

CETEJ35 - Java Web - JAVA_XXX (2024_01)

[Meus cursos](#) / [CETEJ35 - Web \(2024_01\)](#) / [Semana 02: 09/09 a 15/09](#) / [Página Dinâmica](#)

Página Dinâmica

✓ **Feito:** Ver

✓ **Feito:** Gastar pelo menos 20 minutos na atividade

✓ **Feito:** Passar pela atividade até o fim

Fecha: segunda-feira, 2 dez. 2024, 00:00

Nesta aula, nós transformamos a página estática em uma página dinâmica. Isso é necessário porque queremos que a tabela de cidades seja atualizada à medida que novas cidades são inseridas. Para isso, vamos precisar de mais uma tecnologia - o Freemarker. Em seguida, mudamos a página existente para uma nova pasta. Assim, o Spring Boot reconhece a página como uma página dinâmica. Também alteramos a extensão da página. O próximo passo é colocar o código dinâmico na página, usando a sintaxe do Freemarker. Também fazemos os ajustes necessários para implementar o padrão MVC no projeto.

ALTERANDO O CÓDIGO JAVA

Precisamos de uma classe para representar uma cidade. Para isso, vamos criar uma classe **Cidade** dentro do pacote **visao** (Figura a seguir).



A classe **Cidade** tem os atributos que representam uma cidade. No nosso caso, **nome** e **estado**. Nesse momento, tudo que precisamos é de um construtor que receba valores iniciais e **gets** que retornem os valores dos atributos. Observem que essa classe é imutável (**final**). Isso porque não esperamos ficar alterando os dados de uma cidade. Veja como ficou a classe na Figura a seguir.

```

1 package br.edu.utfpr.cp.esjava.crudcidades.visao;
2
3 public final class Cidade {
4     private final String nome;
5     private final String estado;
6
7     public Cidade(final String nome, final String estado) {
8         this.nome = nome;
9         this.estado = estado;
10    }
11
12    public String getEstado() {
13        return estado;
14    }
15
16    public String getNome() {
17        return nome;
18    }
19 }

```

Agora que já temos uma representação de cidade – a classe `Cidade`, podemos alterar a classe `CidadeController` para retornar uma lista de cidades. Vamos começar criando uma lista de cidades.

Para isso, vamos usar um `Set`. O `Set` é uma das interfaces do *Collections framework*, parte da biblioteca padrão do Java. A diferença entre o `Set` e outras interfaces do *framework* é que ele não permite elementos repetidos. Afinal, você não quer ter duas cidades com o mesmo nome no mesmo estado, não é mesmo?

Desde o Java 9, as interfaces do *Collections framework* tem o método estático `of`, usado para criar uma lista com valores pré-definidos. Criamos três novas cidades, estipulando um nome e estado para cada uma, conforme a Figura abaixo.

```

15 public String listar(Model memoria) {
16
17     var cidades = Set.of(
18         new Cidade("Cornélio Procópio", "PR"),
19         new Cidade("Assis", "SP"),
20         new Cidade("Itajaí", "SC")
21     );
22
23     memoria.addAttribute("listaCidades", cidades);

```

Agora, precisamos tornar essa lista de cidades (`cidades`) disponível para que o Freemarker seja capaz de acessá-la e colocar os valores na página. Isso é feito usando um objeto do tipo `org.springframework.ui.Model`.

Para isso, criamos um parâmetro para o método `CidadeController.listar()` do tipo `org.springframework.ui.Model` (linha 15). Esse parâmetro representa uma espécie de memória compartilhada durante as requisições. Toda vez que uma solicitação vier ou for enviada para o navegador, ela traz essa memória junto.

Essa memória pode armazenar valores que você precisa enviar ou receber do navegador. No nosso caso, precisamos enviar a lista de cidades. A classe `org.springframework.ui.Model` tem o método `addAttribute()`, que permite inserir um

par de valores. Nosso caso, queremos criar uma variável com o nome `listaCidades`. Essa é a mesma variável usada na página dinâmica pelo Freemarker para listar as cidades (Veja a Figura na Seção anterior). Como valor, queremos enviar o `Set cidades`, que tem atualmente três cidades (linha 23).

O último passo é ajustar o retorno do método para que ele reflita a mudança na página (linha 23). Veja abaixo o código completo.

```
1  package br.edu.utfpr.cp.esjava.crudcidades.visao;
2
3  import java.util.Set;
4
5  import org.springframework.stereotype.Controller;
6  import org.springframework.ui.Model;
7  import org.springframework.web.bind.annotation.GetMapping;
8
9  @Controller
10 public class CidadeController {
11
12     @GetMapping("/")
13     public String listar(Model memoria) {
14
15         var cidades = Set.of(
16             new Cidade("Cornélio Procópio", "PR"),
17             new Cidade("Assis", "SP"),
18             new Cidade("Itajaí", "SC")
19         );
20
21         memoria.addAttribute("listaCidades", cidades);
22
23         return "/crud";
24     }
25 }
```



Se você executar a aplicação nesse ponto, vai perceber que a cidade Londrina/PR é listada três vezes. A diretiva está funcionando, mas nós não atualizamos os valores com o nome da cidade e seu estado na página `crud.ftlh`. Por isso, vamos voltar na página `/resources/templates/crud.ftlh` e atualizar a tabela nas linhas 43 e 44 conforme a Figura a seguir.

```

40      <tbody>
41      <#list listaCidades as cidade >
42      <tr>
43      <td>${cidade.nome}</td>
44      <td>${cidade.estado}</td>
45      <td>
46      <div class="d-flex d-justify-content-center">
47      <a class="btn btn-warning mr-3">ALTERAR</a>
48      <a class="btn btn-danger">EXCLUIR</a>
49      </div>
50      </td>
51      </tr>
52      </#list>
53      </tbody>

```

Essa sintaxe é usada para acessar a variável cidade, que representa um objeto do tipo **Cidade**. A classe **Cidade**, que criamos anteriormente, tem dois atributos: **nome** e **estado**. Esses atributos são acessados por meio dos **gets**, que retornam o valor atual para cada cidade. Todo código dentro da diretiva **list** é repetido para cada elemento dentro de **listaCidades**. Ao executar a aplicação, você deve ver os valores da lista, conforme a Figura abaixo.

GERENCIAMENTO DE CIDADES

UM CRUD PARA CRIAR, ALTERAR, EXCLUIR E LISTAR CIDADES

Cidade:

Estado:

CRIAR

Nome	Estado	Ações
Itajaí	SC	<div>ALTERAR</div> <div>EXCLUIR</div>
Cornélio Procopio	PR	<div>ALTERAR</div> <div>EXCLUIR</div>
Assis	SP	<div>ALTERAR</div> <div>EXCLUIR</div>

Agora nossa página dinâmica retorna valores presentes em uma lista. Mas não é só isso que você espera, certo? Queremos inserir valores dinamicamente na lista. Também queremos ser capazes de alterar e excluir esses valores. Precisamos de um CRUD completo. Vamos fazer isso na próxima aula, além de conhecer dois padrões muito utilizados no desenvolvimento Web: **MVC** e **Front-controller**.


A próxima aula encerra esse projeto básico e, em seguida, começamos um novo projeto com várias funcionalidades novas.

Retroceder

Avançar

Seguir para...

Verificação de aprendizado - Página Dinâmica ►

✉ Contate o suporte do site 

Você acessou como RAFAEL ROCHA DA SILVA PROENCA (Sair)
CETEJ35 - Web (2024_01)

Tema

Adaptable

Boost

Clássico

Campus

Apucarana

Campo Mourão

Cornélio Procópio

Curitiba

Dois Vizinhos

Francisco Beltrão

Guarapuava

Londrina

Medianeira

Pato Branco

Ponta Grossa

Reitoria

Santa Helena

Toledo

UTFPR

Ajuda

Chat UTFPR

Calendário Acadêmico

Biblioteca

e-Mail

Nuvem (OwnCloud)

Produção Acadêmica

Secretaria Acadêmica

Sistemas Corporativos

Sistema Eletrônico de Informação - SEI

Suporte ao usuário

Criação de curso

Comunidade

Português - Brasil (pt_br)


Deutsch (de)

English (en)

Português - Brasil (pt_br)

Resumo de retenção de dados

Baixar o aplicativo móvel.

🗨 Dê um feedback sobre este software 

Universidade Tecnológica Federal do Paraná - UTFPR
Suporte ao usuário

