


Jakarta EE Web Total


Componentes EJB: Session Beans





Tópicos Abordados




- Enterprise Java Beans
- Session Beans
 - Stateful
 - Stateless
 - Singleton
- Elementos de um session bean
 - Classe do bean
 - Interfaces locais e remotas
- Criando session beans
- Referenciando session beans
- Ciclo de vida



Enterprise Java Beans



- Chamados também de EJBs
- Componentes da plataforma Java EE que implementam a lógica de negócio da aplicação
- O uso de EJBs favorece a escalabilidade
 - Os EJBs podem ser distribuídos em diferentes servidores
- O container fornece diversos serviços aos EJBs, como segurança e transações
 - O foco do desenvolvedor pode ficar 100% na implementação da lógica de negócio



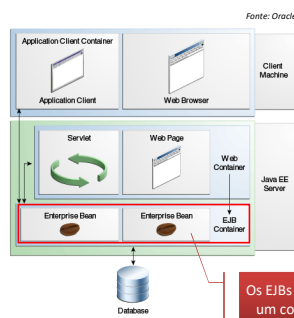
Enterprise Java Beans



- Os EJBs foram introduzidos na plataforma J2EE
- Tinham uma série de limitações
- Configuração complicada
- Muitos artefatos
- Nas últimas versões do Java EE os EJBs sofreram uma melhora significativa



Enterprise Java Beans



Os EJBs são executados por um container específico



Tipos de EJBs

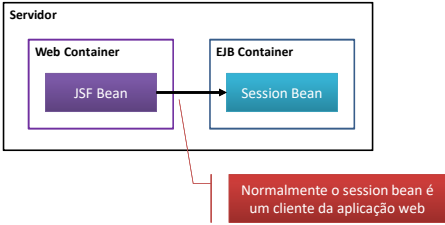


- Existem 2 tipos de EJBs
 - Session Beans
 - Stateful
 - Stateless
 - Singleton
 - Message-Driven Beans



Session Beans

- Um session bean possui métodos que podem ser invocados por um cliente

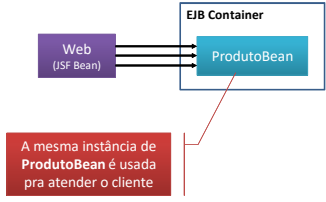


The diagram shows a 'Servidor' box containing a 'Web Container' and an 'EJB Container'. Inside the 'Web Container' is a 'JSF Bean'. Inside the 'EJB Container' is a 'Session Bean'. An arrow points from the 'JSF Bean' to the 'Session Bean'. A red callout box points to the 'Session Bean' with the text: 'Normalmente o session bean é um cliente da aplicação web'.

Normalmente o session bean é um cliente da aplicação web

Stateful Session Bean

- Está atrelado a um cliente específico



The diagram shows a 'Web (JSF Bean)' box connected by three parallel lines to a 'ProdutoBean' box inside an 'EJB Container'. A red callout box points to the 'ProdutoBean' with the text: 'A mesma instância de ProdutoBean é usada pra atender o cliente'.

A mesma instância de **ProdutoBean** é usada pra atender o cliente

Stateful Session Bean: Características

- Mantém estado conversacional
 - O cliente interage com o mesmo bean
- Como o bean não é compartilhado, é possível armazenar dados nos seus atributos
- É preciso que exista um objeto pra cada cliente acessando a aplicação
 - Isso pode causar problemas em aplicações que demandam uma grande quantidade de acessos simultâneos

Stateless Session Bean

Softblue

- Não está atrelado a um cliente específico
 - Pode atender vários clientes

Qualquer instância de **ProcessarBean** pode ser usada pra atender o cliente

Stateless Session Bean: Características

Softblue

- Não mantém estado conversacional
 - Diferentes instâncias do bean podem ser designadas pra atender os clientes
- Como o bean é compartilhado, não é possível armazenar dados nos seus atributos
- Poucas instâncias de beans podem atender muitos clientes
 - Essa é uma característica interessante em cenários onde a escalabilidade é desejada

Singleton Session Bean

Softblue

- Existe apenas uma instância do bean
 - Ela é compartilhada entre todos os clientes

A mesma instância de **CacheBean** é usada pra atender todos os clientes

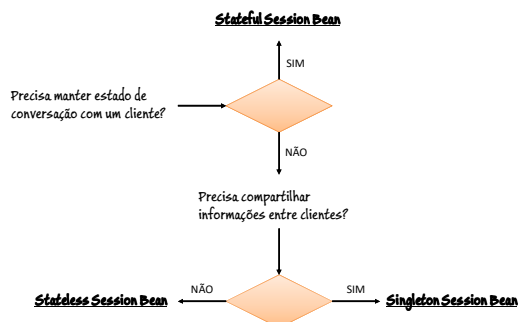
Singleton Session Bean: Características



- A instância é única durante todo o tempo de vida da aplicação
 - Ela é compartilhada entre todos os clientes, que podem acessá-la simultaneamente



Qual tipo de Session Bean escolher?



Elementos de um Session Bean



- Um EJB é composto por:
 - Classe do bean
 - Obrigatória
 - *Business interface* (interface do bean)
 - Opcional
- Se um EJB possuir uma *business interface*, ela pode ser:
 - Local
 - O cliente do EJB e o EJB precisam estar executando na mesma JVM
 - Remota
 - O cliente do EJB e o EJB podem estar executando em JVMs diferentes



Elementos de um Bean: Cenário #1

Softblue

- Bean com interface remota

```
graph LR
    subgraph JVM_A [JVM A]
        JSF_Bean_A[JSF Bean]
        Service["@Remote Service"]
        ServiceBean["<< class >> ServiceBean"]
        JSF_Bean_A --> Service
        ServiceBean -.-> Service
    end
    subgraph JVM_B [JVM B]
        JSF_Bean_B[JSF Bean]
    end
    JSF_Bean_B --> Service
```

Elementos de um Bean: Cenário #2

Softblue

- Bean com interface local

```
graph LR
    subgraph JVM_A [JVM A]
        JSF_Bean_A[JSF Bean]
        Service["@Local Service"]
        ServiceBean["<< class >> ServiceBean"]
        JSF_Bean_A --> Service
        ServiceBean -.-> Service
    end
    subgraph JVM_B [JVM B]
        JSF_Bean_B[JSF Bean]
    end
    JSF_Bean_B --> Service
```

Elementos de um Bean: Cenário #3

Softblue

- Bean sem interface

```
graph LR
    subgraph JVM_A [JVM A]
        JSF_Bean_A[JSF Bean]
        ServiceBean["<< class >> ServiceBean"]
        JSF_Bean_A --> ServiceBean
    end
    subgraph JVM_B [JVM B]
        JSF_Bean_B[JSF Bean]
    end
    JSF_Bean_B --> ServiceBean
```

É como se tivesse uma interface local

Criando um Session Bean: Classe



- A classe do EJB é uma classe comum, anotada com o tipo do bean (`@Stateless`, `@Stateful` ou `@Singleton`)

Stateless Session Bean sem interface

```
@Stateless
public class TemperatureConverterBean {
    public double celsiusToFahrenheit(double tc) {
        // lógica de negócio
    }
}
```



Criando um Session Bean: Interface



- A interface do EJB é anotada com `@Local` ou `@Remote`

Interface local

```
@Local
public interface TemperatureConverter {
    public double celsiusToFahrenheit(double tc);
}
```

```
@Stateless
public class TemperatureConverterBean implements TemperatureConverter {
    public double celsiusToFahrenheit(double tc) {
        // lógica de negócio
    }
}
```



Criando um Session Bean: Interface



- Outra opção é anotar apenas a classe

```
public interface TemperatureConverter {
    public double celsiusToFahrenheit(double tc);
}
```

```
@Stateless
@Remote(TemperatureConverter.class)
public class TemperatureConverterBean implements TemperatureConverter {
    public double celsiusToFahrenheit(double tc) {
        // lógica de negócio
    }
}
```



Referenciando um Session Bean

- A referência a um EJB pode ser injetada via CDI

JSF Bean

```
@Named("form")
@RequestScoped
public class FormBean implements Serializable {
    @EJB
    private TemperatureConverter temperatureConverter;
}
```

Interface

JSF Bean

```
@EJB
private TemperatureConverterBean temperatureConverter;
```

Classe do bean

Chamadas Assíncronas

- Os Session Beans suportam métodos assíncronos
 - O retorno ocorre antes mesmo do método terminar de executar
- A annotation `@Asynchronous` é utilizada
- O método deve retornar
 - `void`
 - `Future<?>`

Chamadas Assíncronas

```
@Asynchronous
public void m1() {
    // lógica de negócio
}
```

O objeto `Future<?>` possui métodos para checar se a execução terminou e pegar o resultado

```
@Asynchronous
public Future<String> m1() {
    // lógica de negócio
    return new AsyncResult<String>("ABC");
}
```