Capítulo 5

Desenvolvimento de componentes

Agora que sabemos o que é um componente, precisamos entender como implementá-lo. Seguiremos com a visão do Java, por isso, focaremos em componentes back-end implementados utilizando o Enterprise Java Beans (EJB) em sua versão 3.0. O EJB é um framework que tem como objetivo simplificar o processo de desenvolvimento de componentes para aplicações distribuídas construídas com a linguagem de programação Java (GONÇALVES, 2007).

Ao utilizar o EJB, o desenvolvedor pode manter seu foco na lógica de negócio, ou na funcionalidade abordada por seu componente, em vez de distribuí-lo entre cada parte do sistema. Vamos entender como funciona a elaboração de um componente EJB.

1 Elaboração de um componente

Para utilizar o EJB, é necessário ter um servidor de aplicações web que suporte o seu uso. A funcionalidade que o servidor possui para receber um arquivo componente EJB é chamada de container EJB, responsável por garantir que o ciclo de vida de um componente seja considerado de maneira íntegra. Alguns servidores que possuem essa característica são: Oracle Glassfish, Redhat WildFly/JBoss, IBM Websphere, entre outros (GONÇALVES, 2007).

São características importantes de um container EJB:

- **Gerenciamento de transação:** manter o controle entre os componentes de origem e destino.
- **Segurança:** garantir a consistência da informação entre as transações realizadas.
- **Controle de concorrência:** oferecer suporte a chamadas assíncronas.
- **Serviço de rede:** aceitar os principais protocolos de comunicação via rede.
- **Gerenciamento de recursos:** otimizar o uso de recursos compartilhados pelos componentes.
- **Persistência dos dados:** realizar conexão com o banco de dados de maneira segura.
- **Serviço de mensageria:** gerenciar a fila de mensagens entre os componentes.

Para construir o EJB, precisamos considerar a funcionalidade do negócio que ele irá abordar e em seguida identificar qual tipo de EJB é o mais adequado para a situação. Existem três tipos de EJB:

- **Session bean:** utilizado para implementar as regras de negócios do componente.
- **Message-driven bean**: feito para a troca de mensagens entre os componentes. Hoje existem ferramentas focadas nessa troca de mensagens.
- **Entity bean (até a versão EJB 2.x):** representa as tabelas do banco de dados (foi substituído pelo JPA).

Agora que já sabemos os tipos existentes de EJB, vamos entender como implementá-los.

2 Aplicação em solução baseada na web

Suponha que o contexto é uma aplicação para uma biblioteca on-line de uma universidade. A partir disso, precisamos implementar o componente para empréstimos de livros aos alunos da instituição. Vamos construir um session bean.

2.1 Session bean

O session bean é um componente que mantém a lógica de negócio da aplicação. Através dele, é possível fazer com que outros componentes utilizem dessa lógica de negócio para validar ou acessar o sistema a que ele pertence (SAKURAI, 2020). Podemos construir um session bean de duas maneiras: stateless ou stateful, que basicamente significam manter ou não o estado desse componente.

Mas, antes de pensarmos no estado do componente, é importante entendermos como funciona o ciclo de vida do EJB, que é controlado pelo container existente no servidor de aplicação. Dessa forma, criar um componente utilizando a notação do Java, através da palavra reservada

new, conforme apresentado a seguir, é proibido. Essa ação não funcionará corretamente.

```
ComponenteEJB ejb = new ComponenteEJB(); // Má prática
```

Conseguimos compilar e executar o código anterior tranquilamente, sem problemas. Porém, como ele não foi criado pelo container do EJB, não será possível realizar seu monitoramento de maneira adequada, prejudicando assim o funcionamento do sistema. Para isso, devemos utilizar o servidor web na criação da instância de um componente EJB no sistema, conforme o código:

```
InitialContext ctx = new InitialContext();
ComponenteEJB ejb = (ComponenteEJB) ctx.
lookup("nomeDoComponente");
```

Agora, sim, o EJB foi instanciado dentro do container, ganhando todos os benefícios mencionados anteriormente.

2.1.1 Stateless session bean

Esse tipo de componente EJB, como o nome já diz, tem a duração de apenas uma chamada simples de seu método. Dessa maneira, após a execução e obtenção do resultado, já pode ser removido do servidor. O passo a passo de um componente stateless no sistema é:

- o container cria o componente;
- executa o método chamado;
- o objeto recuperado é enviado a quem fez a requisição;

- o container remove o componente da memória.

Esse tipo de componente apresenta um problema: a cada requisição realizada, uma nova instância do componente é criada no servidor, por isso geralmente trabalha-se com um pool de componentes no container EJB. Assim, é possível otimizar o processo e o consumo da memória do servidor.

O stateless session bean é formado por uma interface que pode ser remota, local ou ambas. Quando uma aplicação cliente do componente está em outro servidor web ou é uma aplicação desktop, deve-se utilizar a interface Remote (SAKURAI, 2020). Ela faz parte do pacote javax.ejb. Vamos criar uma interface *ServicoRemote* para nossa biblioteca:

```
import javax.ejb.Remote;

@Remote
public interface ServicoRemote {
    void emprestaLivro(Livro livro);
}
```

Vamos construir também a opção local, que é utilizada para fazer uma chamada EJB que está no mesmo servidor web da aplicação cliente. Será criada a interface *ServicoLocal*, a partir da implementação:

```
import java.util.List;
import javax.ejb.Local;

@Local
public interface ServicoLocal {
    List<Livro> buscaLivros(Categoria categoria) throws
Exception;
}
```

Perceba que podemos chamar tanto o serviço remoto quanto o local para esse EJB. Sendo assim, é importante sempre analisar a situação antes de decidir qual tipo será implementado. Verifique por onde esse componente será acessado, qual a necessidade de distribuí-lo. Todas essas questões irão definir a estratégia de implementação mais adequada. Vale lembrar que podemos ter ambas as interfaces em um mesmo componente.

Agora podemos criar uma classe que implemente a interface. Essa classe será a responsável por implementar efetivamente a lógica do componente. Ela precisa ter a anotação *stateless* também pertencente ao pacote javax.ejb. No exemplo, utilizaremos apenas a interface remota para ilustrar:

```
import javax.ejb.Stateless;

@Stateless
public class ServicoBean implements ServicoRemote {
    public void emprestaLivro(Livro livro) {
        // Código para emprestar o livro
    }
}
```

Pronto, nosso componente já pode ser implantado no servidor de produção.

2.1.2 Stateful session bean

Ao contrário do componente anterior, um componente stateful pode continuar sua vida no container EJB até que a sessão do usuário seja encerrada. Esse tipo de componente também possui a capacidade de manter o estado anterior de seus atributos, uma vez que ele se mantém na memória após sua execução (SAKURAI, 2020).

Porém, não apenas de vantagens é feito esse tipo de componente. Se o cliente acessa múltiplas vezes a aplicação, uma quantidade grande de objetos do mesmo componente pode ser criada e associada a esse cliente, o que pode gerar problemas de desempenho e, principalmente, de memória no servidor.

Mas, com todos os objetos criados, é possível navegar entre os estados para verificar o que aconteceu anteriormente com o objeto. Basicamente, pode-se utilizar um *control + z* nas informações armazenadas através daquele componente. Podemos ter os dois tipos de serviços para um componente stateful. A única coisa que muda é a remoção da anotação *stateless* e a inclusão do stateful, conforme apresentado no código:

```
import javax.ejb.Stateful;

@Stateful
public class ServicoBean implements ServicoRemote {
    public void emprestaLivro(Livro livro) {
        // Código para emprestar o livro
    }
}
```

As mesmas regras do pool de componentes podem ser aplicadas para os componentes stateful. Mas lembre-se de que é preciso saber como utilizar esse tipo de componente, pois a memória do servidor pode acabar com o espaço disponível, levando o sistema a um colapso.

Considerações finais

Neste capítulo, apresentamos como criar um componente utilizando o EJB. Compreendemos que os componentes podem possuir o controle ou não de um estado anterior através das anotações *stateless* e *stateful*.

Também abordamos que existem dois tipos de interface: as remotas, para atender aos sistemas de diferentes servidores, ou as locais, para atender aos sistemas de um mesmo servidor. Com isso, agora você está apto a construir o seu componente.

Referências

GONÇALVES, Edson. **Desenvolvendo aplicações web com JSP, Servlets, JavaServer Faces, Hibernate, EJB 3 Persistence e Ajax**. Rio de Janeiro: Ciência Moderna, 2007.

SAKURAI, Rafael Guimarães. **Desenvolvimento distribuído com Java EE**. [*S. l.*:]GitBooks, 2020. Disponível em: https://rafaelsakurai.gitbooks.io/desenvolvimento-distribuido/content/. Acesso em: 17 dez. 2020.