EJB2

António Sardinha

May 2020

1 Ciclo de vida de um EJB

- ***Stateful Session Bean***
  - O cliente obtém a referência para o Bean (*Create*)
  - O Bean é removido da memória para o disco (*Passive*)
  - Único método invocado pelo utilizador (*Remove*)
- ***Stateless Session Bean***
  O EJB *container* cria uma pool de *stateless beans*
  - Para cada instância:
    1. *Dependency injection, if any*
    2. *PostConstruct callback, if any*
- ***Singleton Session Bean***
  - Se o *Singleton bean* tem a anotação *@Startup*, a única instância do Bean é iniciada pelo *container* quando é feito o *deploy* da aplicação.
  - Possui os mesmos estados que um *Stateless bean.*
- ***Message Driven Bean***
  - O EJB *container* criar uma pool de MDBs.

2 *Timers*

- O serviço de Timer do contentor de JEBs permite:
  - Programar no tempo notificações para todos os tipos de EJB com **exceção de *Stateful*** *Session Beans*
- São possíveis notificações que ocorrem:
  - De acordo com um determinado calendário
  - Num tempo específico (e.g.,12 de Setembro, 9:00 a.m.)

  - Após um certo período de tempo (e.g., dentro de 4 dias)
  - Em intervalos de tempo (e.g., cada 3 minutos)

- Os *Timers* podem ser programados ou automáticos
- *Timers* programados são criados por invocação de um dos métodos da interdace *TimerService*
  - Quando expira o tempo de um *timer* programado é executado o método anotado com @Timeout
- Criar um Timer programado.(1)
- Criar um Timer programado.(2)
- Criar um Timer programado.(3) Por omissão os *Timers* são persistentes.
  - Se o servidor falha, o timer fica guardado em memória persistente e reativado quando o servidor recupera.
  - Se o timer expirar enquanto o servidor estiver inativo, o método @Timeout é invocado, após o restart do servidor.

```
//Cria   o de um timer programado (1)
long duration = 60000;
Timer timer = timerService.createSingleActionTimer
(duration, new TimerConfig());
SimpleDateFormatter formatter = new SimpleDateFormatter
("MM/dd/yyyy     at     HH:mm");
Date date = formatter.parse( 5 /05/2020 at 13:00");

Timer timer = timerService.createSingleActionTimer(date,
new TimerConfig());


//Cria   o de um timer programado (2)
ScheduleExpression schedule = new ScheduleExpression();
schedule.dayOfWeek("Mon");
schedule.hour("12-17, 23");
Timer timer = timerService.createCalendarTimer(schedule);
```

3 EJB Timers

- Criar um Timer automático (1)
  - Timers automáticos são criados após o *deploy* de um *Bean* que contém um método anotado com java.ejb.Schedule ou java.ejb.Schedules

  - Um *bean* pode ter vários timers automáticos (os timers programados são únicos por bean)
  - Um método anotado com @Schedule funciona como um método de *timeout* para o calendário especificado nos atributos da anotação.

- Criar um Timer automático (2)
  - Usar *Schedules* para especificar vários calendários para o mesmo método.

- Especificação de Intervalos
  - Numa expressão da forma x/y, x o ponto de partida e y o intervalo.
  - O *wildcard* (*) pode ser usado na posição x e equivale a x = 0.

```
//Criacao de um Timer autom tico
@Schedules ({
@Schedule(dayOfMonth="Last"),
@Schedule(dayOfWeek="Fri", hour="23")
})
public void doPeriodicCleanup() { ... }

//Criacao de um Timer autom tico
minute="*/10    ; // significa, todos os 10 minutos come ando  s  0 horas.
hour="12/2    ; // significa todas as duas horas a come ar ao meio-dia.
```