="2013"

Expressão de Agendamento

- É possível a especificação de múltiplos valores em uma expressão – através de “curingas”
  - “*” representa todos os valor permitidos
  - Ex.: minute=“*”

- Especificando uma lista de valores
  - Dois ou mais valores separados por “,”
  - Ex.: dayOfWeek=“Tue, Thu”

Expressão de Agendamento

- Especificando um intervalo de valores
  - Usando o “-”para definir um intervalo inclusivo
  - Ex.: hour=“9-17”
  - Ex.: dayOfWeek=“5-1”
  - Ex.: dayOfMonth=“25-Last, 1-5”

Expressão de Agendamento

- Especificando períodos
  - Utilizando a “/” na forma “x/y”, onde “x” representa o ponto inicial e “y” o período
  - Ex.: minute=“*/10”
    - A cada 10 minutos = “0, 10, 20, 30, 40, 50”
  - Ex.: hour=“12/2”
    - A cada duas horas, iniciando ao meio dia

TEMPORIZADORES PROGRAMÁTICOS

Temporizadores Programáticos

- Quando um temporizador programático expira, o contêiner executa o método anotado com ***@Timeout***

```
@Timeout
public void timeout(Timer timer) {
   System.out.println("TimerBean: timeout occurred");
}
```

  - Tal método deve retornar “void”
  - Opcionalmente receber um ***javax.ejb.Timer***

Criando Temporizadores Programáticos

- Por meio da invocação de um dos métodos da interface ***TimerService***
- A expiração do temporizador pode ser expressa por duração ou uma data absoluta
  - Número de milissegundos após o "disparo"
  - Uma data absoluta – objeto *java.util.Date*

Criando Temporizadores Programáticos

```
long duration = 60000; // 60 milissegundos
Timer timer = timerService.createSingleActionTimer
                                (duration, new TimerConfig());

//OU

SimpleDateFormatter formatter = new SimpleDateFormatter
                                        ("MM/dd/yyyy 'at' HH:mm");
Date date = formatter.parse("05/01/2010 at 12:05");
Timer timer = timerService.createSingleActionTimer
                                (date, new TimerConfig());
```

Temporizadores Baseados em Calendário

- Definição de uma expressão por meio de um objeto ***javax.ejb.ScheduleExpression***
- Passado como parâmetro para o método ***TimerService.createCalendarTimer***

```
ScheduleExpression schedule = new ScheduleExpression();
schedule.dayOfWeek("Mon");
schedule.hour("12-17, 23");
Timer timer = timerService.createCalendarTimer(schedule);
```

TEMPORIZADORES AUTOMÁTICOS

Temporizadores Automáticos

- Criados pelo contêiner EJB quando na implantação de um bean anotado com ***@Schedule*** ou ***@Schedules***
  - Ou através do descritor de implantação
  - Um EJB pode ter múltiplos métodos de "*timeout*"
  - O atributo opcional "*persistent*" (*boolean*) permite suportar, ou não, à reinicialização do servidor

Temporizadores Automáticos

- Também pode ser especificada uma *Time Zone* específica para o agendamento
- Exemplo:

```
@Schedule(dayOfWeek="Sun", hour="0")
public void cleanupWeekData() { ... }
```

Múltiplos Agendamentos

- A anotação ***@Schedules*** permite que sejam definidos múltiplos agendamentos referentes a um dado método

```
@Schedules ({
   @Schedule(dayOfMonth="Last"),
   @Schedule(dayOfWeek="Fri", hour="23")
})
public void doPeriodicCleanup() { ... }
```

INFORMAÇÕES ADICIONAIS

Cancelando Temporizadores

- Um temporizador pode ser cancelado por meio dos seguintes eventos:
  - Quando o mesmo expira e o método de “timeout” é executado pelo contêiner
  - Quando o *bean* invoca o método ***cancel*** da interface ***Timer***

Salvando um Temporizador

- Para salvar um temporizador para uma referência futura:
  - Invocar o método ***getHandle*** e armazenar o objeto ***TimerHandle*** no banco de dados
    - O referido objeto é serializável
  - Para re-instanciar o objeto Timer, deve ser recuperado o ***TimerHandle*** e invocado o método ***getTimer***

Recuperando as Informações do Temporizador

- Informações adicionais podem ser recuperadas do temporizador

```
public long getTimeRemaining();
public java.util.Date getNextTimeout();
public java.io.Serializable getInfo();
```

- Para ter acesso a todos os *beans* com temporizadores ativos pode ser usado o método ***getTimers*** da interface ***TimerService***

Transações e Temporizadores

- Um EJB cria, normalmente, um temporizador em uma transação
  - Em caso de *rollback* – o temporizador é cancelado
- Se o *bean* cancelar um temporizador em uma transação – em caso de *rollback*, o cancelamento é desfeito
  - O tempo é restaurado ao do início da transação

EXEMPLO COMPLETO

O Exemplo

- ***TimerSessionBean*** – um *bean* de sessão *singleton* com os dois tipos de temporizadores
  - Os métodos *setTimer* e *@Timeout* são utilizados para definir o temporizador programático
  - Também há a definição de um temporizador automático com a anotação *@Schedule*

Ex.Temporizador Programático

```
...
   public void setTimer(long intervalDuration) {
      logger.info("Setting a programmatic timeout for " +
            intervalDuration + " milliseconds from now.");
      Timer timer = timerService.createTimer(intervalDuration,
            "Created new programmatic timer");
   }

   @Timeout
   public void programmaticTimeout(Timer timer) {
      this.setLastProgrammaticTimeout(new Date());
      logger.info("Programmatic timeout occurred.");
   }
...
```

Ex. Temporizador Automático

```
...
   @Schedule(minute="*/1", hour="*")
   public void automaticTimeout() {
      this.setLastAutomaticTimeout(new Date());
      logger.info("Automatic timeout occured");
   }
...
```

O Código Completo...

```
package timersession.ejb;
import java.util.Date;
import java.util.logging.Logger;
import javax.annotation.Resource;
import javax.ejb.Schedule;
import javax.ejb.Stateless;
import javax.ejb.Timeout;
import javax.ejb.Timer;
import javax.ejb.TimerService;

@Singleton
public class TimerSessionBean {
    @Resource
    TimerService timerService;
```

```
private Date lastProgrammaticTimeout;
private Date lastAutomaticTimeout;

private Logger logger = Logger.getLogger
     ("com.sun.tutorial.javaee.ejb.timersession.TimerSessionBean");

public void setTimer(long intervalDuration) {
   logger.info("Setting a programmatic timeout for "
         + intervalDuration + " milliseconds from now.");
   Timer timer = timerService.createTimer(intervalDuration,
         "Created new programmatic timer");
}
@Timeout
public void programmaticTimeout(Timer timer) {
   this.setLastProgrammaticTimeout(new Date());
   logger.info("Programmatic timeout occurred.");
}
```

```
@Schedule(minute="*/1", hour="*")
public void automaticTimeout() {
    this.setLastAutomaticTimeout(new Date());
    logger.info("Automatic timeout occured");
}

public String getLastProgrammaticTimeout() {
    if (lastProgrammaticTimeout != null) {
        return lastProgrammaticTimeout.toString();
    } else {
        return "never";
    }

}
```

```
    public void setLastProgrammaticTimeout(Date lastTimeout) {
        this.lastProgrammaticTimeout = lastTimeout;
    }

    public String getLastAutomaticTimeout() {
        if (lastAutomaticTimeout != null) {
            return lastAutomaticTimeout.toString();
        } else {
            return "never";
        }
    }

    public void setLastAutomaticTimeout(Date lastAutomaticTimeout)
    {
        this.lastAutomaticTimeout = lastAutomaticTimeout;
    }
}
```