→ Enterprise java Bens

→ Session beans

A – Aplicação cliente acede a um session bean

1 – Siga o tutorial cujo link se apresenta abaixo para implementar um exemplo que usa uma aplicação cliente para aceder a um Stateless Session bean.

Este tutorial cria uma biblioteca (java library) que irá conter a interface remota para o EJB que irá ser criado. Qualquer cliente que pretenda aceder ao Bean, apenas necessita de adicionar a biblioteca ao seu projeto. Para criarmos o Bean precisamos de incluí-lo numa *Enterprise application*, para depois ser implantado *(fazer o deploy)* no servidor.

Finalmente é necessário construir a aplicação cliente, à qual se adiciona a biblioteca criada inicialmente, e a partir desse cliente podemos aceder ao Bean.

Nota: - *Ao criar uma Enterprise Application ou uma Enterprise Application Client, no tutorial é selecionado o servidor Glassfish 3.1 e o Java EE 6. Se na sua máquina estiver instalado o Glassfish 4 e o Java EE 7, deve usá-los.*

a) Implemente e estude o tutorial: https://netbeans.org/kb/docs/javaee/entappclient.html

b) Acrescente ao bean anterior outros métodos, nomeadamente métodos que recebam parâmetros e que devolvam outros tipos de resultado.
Nota: Antes de executar o cliente onde acede aos novos métodos, aceda ao "tab" services, e em Servers / GlassFish faça Undeploy da sua Enterprise Application, EntAppEJB. No tab projects volte a fazer o Deploy e de seguida pode executar a Enterprise Applicaion Client.

c) Adicione um outro bean ao projeto anterior com funcionalidade à escolha e aceda aos seus métodos a partir da aplicação cliente.

2 - Por analogia com o exemplo anterior criar um novo projeto em que uma aplicação cliente é usada para aceder a um Stateful Session Bean com funcionalidades à escolha.

a) No mínimo criar um método de inicialização, os getters e setters e testar o Bean.

b) Verifique que o estado do bean permanece entre invocações diferentes feitas pelo mesmo cliente.

c) O que acontecia com o stateless bean?

B – Aplicação web acede a um session bean

4 – Usar o tutorial https://netbeans.org/kb/docs/web/quickstart-webapps.html para criar uma aplicação web que invoca métodos de um java Bean numa página JSP.

Nota: se ao criar o projeto do tipo web application na pasta Web Pages tiver um ficheiro index.html em vez de um index.jsp, apenas terá de criar um ficheiro index.jsp e eliminar o existente. Para criar o ficheiro index.jsp, deve clicar com o botão direito na pasta do projeto e selecionar New | JSP. Depois é seguir o tutorial novamente.

5 - Ao exemplo anterior pretende-se adicionar um Singleton Session Bean que conte o número de acessos à página web.

a) Construir o Session Bean usando o código abaixo

```
@Singleton
public class CounterBean {
   private int hits = 1;
   // Increment and return the number of hits
   public int getHits() {
      return hits++;
  }
}
```

b) Modificar a página response.jsp para invocar o método getHits. Para isso terá que indicar qual o bean que vai usar (item "use bean") a aceder à propriedade *hits (item "Get Bean Property").*

c) Testar o Bean fazendo *reload* da página e executando novamente a aplicação. Quando é que o contador volta a "1"? No item "use bean", explore os diferentes tipos de *scope* possíveis.

C – Timers

6 – Timer Programado

Considere a aplicação do exercício 1, aplicação cliente que acede a um session stateless ejb. Para adicionar à aplicação um Timer programado adicione ao stateless bean da aplicação, após o cabeçalho da classe, o seguinte código:

```
@Resource
TimerService timerService;
```

No interior da classe (Bean) definir um método **local** anotado com @Timeout. Por exemplo,

```
@Timeout
   public void metodoTimeout(Timer timer) {
      System.out.println ("Ocorreu um timeout" + new Date());
   }
```

- Para testar o método criar um novo método remoto do Bean, durationTimer, onde irá criar um objeto do tipo Timer. Quando esse Timer expirar será executado o método anotado com @Timeout. O método remoto durationTimer (ou outro nome à sua escolha) deve receber um parâmetro do tipo long (e.g., intervalDuration) que corresponde ao intervalo de tempo que irá decorrer até o Timer expirar.

No método durationTimer instancie o timer com a instrução:

```
Timer timer = timerService.createTimer(intervalDuration, "Created new timer");
```

Finalmente, deve modificar a aplicação cliente de forma a invocar o método que cria o timer.

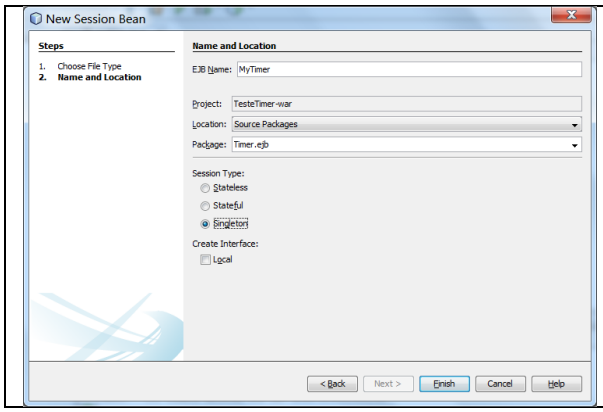
- Testar o exemplo pedindo ao utilizador vários valores para a duração do timeout.

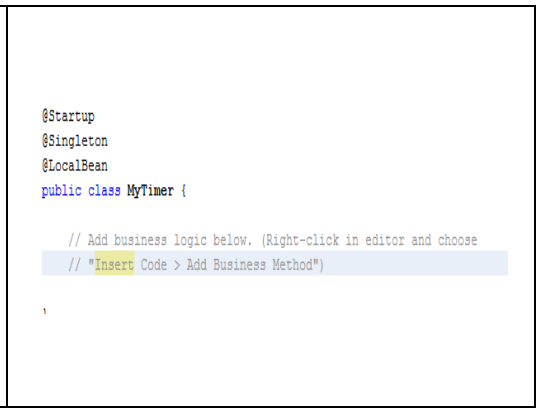
Nota: É possível ver o output do método "metodoTimeout" na consola do servidor GlassFish.

7 – Timer Automático

a) Criar uma "enterprise application" (nome = TesteTimer) com um module "web Application". Isto é, criar um novo projecto, na categoria javaEE seleccionar EnterpriseApplication, dar o nome ao projecto, após Next, seleccionar o servidor e a versão do javaEE e seleccionar a opção "Create Web Application Module".

b) No módulo TesteTimer-war, Source Packages, criar um Session Singleton Bean com o nome MyTimer. Na classe do Bean, inserir a anotação @Startup e o código do método automaticTimeout() listado abaixo.



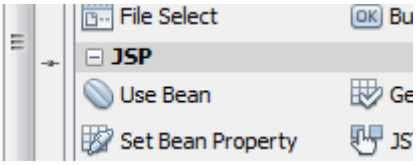


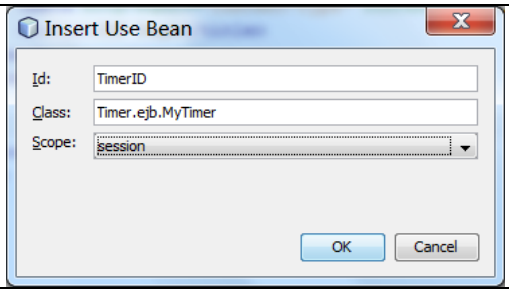
```
@Startup
@Singleton
@LocalBean
public class MyTimer {

    // Add business logic below. (Right-click in editor and choose
    // "Insert Code > Add Business Method")

}
```

```
@Schedule(minute = "*/1", hour = "*")
   public void automaticTimeout() {
      System.out.println ("Automatic timeout occurred " + new Date());
   }
```

c) Após corrigir os erros, editar o ficheiro index.jsp em TesteTimer-war, Web Pages. Se o ficheiro não existe, comece por criá-lo. Para editar o ficheiro index.jsp, no menu Window | IDE Tools selecione a Palette caso não tenha essa janela aberta.

- Na linha a seguir a "<h1>Hello World!</h1>" inserir o item UseBean da secção JSP da Palette. Dar um nome ao item, indicar o nome do Bean que construiu e escolher o scope "session".





d) Executar a aplicação TesteTimer e observar o que acontece no Browser e na consola do servidor GlassFish.

e) Pode terminar o Timer: - Aceder a "Services" "Server" GlassFish Server" "Applications" em TesteTimer fazer undeploy.

f) Testar outros calendários (Schedules).

g) Na página index.jsp, comentar a linha "use Bean" e perceber o que acontece!