Transcrição

Aprendemos o funcionamento das requisições e seus métodos diferentes, isto é, GET e POST e as formas de escrever estas especificações no código da Servlet. A próxima meta é completar a aplicação, por enquanto não realizamos um cadastro de fato, afinal não fazemos nada com o registro da empresa. A ideia é criarmos um modelo para as empresas utilizando uma classe que as representará.

Dentro de `br.com.alura.gerenciador.servlet`, clicaremos a nova classe `Empresa`. Primeiramente, a empresa deve ter um `nome`. A ideia é simularmos o mais próximo o possível uma aplicação real, portanto uma classe em nosso modelo deve possui a identificação(`id`) do banco de dados, ou seja, sua chave primária (*primary key*).

```
package br.com.alura.gerenciador.servlet;

public class Empresa{

    private int id;
    private String nome;

}
```

Em seguida, geraremos de forma automática os métodos *getters* e *setters*, o que nem sempre é uma boa prática, mas neste caso atenderá as necessidades do projeto. Clicaremos com o botão direito no interior da classe `Empresa` e selecionaremos as opções "Source > Generate Getters and Setters". Na nova caixa de diálogo selecionaremos os atributos `id` e `nome`.

```
package br.com.alura.gerenciador.servlet;

public class Empresa{

    private Integer id;
    private String nome;

    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
}
```

Em `NovaEmpresaServelt`, a partir dos parâmetros construídos, criaremos uma empresa (`Empresa empresa = new Empresa()`) e utilizaremos o `setNome()` para realizar uma chamada em `nomeEmpresa`.

```
@WebServlet("/novaEmpresa")
public class NovaEmpresaServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse respo
            throws ServletException, IOException {
        System.out.println("Cadastrando nova empresa");

        String nomeEmpresa = request.getParameter("nome");
        Empresa empresa = new Empresa();
        empresa.setNome(nomeEmpresa);

        PrintWriter out = response.getWriter();
        out.println("<html><body>Empresa " + nomeEmpresa + " cadastrada com suc
    }

}
```

Temos um objeto e encapsulamos `nome`, e agora simularemos um acesso ao banco de dados, que é feito por meio de uma classe especializada. Não usaremos um banco de dados verdadeiro, afinal esse seria um movimento mais complexo que será abordado em outros cursos. O que faremos é simular um banco utilizando a classe `Banco`. A classe não será compilada, pois ela não existe no projeto, portanto a criaremos.

```
    String nomeEmpresa = request.getParameter("nome");
        Empresa empresa = new Empresa();
        empresa.setNome(nomeEmpresa);

        Banco banco = new Banco();

        PrintWriter out = response.getWriter();
        out.println("<html><body>Empresa " + nomeEmpresa + " cadastrada com suc
    }

}
```

A partir da classe `Banco` adicionaremos a `empresa` no banco. Utilizaremos o método `adiciona()`, que precisará ser criado.

```
    String nomeEmpresa = request.getParameter("nome");
        Empresa empresa = new Empresa();
        empresa.setNome(nomeEmpresa);

        Banco banco = new Banco();
        banco.adiciona(empresa);

        PrintWriter out = response.getWriter();
        out.println("<html><body>Empresa " + nomeEmpresa + " cadastrada com suc
    }

}
```

Com a classe `Banco` e o método `adiciona()` criados, iremos refletir sobre como simular a inserção da informação em um banco de dados. Vamos relembrar a base da orientação à objetos: existe uma principal que possibilitará a criação de outros "n" objetos. É este conceito que iremos aproveitar, pois iremos associar à classe um atributo de lista que guardará todas as empresas.

Em `Banco`, criaremos um atributo privado estático , pois se relacionada à classe e não ao objeto. Importaremos o `List`, uma interface. Devemos tomar cuidado com a importação, pois ela deve referir-se à `List - java.util`.

```
paclage br.com.alura.gerenciador.servlet

import java.util.List;

public class Banco {

    private static List<E>

    public void adiciona(Empresa empresa) {

    }
}
```

Especificaremos que esta lista só guardará objetos do tipo `Empresa`. A seguir, deveremos decidir qual implementação usaremos, no caso usaremos `ArrayList() - java.util.ArrayList`. Como usaremos o mesmo *generics*, não precisamos preencher o `ArrayList`, isto é, ele também trabalhará com objetos do tipo `Empresa`. Feito isso, chamaremos o método `add()` que receberá `Empresa`.

Iremos, ainda, adicionar outro método que chamaremos de `getEmpresas()`, que também devolve uma lista. Lembrando que `lista`, é uma tributo da classe `Banco`.

```
paclage br.com.alura.gerenciador.servlet

import java.util.List;

public class Banco {

    private static List<Empresa> lista = new ArrayList<>();

    public void adiciona(Empresa empresa) {
        lista.add(empresa);

    public List<Empresa> getEmpresas(){
        return Banco.lista
    }

}
```

Dessa forma, por via da classe `Banco` acessamos a `lista` de empresas. As novas empresas serão adicionadas nesta lista. Testaremos a nossa aplicação no navegador: digitaremos a URL `localhost:8080/gerenciador/fornNovaEmpresa.html` e teremos acesso ao formulário com o campo "Nome". Escreveremos o nome da empresa "Google".

Ao pressionarmos o botão "Enviar", receberemos a mensagem `Empresa Google cadastrada com sucesso`. Como não definimos em nosso código nenhum comando de impressão, como `System.out.prinln()`, não teremos uma confirmação no console. Mas a mensagem indica que o cadastro foi efetuado.

Até agora, definimos as classes `Empresa` e `Banco`. Nosso modelo também foi construído por meio dos Servlet. Nosso próximo objetivo será fazer a listagem das empresas.