



Definido modelo

Transcrição

Aprendemos o funcionamento das requisições, seus diferentes métodos (GET e POST) e as formas de escrever estas especificações no código do Servlet.

A próxima meta é completar a aplicação, já que 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 uma nova classe chamada `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 possuir 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;
```

```
}
```

[COPIAR CÓDIGO](#)

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(int id) {  
        this.id = id;  
    }
```

```
    public String getNome() {  
        return nome;  
    }
```

```
    public void setNome(String nome) {  
        this.nome = nome;  
    }
```

```
}
```

[COPIAR CÓDIGO](#)

Em `NovaEmpresaServlet`, a partir dos parâmetros construídos, criaremos novo objeto da classe `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 response)
    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");
}

}
```

[COPIAR CÓDIGO](#)

Temos um objeto e encapsulamos `nome`, e agora simularemos um acesso ao banco de dados, o 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 ainda não existe no projeto, então precisaremos criá-la. O próprio Eclipse nos dá essa opção deixando o cursor sobre a classe `Banco` e clicando em *"Create class 'Banco'"*.

```
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");
}
```

```
}
```

[COPIAR CÓDIGO](#)

Na nova classe `Banco` , teremos:

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

```
public class Banco {  
}
```

[COPIAR CÓDIGO](#)

Assim, nosso código voltará a compilar. A partir dessa classe, 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 + " c  
}  
  
}
```

[COPIAR CÓDIGO](#)

A criação do método `adiciona()` também pode ser feita com auxílio da caixa de diálogo do Eclipse. Na classe `Banco` , teremos:

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

```
public class Banco {
```

```
    public void adiciona(Empresa empresa) {
```

```
    }
```

```
}
```

[COPIAR CÓDIGO](#)

Com a classe `Banco` e o método `adiciona()` criados, vamos refletir sobre como simular a inserção da informação em um banco de dados. Na orientação à objetos: existe uma classe principal que possibilitará a criação de inúmeros objetos que representem os dados. É este conceito que aproveitaremos aqui, pois iremos associar à classe um atributo de lista que guardará todas as empresas.

Em `Banco`, criaremos um atributo privado estático, pois se relaciona à classe e não ao objeto. Importaremos o `List`, uma interface do pacote `java.util`.

Com *generics*, informaremos que essa lista só guardará objetos do tipo `Empresa`.

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

```
import java.util.List;
```

```
public class Banco {
```

```
    private static List<Empresa>
```

```
    public void adiciona(Empresa empresa) {
```

```
    }
```

```
}
```

[COPIAR CÓDIGO](#)

A seguir, deveremos decidir qual implementação usaremos - nesse caso, `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 um atributo estático da classe `Banco` .

```
package 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;
    }

}
```

[COPIAR CÓDIGO](#)

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

<http://localhost:8080/gerenciador/formNovaEmpresa.html>

(<http://localhost:8080/gerenciador/formNovaEmpresa.html>) e teremos acesso ao formulário com o campo "Nome". Nele, 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.println()` , 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.