

Introdução a Serviços Web Java com Spring Boot e JPA/Hibernate



Definições iniciais

- **Spring Framework:** é um framework desenvolvido para a plataforma Java que objetiva facilitar o desenvolvimento de aplicações e utiliza bastante os conceitos de inversão de controle e injeção de dependência (forma de aplicar a inversão de controle), esses conceitos basicamente buscam diminuir o acoplamento do código (em outras palavras ele provê instâncias de classes que um objeto precisa sem que este precise instanciá-los). Um exemplo de injeção de dependência é o uso da anotação **@Autowired** (utilizaremos futuramente).

Outro Exemplo (Constructor Injection):

```
public class VendaItem {  
    public void vender(Item item){  
        // Aqui temos um acompanhamento!!!  
        // Para vender um item, a classe VendaItem precisará saber o nome do  
        // arquivo que a classe Gravadora necessita  
        Gravadora gravadora = new Gravadora("arquivol.txt");  
        gravadora.gravar(item);  
    }  
}
```

```
public class VendaItem {  
    private Gravadora gravadora;  
  
    // Aqui a gravadora é "injetada" na VendaItem  
    public VendaItem(Gravadora gravadora) {  
        this.gravadora = gravadora;  
    }  
  
    // Como a gravadora "já vem pronta", a VendaItem passa  
    // a saber vender sem depender do arquivo que a gravadora  
    // necessita  
    public void vender(Item item){  
        gravadora.gravar(item);  
    }  
}
```



Definições iniciais

- **Spring Boot:** é uma ferramenta que objetiva facilitar o processo de configuração e publicação de aplicações que usem o framework Spring, ou seja, a ideia é ter o projeto rodando rapidamente sem grandes esforços com configurações.
- **Spring Tool Suite:** é uma versão do Eclipse que já vem com os plugins para trabalhar com Spring Boot.
 - O Spring Tool Suite será o IDE que utilizaremos para criar serviços web Java.
 - Para obtê-lo basta ir no link: <https://spring.io/tools> ou procurar por Spring Tool Suite no Google.
 - A versão Windows é apenas um arquivo compactado que irá gerar um diretório. Nesse diretório basta procurar o arquivo .exe com o nome de SpringToolSuite e executá-lo (para facilitar sugiro colocar um atalho na área de trabalho).



Definições iniciais

- **JPA (Java Persistence API):** é a especificação que define como camadas de persistência devem trabalhar, ou seja, ela define como deve ser feito o mapeamento objeto-relacional (ORM).
- **Hibernate:** é uma implementação de camada de persistência, ou seja, é uma ferramenta que implementa o mapeamento objeto-relacional.

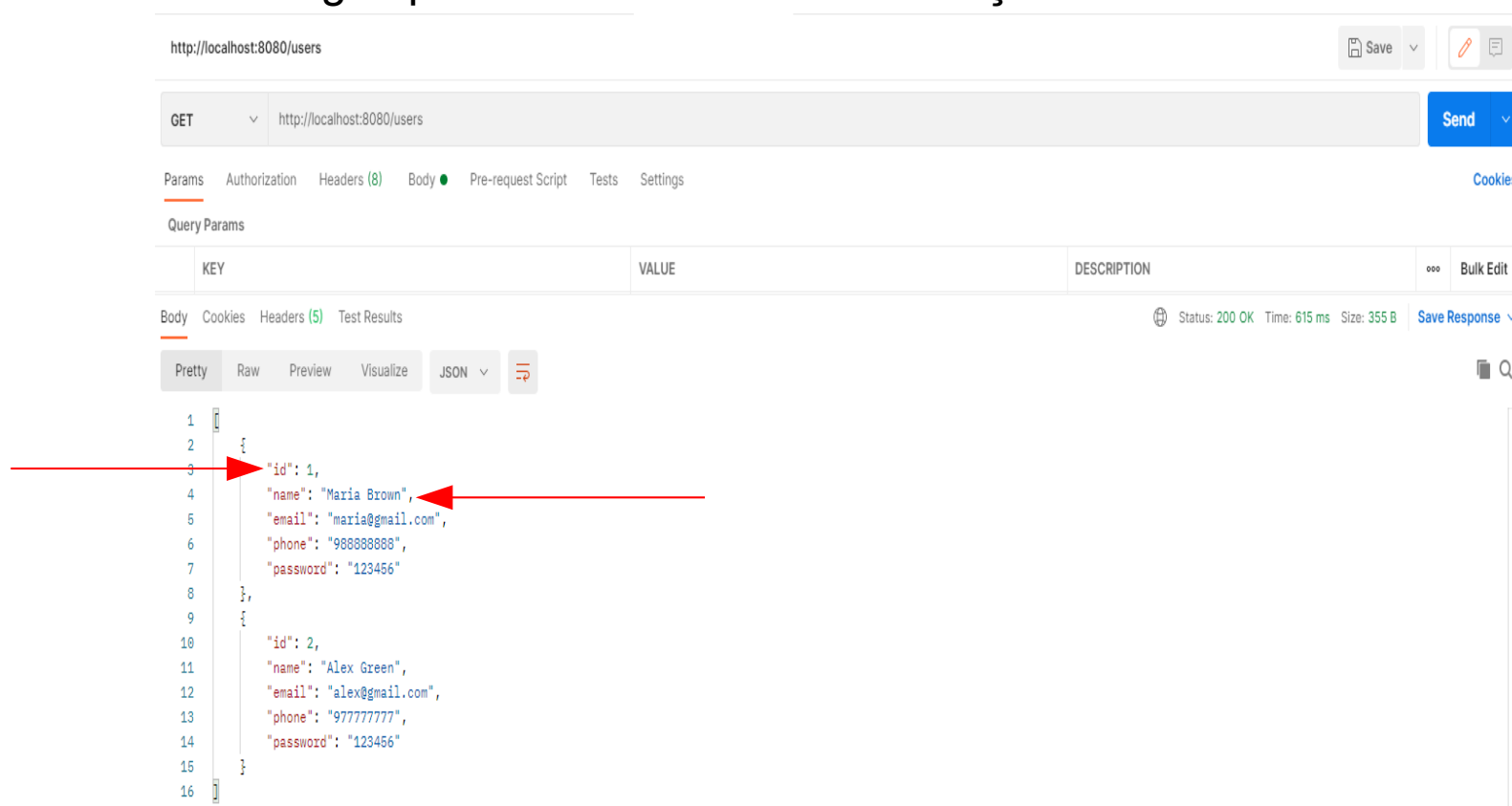
Em outras palavras, o JPA seria a “interface” e o Hibernate é a “implementação” dessa interface.

Existem diversos concorrentes para o Hibernate como por exemplo EclipseLink (Eclipse Foundation) ou OpenJPA (Apache).



Ferramentas Complementares

- **Postman:** é uma ferramenta que possibilita analisar/testar serviços web Java com facilidade. Ele permite realizar solicitações HTTP sem a necessidade de escrever uma série de códigos para analisar/testar um serviço.



<http://localhost:8080/users>



Ferramentas Complementares

The screenshot displays the Postman REST client interface. At the top, the URL bar shows `http://localhost:8080/users/1`. Below it, the method is set to `PUT` and the same URL is repeated. A red arrow points to the URL in the method bar, and another red arrow points to the `Send` button. The `Body` tab is selected, showing a JSON payload:

```
{
  "name": "Bob Brown",
  "email": "bob@gmail.com",
  "phone": "977557755"
}
```

 A red arrow points to the `"name": "Bob Brown",` line. The bottom section shows the response area with tabs for `Body`, `Cookies`, `Headers`, and `Test Results`. The `Body` tab is active, showing a status of `200 OK`, a time of `148 ms`, and a size of `255 B`. The response content is currently empty.



Ferramentas Complementares

The screenshot displays a REST client interface with the following components:

- URL Bar:** Shows the URL `http://localhost:8080/users/1`. A red arrow points to the end of the URL.
- Method:** A dropdown menu is set to `PUT`.
- Request Body:** The `Body` tab is selected, showing a JSON object:

```
1 {  
2   "name": "Bob Brown",  
3   "email": "bob@gmail.com",  
4   "phone": "977557755"  
5 }
```

A red arrow points to the `"name"` field.
- Response:** The `Body` tab is selected, showing a JSON object:

```
1 {  
2   "id": 1,  
3   "name": "Bob Brown",  
4   "email": "bob@gmail.com",  
5   "phone": "977557755",  
6   "password": "123456"  
7 }
```

Red arrows point to the `"id": 1` and `"name": "Bob Brown"` fields.
- Status Bar:** At the bottom right, it shows `Status: 200 OK`, `Time: 152 ms`, `Size: 255 B`, and a `Save Response` button.



Ferramentas Complementares

http://localhost:8080/users

GET http://localhost:8080/users Send

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 23 ms Size: 351 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 1,
4     "name": "Bob Brown",
5     "email": "bob@gmail.com",
6     "phone": "977657755",
7     "password": "123456"
8   },
9   {
10    "id": 2,
11    "name": "Alex Green",
12    "email": "alex@gmail.com",
13    "phone": "977777777",
14    "password": "123456"
15  }
16 }
```

http://localhost:8080/users



Ferramentas Complementares

- **Banco de dados H2:** é um banco de dados escrito em Java que funciona em memória volátil, ou seja, não é armazenado em HD/SSD. Dessa forma, a cada instanciação da aplicação ele será reconstruído.
- O H2 possui um console de fácil uso e acesso através do navegador.
- O objetivo desse banco é não gastar tempo configurando bancos de dados e criando a estrutura desses dados. Em outras palavras, é um banco muito interessante para ser usado como banco para testes da aplicação que está sendo desenvolvida. Nesse contexto, apenas quando a aplicação estiver mais madura precisaremos nos preocupar com questões específicas referentes ao banco de dados que efetivamente pretendemos utilizar.

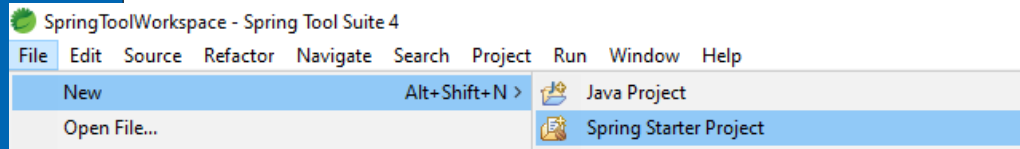


Ferramentas Complementares

- **Maven:** é uma ferramenta gerenciadora de dependências que objetiva facilitar a criação de projetos Java (também há suporte para outras tecnologias). Ele possibilita que o desenvolvedor não tenha de saber de imediato quais bibliotecas o projeto precisa para compilar e executar, além de automatizar o processo de construção do pacote do aplicativo.
- O Spring Tools Suite que vamos utilizar já possui integração nativa com o Maven, assim diversos processos trabalhosos de configuração já serão feitos de forma automatizada.



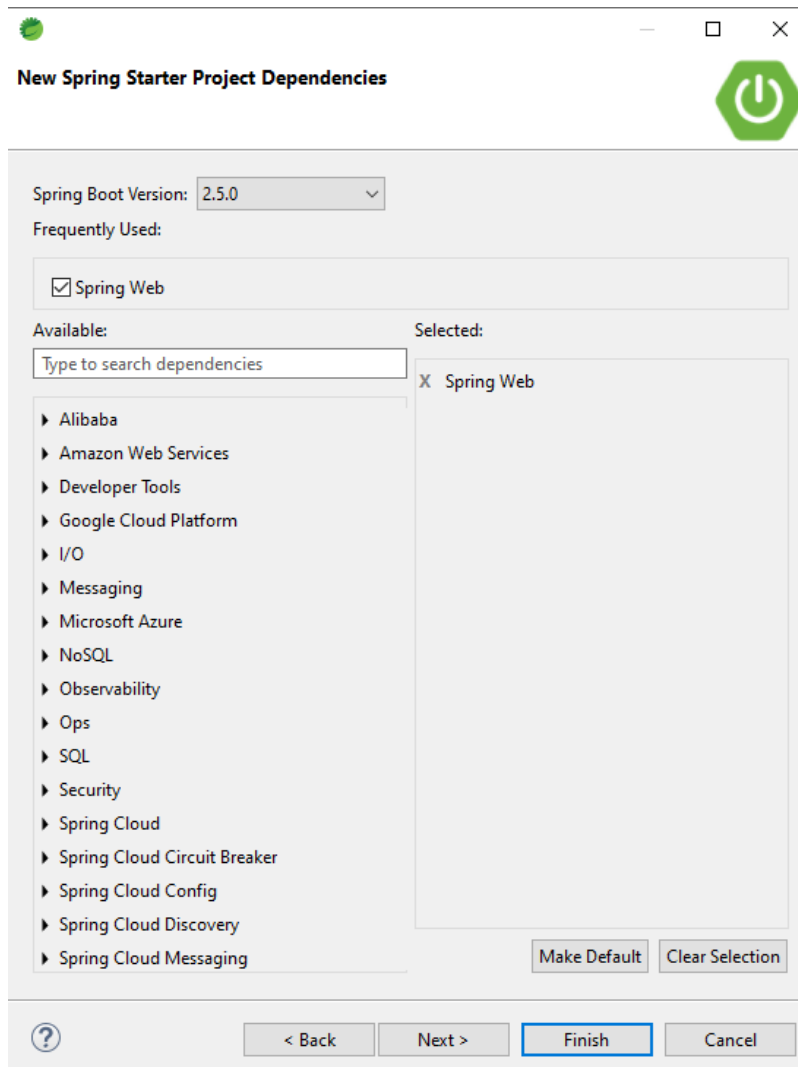
Criando o projeto (primeiro serviço)



- **Name:** nome do projeto.
- **Group:** pacote básico do seu projeto.
- **Artifact:** é o mesmo nome que o nome do projeto.
- **Package:** pacote básico do projeto.

A screenshot of the Spring Starter Project wizard. The 'Service URL' is set to 'https://start.spring.io'. The 'Name' is 'abcde'. The 'Use default location' checkbox is checked. The 'Location' is 'C:\Users\Giovany\Documents\SpringToolWorkspace\abcde'. The 'Type' is 'Maven', 'Packaging' is 'Jar', 'Java Version' is '11', and 'Language' is 'Java'. The 'Group' is 'com.servicos', 'Artifact' is 'abcde', 'Version' is '0.0.1-SNAPSHOT', 'Description' is 'Curso de Spring Boot', and 'Package' is 'com.servicos.abcde'. The 'Working sets' section has 'Add project to working sets' unchecked. The 'Next >' button is highlighted with a red arrow.

Criando o projeto (primeiro serviço)

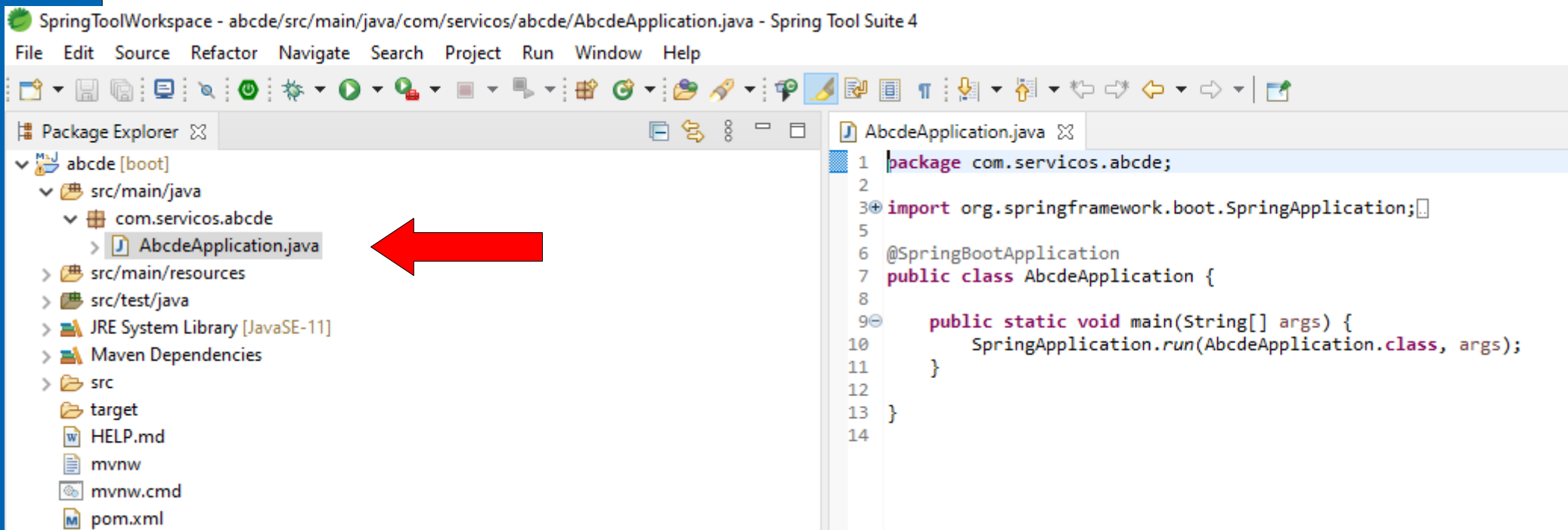


- Vamos selecionar **Spring Web** e clicar em **Finish**.



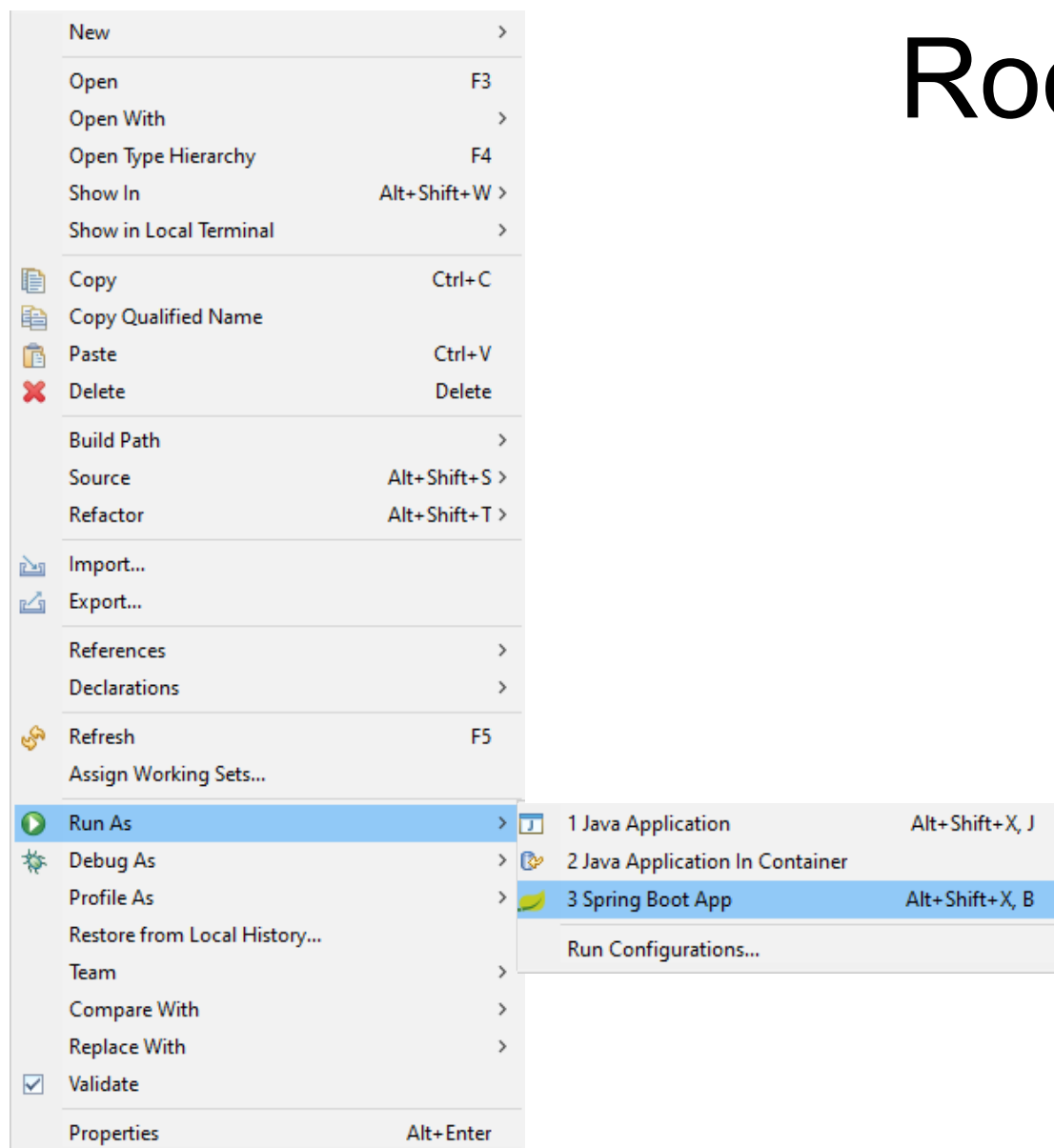
Criando o projeto (primeiro serviço)

- Com o projeto criado já temos o ponto de partida para o nosso serviço Java:



Rodando o serviço

- Clicando com o botão direito na classe principal (AbcdeApplication) e selecionando Run As→Spring Boot App temos a execução da aplicação.



Rodando o serviço

```
AbcdeApplication.java
1 package com.servicos.abcde;
2
3 import org.springframework.boot.SpringApplication;
4
5
6 @SpringBootApplication
7 public class AbcdeApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(AbcdeApplication.class, args);
11     }
12 }
13
14
```

Console

abcde - AbcdeApplication [Spring Boot App] C:\sts-4.10.0.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe (9 de jun. de 2021 15:22:51)

```

  ____  _
 / ___|| | | |
| |___| |_| |
 \___ \|  _/
      |_|_|

:: Spring Boot :: (v2.5.0)

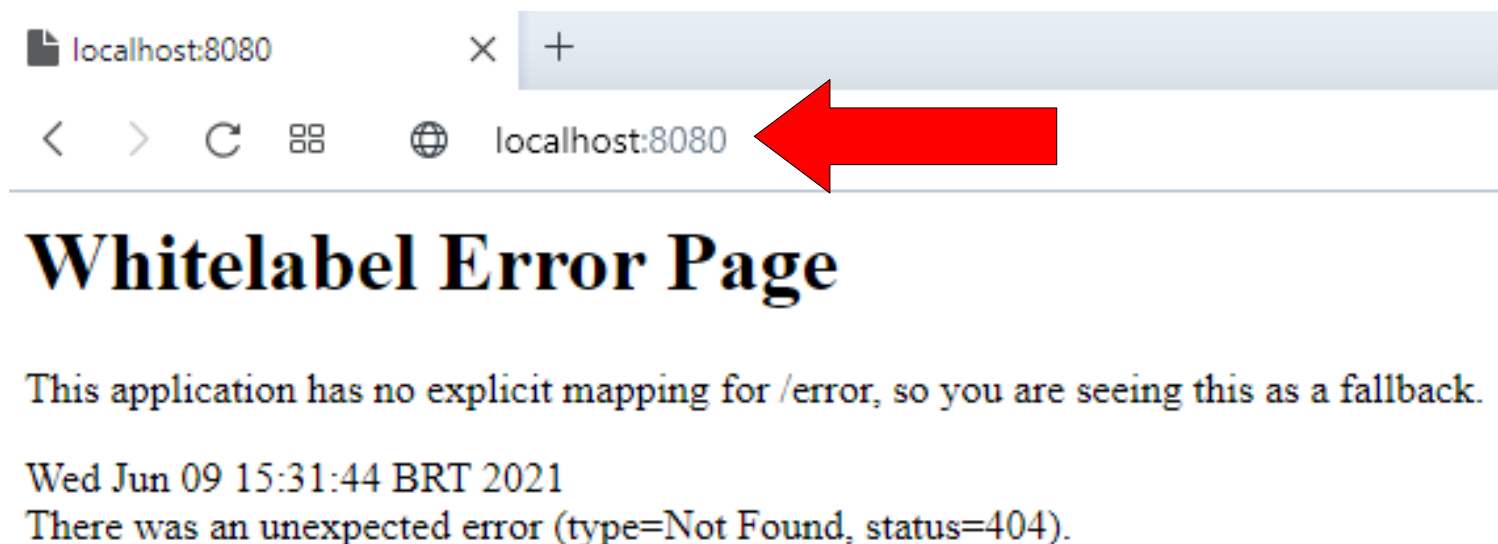
2021-06-09 15:22:56.273 INFO 14216 --- [main] com.servicos.abcde.AbcdeApplication : Starting AbcdeApplication using Java 15.0.2 on DESKTOP-JKB61SJ with PID 14216
2021-06-09 15:22:56.278 INFO 14216 --- [main] com.servicos.abcde.AbcdeApplication : No active profile set, falling back to default profiles: default
2021-06-09 15:22:57.946 INFO 14216 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-06-09 15:22:57.965 INFO 14216 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-06-09 15:22:57.966 INFO 14216 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.46]
2021-06-09 15:22:58.118 INFO 14216 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2021-06-09 15:22:58.118 INFO 14216 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1749 ms
2021-06-09 15:22:58.794 INFO 14216 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-06-09 15:22:58.810 INFO 14216 --- [main] com.servicos.abcde.AbcdeApplication : Started AbcdeApplication in 3.327 seconds (JVM running for 5.36)
2021-06-09 15:22:58.812 INFO 14216 --- [main] o.s.b.a.ApplicationAvailabilityBean : Application availability state LivenessState changed to CORRECT
2021-06-09 15:22:58.815 INFO 14216 --- [main] o.s.b.a.ApplicationAvailabilityBean : Application availability state ReadinessState changed to ACCEPTING_TRAFFIC
```



Testando o serviço

Notar que o serviço está rodando em **localhost** na porta **8080**

- Basta ir em um navegador:



- A mensagem que aparece é a mensagem personalizada de erro do Spring Boot.



Dúvidas?

