

CETEJ35 - Java Web - JAVA_XXX (2024_01)

[Meus cursos](#) / [CETEJ35 - Web \(2024_01\)](#) / [Semana 04: 23/09 a 29/09](#) / [Criando, Alterando e Excluindo](#)

Criando, Alterando e Excluindo

✓ **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 finalizamos a implementação das quatro operações CRUD. Isso significa que nosso usuário será capaz de criar, alterar, excluir e listar as cidades em uma base de dados. Observe que ainda estamos usando uma base local, baseada em uma lista em memória. Nós vamos evoluir esse projeto até integrarmos essa base com um banco de dados.

AJUSTANDO O CÓDIGO

Durante o desenvolvimento desse projeto, vamos fazer várias mudanças no código. Tradicionalmente, precisaríamos parar a execução do aplicativo e iniciar novamente para ver cada alteração. Isso é trabalhoso e pouco produtivo. Por isso, o Spring Boot fornece um recurso chamado *Hot Swapping*. Esse recurso reinicia a aplicação cada vez que o código for salvo. Para ativar o recurso, basta inserir a dependência `spring-boot-devtools`, conforme mostra a Figura abaixo.



```
35
36     <dependency>
37         <groupId>org.springframework.boot</groupId>
38         <artifactId>spring-boot-devtools</artifactId>
39     </dependency>
```

Nesse primeiro momento, vamos continuar usando um objeto `Set` para armazenar a lista de cidades. Contudo, o código atual mantém a lista como uma variável local. Para que a lista fique acessível para os outros métodos que vamos criar, vamos transformar a lista em um atributo da classe, conforme mostra a Figura a seguir.

```
9  @Controller
10 public class CidadeController {
11
12     private Set<Cidade> cidades;
13 }
```

No código anterior, a lista era iniciada usando uma fábrica por meio do método `Set.of`. Esse método cria uma lista imutável. Agora, precisamos de uma lista mutável, pois vamos inserir e remover cidades da lista. Para isso, vamos iniciar uma lista vazia usando o construtor da classe `CidadeController`. Com isso, podemos remover a variável `cidades` do método `listar`. Veja como ficou o código completo na Figura abaixo.

```

10 @Controller
11 public class CidadeController {
12
13     private Set<Cidade> cidades;
14
15     public CidadeController() {
16         cidades = new HashSet<>();
17     }
18
19     @GetMapping("/")
20     public String listar(Model memoria) {
21
22         memoria.addAttribute("listaCidades", cidades);
23
24         return "/crud";
25     }
26 }

```

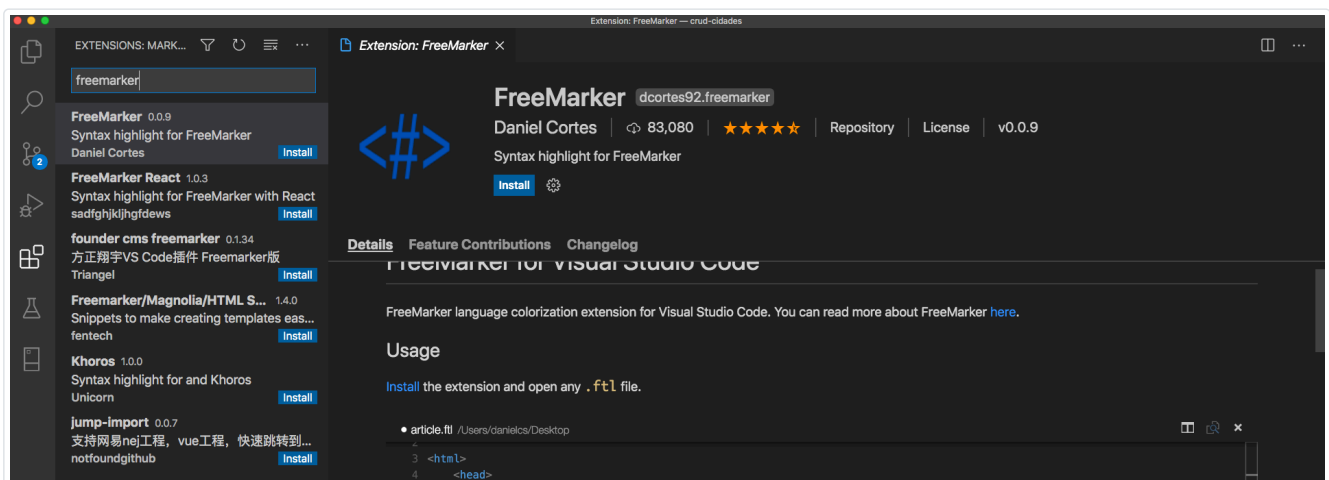
O **HashSet** é uma das implementações do **Set**, assim como o **ArrayList** é uma das implementações do **List**. Tanto o **List** quanto o **Set** fazem parte do *Collections Framework* do Java.

Se você está usando o VS Code, deve ter percebido que perdemos a formatação e o destaque do código quando alteramos a extensão da página Web de **crud.html** para **crud.ftlh**. Isso porque o VS Code não reconhece a extensão **.ftlh** por padrão. Por isso, vamos usar um plugin para o Freemarker, produzido por Daniel Cortes.



Se você está usando uma IDE ou outro editor, procure saber como é o suporte à páginas Freemarker na sua IDE/editor.

Para encontrar esse plugin, basta ir na aba de extensões do VS Code e procurar por *Freemarker*. Clique no botão instalar, e o plugin já estará disponível.



Contudo, se você abrir a página **crud.ftlh**, vai perceber que nada aconteceu. Isso porque o plugin, por padrão, espera por páginas com extensão **.ftl**, que é o padrão do Freemarker. Mas o Spring Boot usa um outro padrão, o **.ftlh**. Não se preocupe, isso é uma chance para usarmos o mecanismo de flexibilização do Spring Boot: o arquivo **resources/application.properties**.

O Spring Boot é altamente flexível, e o arquivo `resources/application.properties` tem um papel fundamental nessa flexibilidade. Por meio desse arquivo é possível fazer ajustes como o que precisamos nesse momento. Para que o Spring Boot reconheça uma página com a extensão `.ftl`, basta adicionar a linha de código abaixo direto no arquivo `resources/application.properties`.

```
spring.freemarker.suffix=.ftl
```

Salve o arquivo e, em seguida, altere a extensão do arquivo `crud.ftlh` para `crud.ftl`. Pronto, agora temos o melhor dos dois mundos: *Syntax Highlighting* e o Spring Boot funcionando!

Ao executar o projeto você vai perceber que não tem mais uma lista de cidades na tabela. Isso porque inicializamos a lista vazia. Nas próximas seções vamos implementar os demais métodos do CRUD e popular a tabela..

O código desenvolvido nesta Seção está disponível no [Github](#), na branch `semana03-10-crud-ajustes`.


Retroceder

Avançar

◀ Atividade I WebConf

Seguir para...

Verificação de aprendizado - Criando, Alterando e Excluindo ▶

✉ 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 Procopio
- 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

