

CETEJ35 - Java Web - JAVA_XXX (2024_01)

[Meus cursos](#) / [CETEJ35 - Web \(2024_01\)](#) / [Semana 01: 02/09 a 08/09](#) / [Gerenciamento de Cidades](#)

Gerenciamento de Cidades

✓ **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

Nessa primeira aula prática, daremos início a um projeto do tipo CRUD. Um CRUD é um aplicativo que permite criar (*create*), ler (*read*), alterar (*update*) e excluir (*delete*) dados. Como referência, vamos usar os dados de cidades. Uma cidade está associada a um único estado. Portanto, para criar uma cidade precisamos informar o nome da cidade e o estado onde ela se encontra.

EXPLORANDO O PROJETO CRIADO

A Figura abaixo mostra a estrutura do projeto Maven criado pelo *Spring Initializr*. Na raiz, o arquivo mais importante é o **pom.xml**. É esse arquivo que armazena as configurações do Maven, o que inclui meta-dados e dependências, entre outros. Dentro da pasta **src**, existem duas pastas: **main** e **test**. Enquanto a pasta **main** é usada para os arquivos-fonte, a pasta **test** é usada para os arquivos de teste, como testes unitários, por exemplo.



```

.
├── HELP.md
├── LICENSE
├── mvnw
├── mvnw.cmd
├── pom.xml
├── src
│   ├── main
│   │   ├── java
│   │   │   ├── br
│   │   │   │   ├── edu
│   │   │   │   │   ├── utfpr
│   │   │   │   │   │   ├── cp
│   │   │   │   │   │   │   ├── espjava
│   │   │   │   │   │   │   │   ├── crudidades
│   │   │   │   │   │   │   │   └── CrudCidadesApplication.java
│   │   ├── resources
│   │   │   ├── application.properties
│   │   │   ├── static
│   │   │   └── templates
│   └── test
│       ├── java
│       │   ├── br
│       │   │   ├── edu
│       │   │   │   ├── utfpr
│       │   │   │   │   ├── cp
│       │   │   │   │   │   ├── espjava
│       │   │   │   │   │   │   ├── crudidades
│       │   │   │   │   │   │   └── CrudCidadesApplicationTests.java
└── wrapper
    ├── MavenWrapperDownloader.java
    ├── maven-wrapper.jar
    └── maven-wrapper.properties

```



A pasta **main** se divide em **java** e **resources**. Enquanto a pasta **java** guarda as classes Java usadas no projeto, a pasta **resources** guarda os arquivos de configuração (**application.properties**), além de outras duas pastas. A pasta **static** é usada para armazenar conteúdo estático, como páginas **html** estáticas, enquanto a pasta **templates** é usada para armazenar conteúdo dinâmico, como páginas **html** que são alteradas dinamicamente durante a execução da aplicação.

Vocês vão perceber também que o projeto já vem com duas classes Java: uma dentro da pasta **main** e outra na pasta **test**. A classe Java dentro da pasta **main** contém o método que inicializa a aplicação, enquanto a classe Java dentro da pasta **test** é um teste unitário de exemplo.

Com o projeto aberto no seu editor ou IDE de preferência, vamos explorar a classe **CrudCidadesApplication.java**. Como vocês podem perceber, é uma classe Java comum. Ela tem o método **main**, que também é comum em aplicações Java. Esse método é primeiro método a ser executado quando a aplicação é inicializada.

O código dentro do método **main**, na linha 10, e anotação **@SpringBootApplication**, na linha 6, são usados pelo Spring Boot para definir que esse é um projeto Spring Boot e inicializar as configurações padrão para esse tipo de projeto.

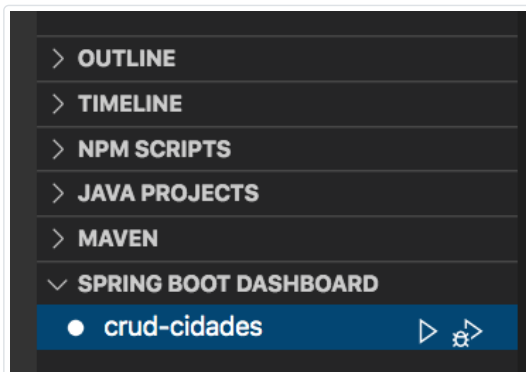
Este projeto está disponível no [Github](#), na branch **semana01-01-projeto-vazio**.

```

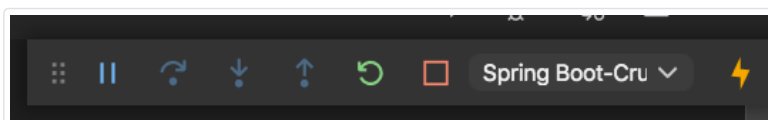
1  package br.edu.utfpr.cp.esjava.crudcidades;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6  @SpringBootApplication
7  public class CrudCidadesApplication {
8
9      Run | Debug
10     public static void main(String[] args) {
11         SpringApplication.run(CrudCidadesApplication.class, args);
12     }
13 }

```

Se você está usando o VS Code com os plug-ins para Java e Spring Boot recomendados na aula anterior, você pode executar a aplicação clicando no botão **play**, no janel **SPRING BOOT DASHBOARD**.



Ao entrar em execução, você vai notar a janela de controle que, normalmente, é mostrada no canto superior direito da janela. O botão **STOP** (quadrado vermelho) é usado para parar a aplicação.



O painel inferior do editor mostra o log de execução do Spring Boot. A penúltima mensagem, **Tomcat started on port(s): 8080 (http) with context path ''**, mostra que a aplicação está em execução e pode ser acessada na porta 8080. Se você estiver usando uma IDE, como Netbeans ou Eclipse, o procedimento de execução não é muito diferente. Basta localizar o botão **play** e dar início à execução. Se você for um adepto da linha de comando, você pode inicializar o aplicativo entrando na pasta do projeto pelo seu terminal e digitando **mvn spring-boot:run**. Observe que você precisa ter seu ambiente configurado, com Java e Maven instalados.

É importante ressaltar que você deve usar a IDE ou editor que você tiver mais familiaridade para ter produtividade neste curso. Se você já tiver mais habilidade com tecnologia e quiser testar algo novo, então fique à vontade para se aventurar com o VS Code. Nada nesse curso é específico do VS Code, por isso, todo o conteúdo pode ser feito no editor/IDE da sua preferência.

TERMINAL PROBLEMS OUTPUT **DEBUG CONSOLE** Filter (e.g. text, lexclude) Spring Boot-CrudCidadesA

```
t/9.0.43]
2021-03-02 16:17:16.775 INFO 7251 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2021-03-02 16:17:16.775 INFO 7251 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2080 ms
2021-03-02 16:17:17.244 INFO 7251 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2021-03-02 16:17:17.713 INFO 7251 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-03-02 16:17:17.732 INFO 7251 --- [main] b.e.u.c.e.c.CrudCidadesApplication : Started CrudCidadesApplication in 4.0 seconds (JVM running for 5.135)
```

Na sua máquina local, basta abrir o navegador e digitar: **http://localhost:8080**. Contudo, observe que a aplicação ainda não faz nada. Por isso, tudo que você vai conseguir é uma mensagem de **erro 404**. Não se preocupe, agora vamos começar a moldar a aplicação da forma como queremos.

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Mar 02 16:22:04 BRT 2021
There was an unexpected error (type=Not Found, status=404).

Retroceder

Avançar

◀ Fórum de discussões - Abertura de Tickets de Suporte

Seguir para...

Verificação de aprendizado - Gerenciamento de Cidades ▶

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

