

Curso de Jogos Digitais

Disciplina de Tecnologias Web

Aula 16

Implementação do Back-end



Professor: André Flores dos Santos



Mas, o que é Spring Boot?

O **Spring Boot** é um **framework Java open source** que tem como objetivo facilitar o processo de criação e configuração em aplicações Java. Consequentemente, ele traz mais agilidade para o processo de desenvolvimento, uma vez que devs conseguem **reduzir o tempo gasto com as configurações iniciais**.

Com o Spring Boot conseguimos abstrair e facilitar a configuração de, por exemplo:

- Servidores;
- Gerenciamento de dependências;
- Configurações de bibliotecas;
- Métricas & health checks (testes);
- Entre outros!

Como o Spring Boot funciona?

Para realizar todo esse processo o Spring Boot utiliza um conceito chamado **convenção sobre configuração**.

Mas o que isso significa? Significa que é uma ferramenta que decide para você a melhor forma de se fazer algo. É o que chamamos de ferramenta opinativa, ela toma as decisões no nosso lugar baseado em convenções, aplicando configurações padrões e facilitando o trabalho.

No entanto ela **não é inflexível** e ainda permite uma configuração diferente da *default* caso o usuário assim deseje.

Por exemplo, você pode alterar para que ele utilize o Jetty como servidor ao invés do Tomcat que é a configuração padrão.

Uma das maiores vantagens que o Spring Boot trouxe ao desenvolvimento é que toda essa configuração não necessita mais ser realizada pelos temidos XMLs, embora ele ainda suporte esse tipo de configuração. A maior parte da **configuração pode ser feita de forma programática** via anotações.

O Spring Boot é composto por vários módulos que ajudam nesse processo. Alguns deles são:

Spring Boot

É o módulo principal que ajuda na configuração e integração dos outros módulos.

Spring Boot Starters

Os starters são dependências que agrupam outras dependências com um propósito em comum.

Dessa forma, somente uma configuração é realizada no seu gerenciador de dependências.

Por exemplo, o spring-boot-starter-amqp, é um starter que permite a construção de soluções de mensageria baseadas em AMQP e RabbitMQ.

Ao realizar a configuração no meu gerenciador de dependência se define somente o starter:

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
  </dependency>
</dependencies>
```

O Spring Boot é composto por vários módulos que ajudam nesse processo. Alguns deles são:

- Spring Boot Starter Web:** Auxilia na construção de aplicações web trazendo já disponíveis para uso Spring MVC, Rest e o Tomcat como servidor.
- Spring Boot Starter Test:** Contém a maioria das dependências necessárias para realizar testes da sua aplicação: Junit, AssertJ, Hamcrest, Mockito, entre outros
- Spring Boot Starter Data JPA:** Facilita a construção da nossa camada de persistência, ajudando na abstração do nosso banco de dados provendo uma série de facilidades para criação de repositories, escrita de queries, entre outros.
Como podem ver, reduzem o número de dependências adicionadas, deixando meu arquivo muito mais limpo.

Spring Boot Autoconfigure

Como dito anteriormente o **Spring Boot trabalha de forma opinativa, tomando decisões para você.**

Mas baseado em que? Essas decisões padrões são baseadas através do conteúdo do seu *classpath*.

O *Autoconfigure* é responsável por ler este conteúdo e realizar as configurações necessárias para que a aplicação funcione. É ele quem gerencia todo o processo de configuração da aplicação.

Spring Boot Actuator

O Spring Boot Actuator é uma ferramenta que permite monitorar e gerenciar as aplicações implantadas. Dentre os recursos disponibilizados temos:

- **Métricas:** Obtém e disponibiliza diversos dados da nossa aplicação, como por exemplo, espaço em disco, memória, tempo de resposta etc.
- **Logging:** Facilita o acesso ao arquivo de log da aplicação por meio de um *endpoint* específico.
- **Heath Checks:** (verificações de saúde) são endpoints que permitem monitorar o estado de uma aplicação em tempo real. Elas verificam se vários aspectos da aplicação estão funcionando corretamente, como conectividade com o banco de dados, disponibilidade de serviços externos, uso de memória, e mais. Esses checks são úteis para garantir que a aplicação está em um estado saudável e pronta para responder a solicitações.
- **Informações da Aplicação:** Permite a disponibilização de informações da aplicação. Por exemplo, versão, informações do git etc.

Spring Boot Test

O Spring Boot Test contém funcionalidades úteis e anotações que facilitam e ajudam a testar sua aplicação.

Spring Boot Devtools

Spring Boot Devtools é um conjunto de funcionalidades que ajuda o trabalho de qualquer dev. Como, por exemplo, restart automático da aplicação quando ocorre alguma mudança no código.

Spring Tool Suite

O Spring nos fornece uma IDE totalmente customizada para o desenvolvimento de aplicações do ecossistema spring: o **Spring Tool Suite (STS)**.

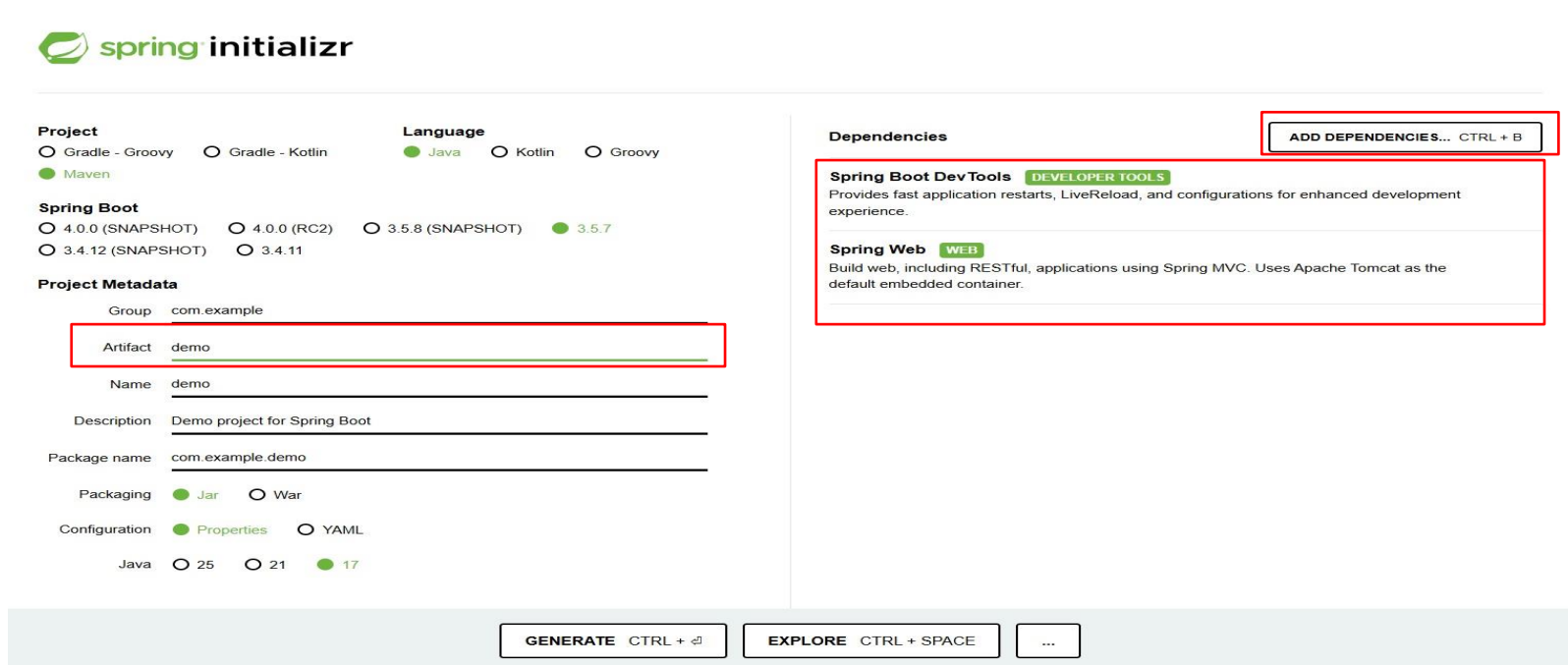
O STS é uma IDE baseada em Eclipse que já vem com algumas funcionalidades facilitadoras para projetos Spring.

download no site oficial.

Além da IDE baseada em Eclipse, o STS já está disponível como plugin para VSCode.

Spring Initializr

E para facilitar a criação de aplicações utilizando outras IDEs a Spring disponibilizou o **Spring Initializr**. Permite a criação de projetos Sprint Boot de forma facilitada.



The screenshot shows the Spring Initializr web interface. It is divided into several sections: Project, Language, Spring Boot, Project Metadata, Dependencies, and a bottom bar with action buttons. The 'Project' section has radio buttons for 'Gradle - Groovy', 'Gradle - Kotlin', and 'Maven' (selected). The 'Language' section has radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'. The 'Spring Boot' section has radio buttons for '4.0.0 (SNAPSHOT)', '4.0.0 (RC2)', '3.5.8 (SNAPSHOT)', '3.5.7' (selected), '3.4.12 (SNAPSHOT)', and '3.4.11'. The 'Project Metadata' section has input fields for 'Group' (com.example), 'Artifact' (demo), 'Name' (demo), 'Description' (Demo project for Spring Boot), and 'Package name' (com.example.demo). The 'Packaging' section has radio buttons for 'Jar' (selected) and 'War'. The 'Configuration' section has radio buttons for 'Properties' (selected) and 'YAML'. The 'Java' section has radio buttons for '25', '21', and '17' (selected). The 'Dependencies' section has a button 'ADD DEPENDENCIES... CTRL + B' and two dependency cards: 'Spring Boot DevTools' (DEVELOPER TOOLS) and 'Spring Web' (WEB). The bottom bar has buttons 'GENERATE CTRL + G', 'EXPLORE CTRL + SPACE', and '...'.

Project

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ Maven

Language

☒ Java ☐ Kotlin ☐ Groovy

Spring Boot

☐ 4.0.0 (SNAPSHOT) ☐ 4.0.0 (RC2) ☐ 3.5.8 (SNAPSHOT) ☒ 3.5.7 ☐ 3.4.12 (SNAPSHOT) ☐ 3.4.11

Project Metadata

Group: com.example

Artifact: demo

Name: demo

Description: Demo project for Spring Boot

Package name: com.example.demo

Packaging: ☒ Jar ☐ War

Configuration: ☒ Properties ☐ YAML

Java: ☐ 25 ☐ 21 ☒ 17

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools **DEVELOPER TOOLS**
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Spring Web **WEB**
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

GENERATE CTRL + G **EXPLORE CTRL + SPACE** ...

Através dele definimos nome do projeto, pacotes, dependências (starters do spring e outros projetos), linguagem (Java, Groovy ou Kotlin). Uma vez definido é só clicar no botão *Generate* e o projeto será criado, gerando um zip pronto para ser importado na IDE de sua preferência.

Escolha essas opções e clique em ‘Generate’ para baixar o pacote do projeto pronto.

Site do SpringBoot:

<https://spring.io/projects>

<https://spring.io/>

<https://spring.io/quickstart>

Exercício treino:

Hello world springboot, vamos praticar o nosso primeiro exemplo!

Gere o projeto inicial em: <https://start.spring.io/>

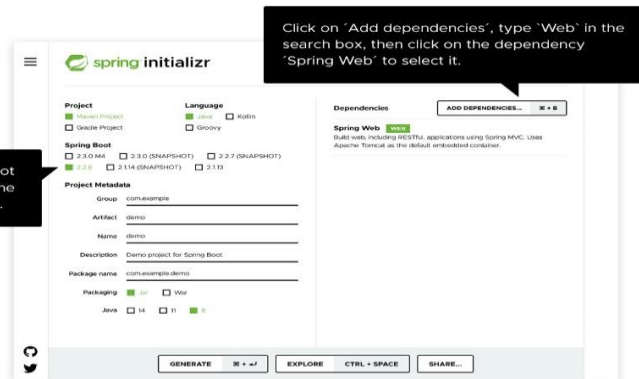
Acesse as informações em: <https://spring.io/quickstart> e preencha o resto do código que precisa para imprimir uma mensagem Hello World! (próximo slide)

Etapa 1: iniciar um novo projeto Spring Boot

Usar start.spring.io para criar um projeto “web”. Na caixa de diálogo “Dependencies”, procure e adicione a dependência “web” conforme mostrado na captura de tela. Clique no botão “Gerar”, baixe o zip e descompacte-o em uma pasta no seu computador.

Atenção! Quando for rodar programas com SpringBoot que utiliza o Apache Tomcat como servidorweb desabilitar o servidor apache do pacote xampp se estiver ativado para não dar conflitos na porta 8080.

The current version of Spring Boot changes regularly. Just choose the latest release (but not snapshot).



Etapa 2: adicione seu código

Abra o projeto em seu IDE e localize o `DemoApplication.java` arquivo na `src/main/java/com/example/demo` pasta.

Agora altere o conteúdo do arquivo adicionando o método extra e as anotações mostradas no código abaixo.

Você pode copiar e colar o código ou apenas digitá-lo.

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

    @GetMapping("/hello")
    public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
        return String.format("Hello %s!", name);
    }
}
```

CÓPIA DE

O `hello()` método que adicionamos foi projetado para receber um parâmetro `String` chamado `name` e, em seguida, combinar esse parâmetro com a palavra "Hello" no código. Isso significa que, se você definir seu nome "Amy" na solicitação, a resposta será "Hello Amy". A `@RestController` anotação informa ao Spring que este código descreve um endpoint que deve ser disponibilizado na web. O `@GetMapping("/hello")` diz ao Spring para usar nosso `hello()` método para responder a solicitações que são enviadas para o `http://localhost:8080/hello` endereço. Finalmente, `@RequestParam` está dizendo ao Spring para esperar um `name` valor na solicitação, mas se não estiver lá, ele usará a palavra "World" por padrão.

Passo 3: Experimente

Vamos construir e executar o programa. Abra uma linha de comando (ou terminal) e navegue até a pasta onde você tem os arquivos do projeto. Podemos construir e executar o aplicativo emitindo o seguinte comando:

Mac OS/Linux:

```
./mvnw spring-boot:run
```

Janelas:

```
mvnw spring-boot:run
```

Obs: **Podemos executar o aplicativo direto no Eclipse** dando um run (botão de executar), a mesma tela do próximo slide irá aparecer no console mostrando se deu tudo certo. Porém sempre devemos rodar em um local ou em outro, pois se rodar os dois ao mesmo tempo irá dar problemas nas requisições.

É mais fácil executar pela interface da IDE escolhida

Você deve ver uma saída muito semelhante a esta:

```

demo ./mvnw spring-boot:run --quiet

      ____
     / ___/
    /  /_  /
   /   /  /
  /_____/

:: Spring Boot ::                (v2.2.4.RELEASE)

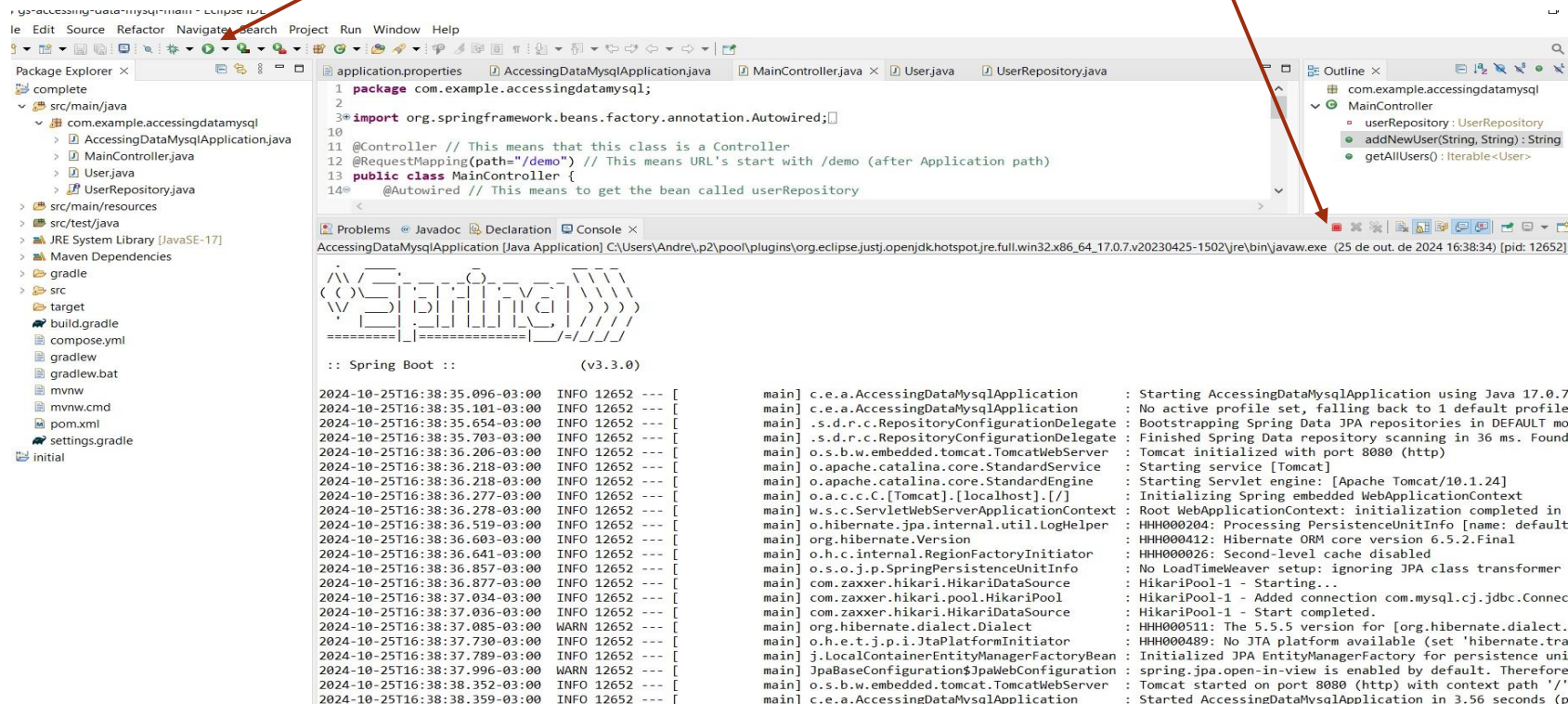
2020-02-14 16:16:47.746 INFO 4838 --- [           main] com.example.demo.DemoApplication : Starting DemoApplication
on Brians-MacBook-Pro.local with PID 4838 (/Users/bclozel/workspace/tmp/demo/target/classes started by bclozel in /Users/bclozel/workspace/tmp/demo)
2020-02-14 16:16:47.748 INFO 4838 --- [           main] com.example.demo.DemoApplication : No active profile set, falling back to default profiles: default
2020-02-14 16:16:48.272 INFO 4838 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-02-14 16:16:48.279 INFO 4838 --- [           main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-02-14 16:16:48.279 INFO 4838 --- [           main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.30]
2020-02-14 16:16:48.323 INFO 4838 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-02-14 16:16:48.324 INFO 4838 --- [           main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 532 ms
2020-02-14 16:16:48.438 INFO 4838 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2020-02-14 16:16:48.533 INFO 4838 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s) 8080 (http) with context path ''
2020-02-14 16:16:48.535 INFO 4838 --- [           main] com.example.demo.DemoApplication : Started DemoApplication in 1.006 seconds (JVM running for 1.248)

```

As últimas linhas aqui nos dizem que a spring começou. O servidor Apache Tomcat incorporado do Spring Boot está agindo como um servidor web e está escutando solicitações na [localhost](http://localhost:8080) porta 8080.

Podemos executar diretamente utilizando o **Eclipse**, IntelliJ ou VScode.

Clicar em cima do Projeto: Run as Java Application. Para parar o projeto clicar em 'stop'.



```
package com.example.accessingdatamysql;

import org.springframework.beans.factory.annotation.Autowired;

@Controller // This means that this class is a Controller
@RequestMapping(path="/demo") // This means URL's start with /demo (after Application path)
public class MainController {

    @Autowired // This means to get the bean called userRepository

    @RequestMapping("/addNewUser")
    public String addNewUser(String name, String email) {
        userRepository.save(new User(name, email));
        return "redirect:/demo/users?msg=added";
    }

    @RequestMapping("/getAllUsers")
    public String getAllUsers() {
        List<User> users = userRepository.findAll();
        return "redirect:/demo/users?msg=loaded";
    }
}
```

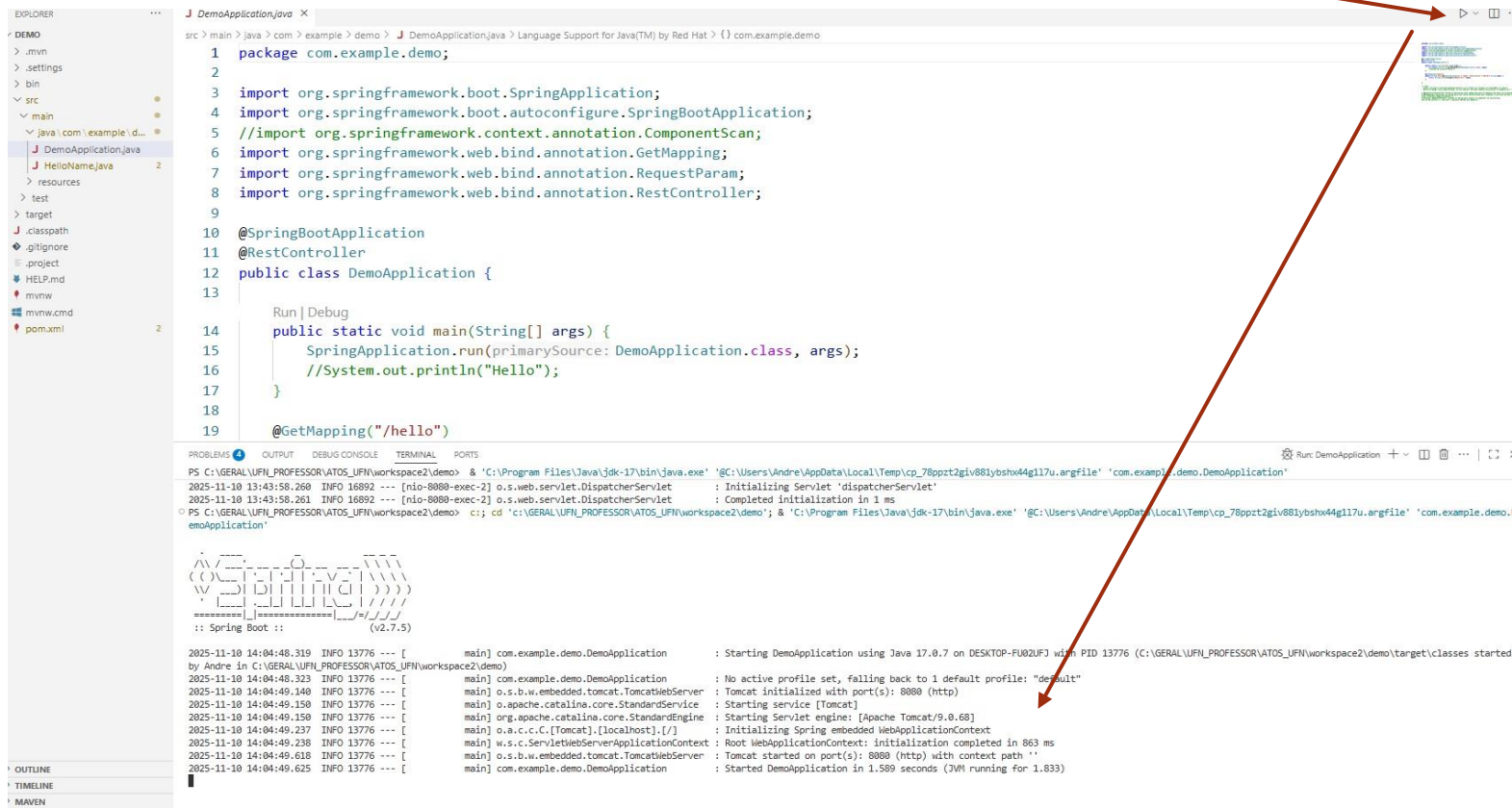
```
AccessingDataMysqlApplication [Java Application] C:\Users\Andre\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.7.v20230425-1502\jre\bin\javaw.exe (25 de out. de 2024 16:38:34) [pid: 12652]

:: Spring Boot :: (v3.3.0)

2024-10-25T16:38:35.096-03:00 INFO 12652 --- [main] c.e.a.AccessingDataMysqlApplication : Starting AccessingDataMysqlApplication using Java 17.0.7
2024-10-25T16:38:35.101-03:00 INFO 12652 --- [main] c.e.a.AccessingDataMysqlApplication : No active profile set, falling back to 1 default profile
2024-10-25T16:38:35.654-03:00 INFO 12652 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode
2024-10-25T16:38:35.703-03:00 INFO 12652 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 36 ms. Found 0 repository beans
2024-10-25T16:38:36.206-03:00 INFO 12652 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2024-10-25T16:38:36.218-03:00 INFO 12652 --- [main] o.apache.catalina.core.StandardEngine : Starting service [Tomcat]
2024-10-25T16:38:36.218-03:00 INFO 12652 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.24]
2024-10-25T16:38:36.277-03:00 INFO 12652 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2024-10-25T16:38:36.278-03:00 INFO 12652 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1.123 s
2024-10-25T16:38:36.519-03:00 INFO 12652 --- [main] o.hibernate.jpa.internal.util.LogHelper : HH000204: Processing PersistenceUnitInfo [name: default]
2024-10-25T16:38:36.603-03:00 INFO 12652 --- [main] org.hibernate.Version : HH000412: Hibernate ORM core version 6.5.2.Final
2024-10-25T16:38:36.641-03:00 INFO 12652 --- [main] o.h.c.internal.RegionFactoryInitiator : HH000026: Second-level cache disabled
2024-10-25T16:38:36.857-03:00 INFO 12652 --- [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2024-10-25T16:38:36.877-03:00 INFO 12652 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-10-25T16:38:37.034-03:00 INFO 12652 --- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.Connection
2024-10-25T16:38:37.036-03:00 INFO 12652 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-10-25T16:38:37.085-03:00 WARN 12652 --- [main] org.hibernate.dialect.Dialect : HH000511: The 5.5 version for [org.hibernate.dialect.Hibernate5Dialect] is not supported
2024-10-25T16:38:37.730-03:00 INFO 12652 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable)
2024-10-25T16:38:37.789-03:00 INFO 12652 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-10-25T16:38:37.996-03:00 WARN 12652 --- [main] jpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore it can't be disabled. Consider setting: 'spring.jpa.open-in-view=false'
2024-10-25T16:38:38.352-03:00 INFO 12652 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
2024-10-25T16:38:38.359-03:00 INFO 12652 --- [main] c.e.a.AccessingDataMysqlApplication : Started AccessingDataMysqlApplication in 3.56 seconds (p
```

Se não houver nenhum erro é acessar os 'end points' de alguma forma

No Vscode instalando a biblioteca SpringBoot Extension Pack. Após executar



The screenshot shows a VS Code editor with a Java project named 'DemoApplication'. The code is a simple Spring Boot application with a REST controller. The terminal output shows the application running successfully on port 8080.

```
src > main > java > com > example > demo > J DemoApplication.java > Language Support for Java(TM) by Red Hat > {} com.example.demo

1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 //import org.springframework.context.annotation.ComponentScan;
6 import org.springframework.web.bind.annotation.GetMapping;
7 import org.springframework.web.bind.annotation.RequestMapping;
8 import org.springframework.web.bind.annotation.RestController;
9
10 @SpringBootApplication
11 @RestController
12 public class DemoApplication {
13
14     public static void main(String[] args) {
15         SpringApplication.run(primarySource: DemoApplication.class, args);
16         //System.out.println("Hello");
17     }
18
19     @GetMapping("/hello")
20 }
```

Run | Debug

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\GERAL\UFN_PROFESSOR\ATOS_UFN\workspace2\demo> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '%C:\Users\Andre\AppData\Local\Temp\c78prrt2g1v881ybsbx44g117u.argfile' 'com.example.demo.DemoApplication'

2025-11-10 13:43:58.260 INFO 16892 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'

2025-11-10 13:43:58.261 INFO 16892 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms

PS C:\GERAL\UFN_PROFESSOR\ATOS_UFN\workspace2\demo> cd 'C:\Users\Andre\AppData\Local\Temp\c78prrt2g1v881ybsbx44g117u.argfile' 'com.example.demo.DemoApplication'

Spring Boot (v2.7.5)

2025-11-10 14:04:48.319 INFO 13776 --- [main] com.example.demo.DemoApplication : Starting DemoApplication using Java 17.0.7 on DESKTOP-FU02UFJ with PID 13776 (C:\GERAL\UFN_PROFESSOR\ATOS_UFN\workspace2\demo\target\classes started by Andre in C:\GERAL\UFN_PROFESSOR\ATOS_UFN\workspace2\demo)

2025-11-10 14:04:48.323 INFO 13776 --- [main] com.example.demo.DemoApplication : No active profile set, falling back to 1 default profile: "default"

2025-11-10 14:04:49.140 INFO 13776 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)

2025-11-10 14:04:49.150 INFO 13776 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]

2025-11-10 14:04:49.150 INFO 13776 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.60]

2025-11-10 14:04:49.237 INFO 13776 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext

2025-11-10 14:04:49.238 INFO 13776 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 863 ms

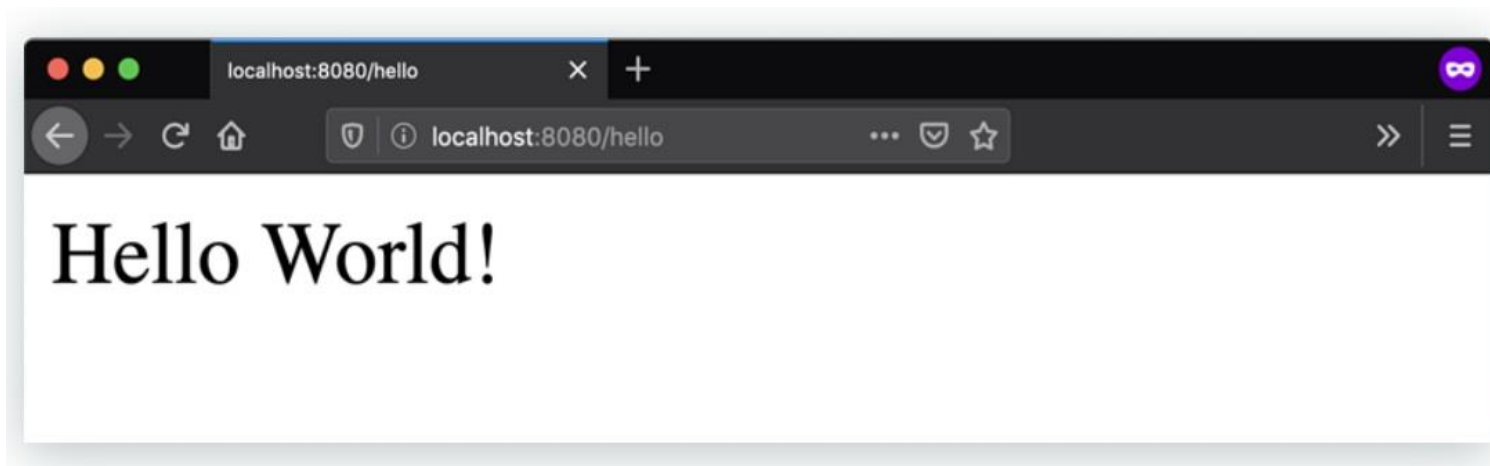
2025-11-10 14:04:49.618 INFO 13776 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''

2025-11-10 14:04:49.625 INFO 13776 --- [main] com.example.demo.DemoApplication : Started DemoApplication in 1.589 seconds (JVM running for 1.833)

Se não houver nenhum erro é só acessar os 'end points' de alguma forma

Após executar o projeto na IDE que você escolheu, podemos testar.

Abra seu navegador e na barra de endereços na parte superior, digite <http://localhost:8080/hello>. Você deve obter uma boa resposta amigável como esta:

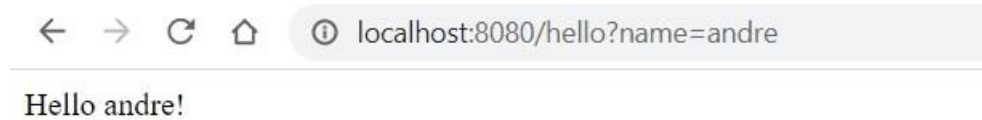


Agora vamos mandar um nome para trocar pela palavra padrão 'world'

```
-
3+ import org.springframework.boot.SpringApplication;
9
0 @SpringBootApplication
1 @RestController
2 public class DemoApplication {
3
4+     public static void main(String[] args) {
5         SpringApplication.run(DemoApplication.class, args);
6         //System.out.println("Hello");
7     }
8
9+     @GetMapping("/hello")
0     public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
1         return String.format("Hello %s!", name);
2     }
3
4 }
-
```

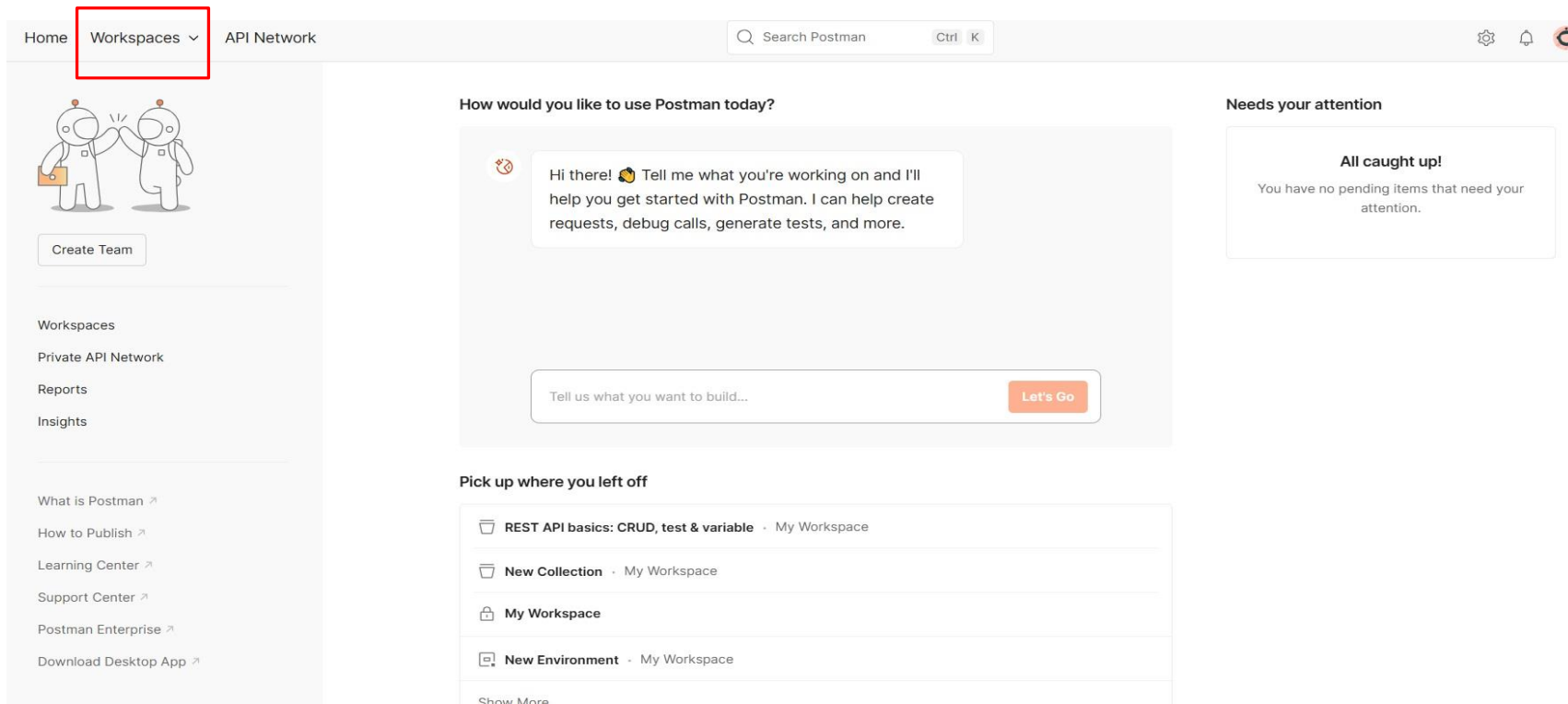
[Digitar http://localhost:8080/hello?name=Andre](http://localhost:8080/hello?name=Andre)

A resposta foi:



Digitar <http://localhost:8080/hello?name=Andre>

Vamos usar a ferramenta Postman: <https://web.postman.co/>. Acessar o site, criar um cadastro e baixar o Postman desktop para testar com EndPoints.



The screenshot shows the Postman web interface. At the top, there is a navigation bar with 'Home', 'Workspaces' (highlighted with a red box), and 'API Network'. To the right of the navigation bar is a search bar labeled 'Search Postman' and a keyboard shortcut 'Ctrl K'. On the left side, there is a sidebar with a 'Create Team' button and a list of links: 'Workspaces', 'Private API Network', 'Reports', 'Insights', 'What is Postman', 'How to Publish', 'Learning Center', 'Support Center', 'Postman Enterprise', and 'Download Desktop App'. The main content area is divided into three sections. The first section, 'How would you like to use Postman today?', features a friendly message from Postman and a 'Let's Go' button. The second section, 'Needs your attention', shows a message that all pending items have been handled. The third section, 'Pick up where you left off', lists recent workspaces and collections, including 'REST API basics: CRUD, test & variable', 'New Collection', 'My Workspace', and 'New Environment'.

Home Workspaces API Network

Search Postman Ctrl K

How would you like to use Postman today?

Hi there! 🙌 Tell me what you're working on and I'll help you get started with Postman. I can help create requests, debug calls, generate tests, and more.

Tell us what you want to build... Let's Go

Needs your attention

All caught up!

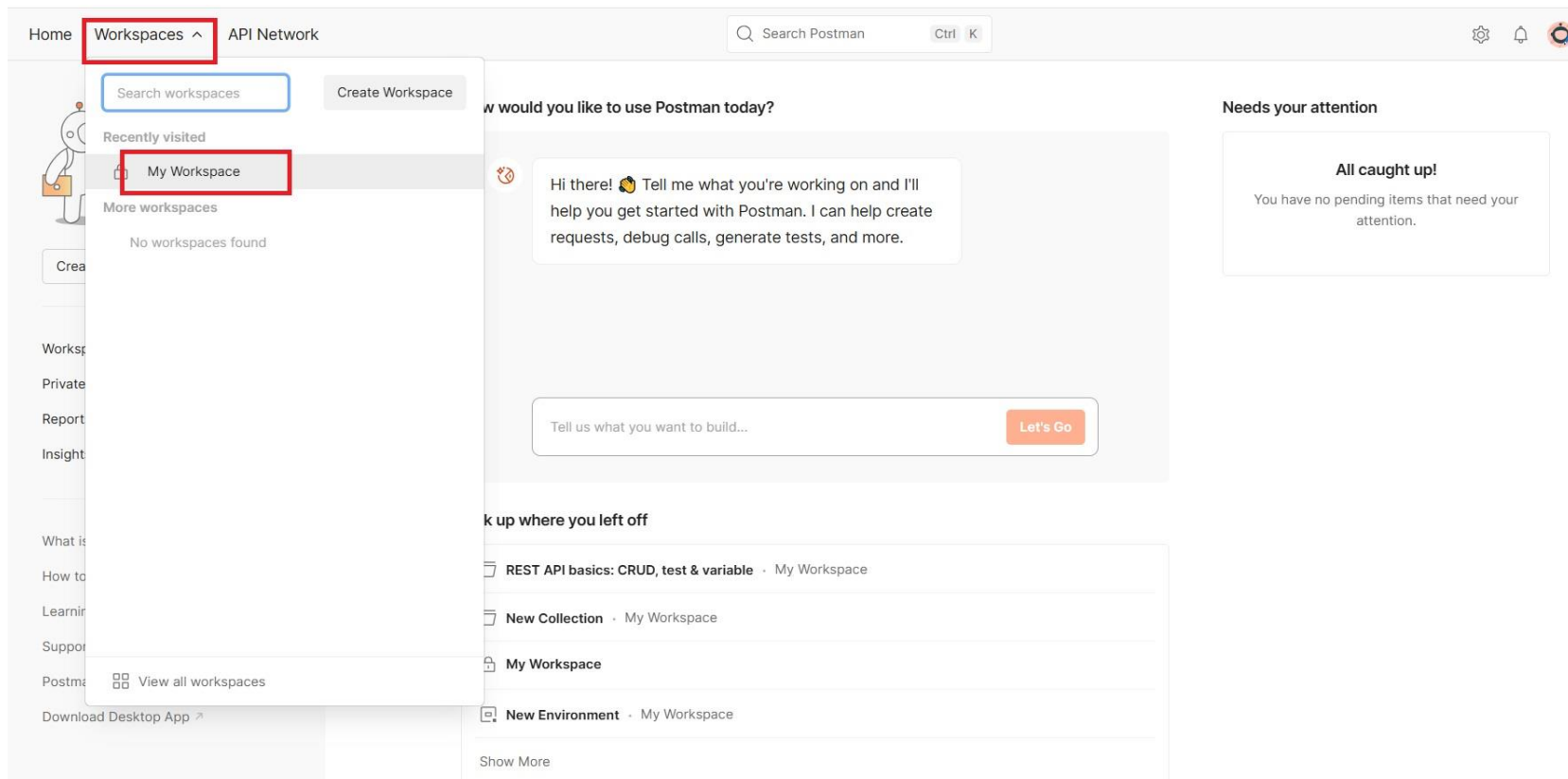
You have no pending items that need your attention.

Pick up where you left off

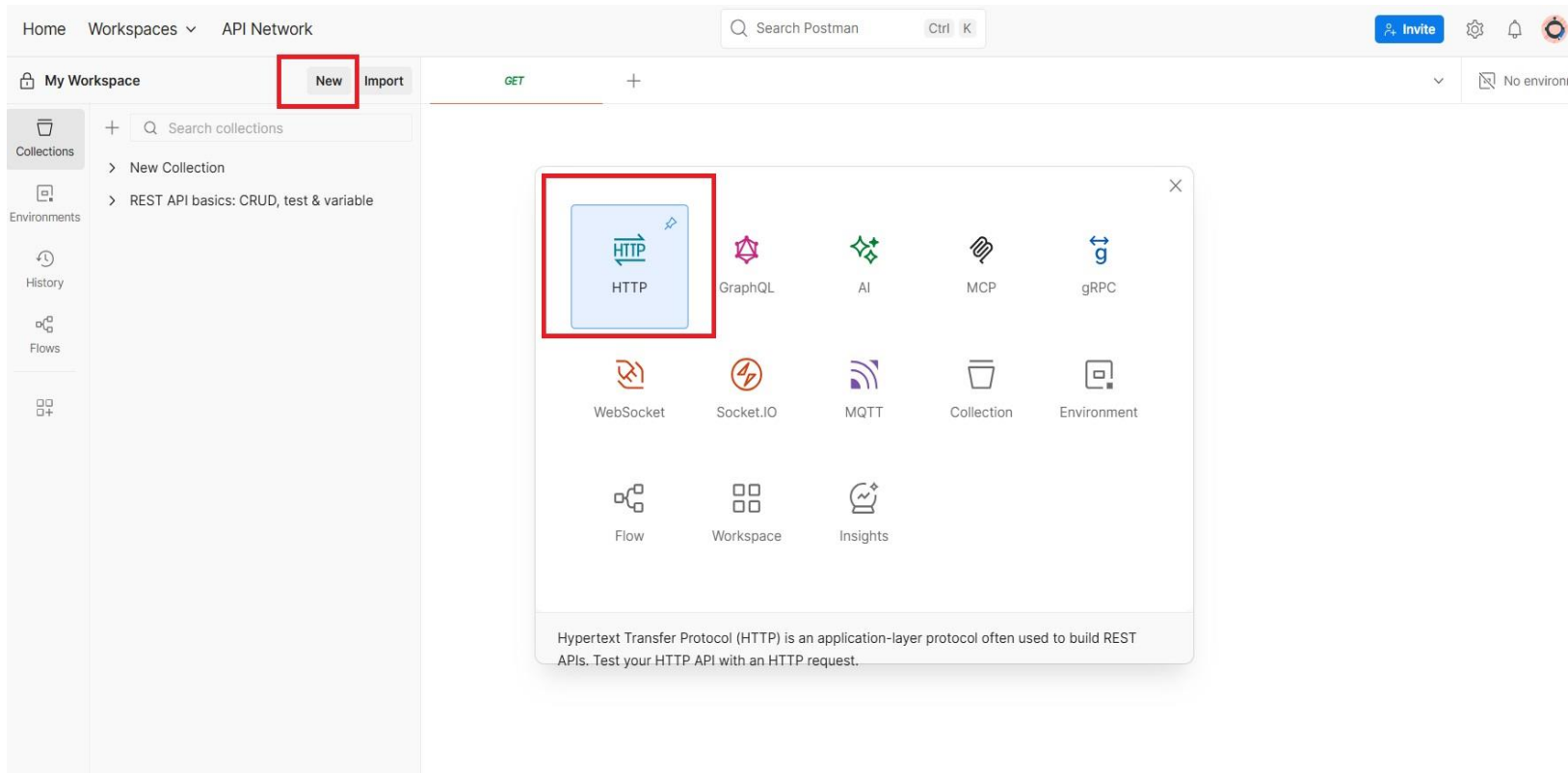
- REST API basics: CRUD, test & variable - My Workspace
- New Collection - My Workspace
- My Workspace
- New Environment - My Workspace

Show More

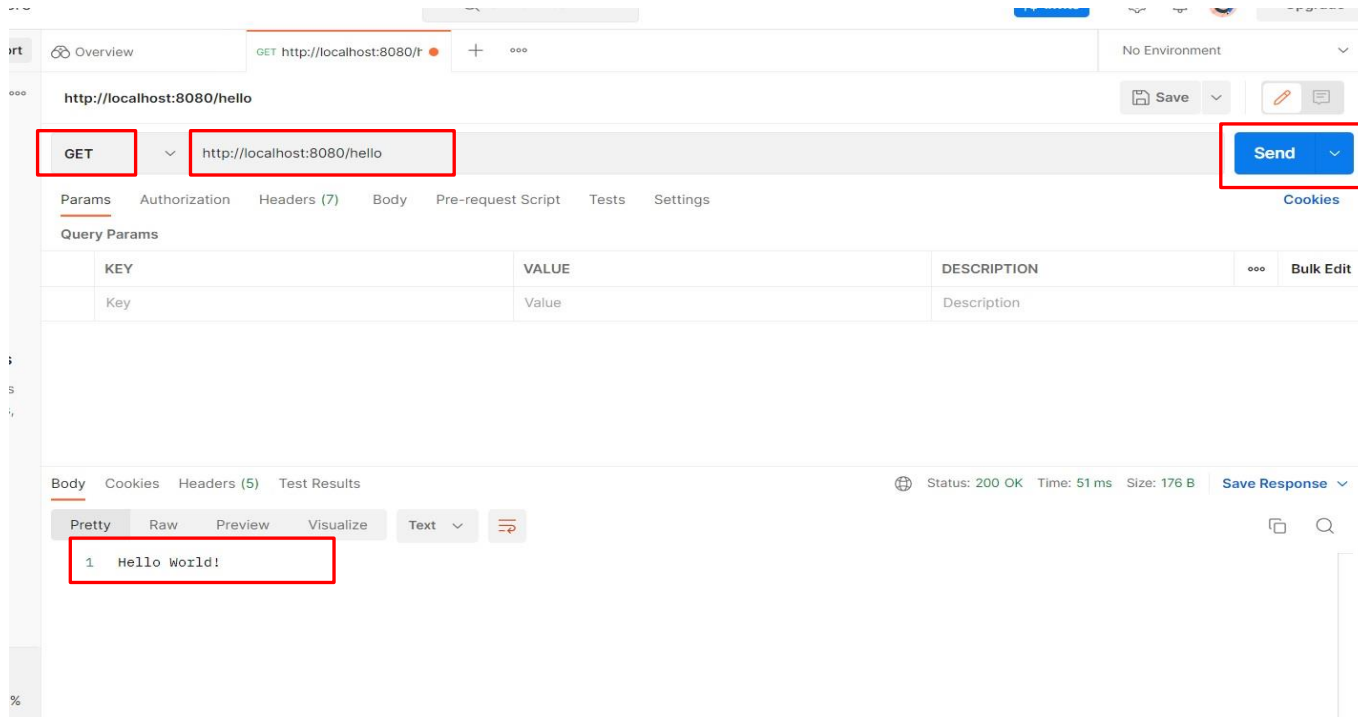
Vamos usar a ferramenta Postman: <https://web.postman.co/>. Acessar o site, criar um cadastro e baixar o Postman desktop para testar com EndPoints.



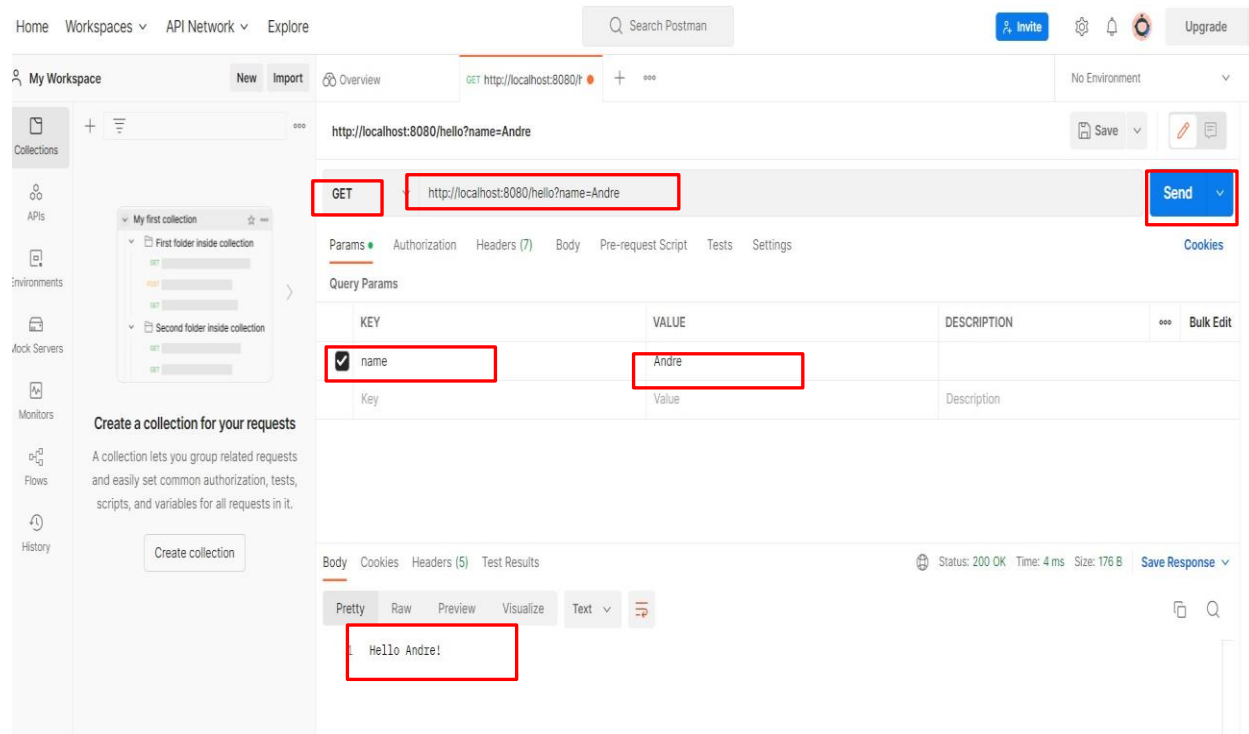
Vamos usar a ferramenta Postman: <https://web.postman.co/>. Acessar o site, criar um cadastro e baixar o Postman desktop para testar com EndPoints.



Vamos usar a ferramenta Postman: <https://web.postman.co/>. Acessar o site, criar um cadastro e baixar o Postman desktop para testar com EndPoints. Se for utilizar via web e solicitar para baixar o Postman Agent, instale ele para poder usar. Ou faça download do Postman Desktop que será executado direto na máquina, sem problemas.



Testando o envio do nome 'Andre' ou o desejado junto.



The screenshot shows the Postman application interface. At the top, there are navigation tabs: Home, Workspaces, API Network, and Explore. A search bar labeled 'Search Postman' is on the right. Below the navigation bar, the 'My Workspace' section is active, showing a list of collections on the left sidebar. The main area displays a request configuration for a GET method to the URL 'http://localhost:8080/hello?name=Andre'. The 'GET' method and the URL are highlighted with red boxes. A 'Send' button is also highlighted with a red box. Below the URL bar, the 'Query Params' tab is selected, showing a table with one parameter: 'name' with the value 'Andre'. The 'name' checkbox and the value 'Andre' are highlighted with red boxes. At the bottom, the 'Body' tab is selected, showing the response 'Hello Andre!' in the 'Text' view, which is also highlighted with a red box. The status bar at the bottom right indicates 'Status: 200 OK', 'Time: 4 ms', and 'Size: 176 B'.

Home Workspaces API Network Explore

Search Postman

Invite Upgrade

My Workspace New Import Overview GET http://localhost:8080/hello?name=Andre No Environment

Save

GET http://localhost:8080/hello?name=Andre Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
<input checked="" type="checkbox"/> name	Andre		
Key	Value	Description	

Create a collection for your requests

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

Create collection

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 4 ms Size: 176 B Save Response

Pretty Raw Preview Visualize Text

Hello Andre!

1. O que são métodos HTTP

- Quando um cliente (navegador, Postman, app) fala com o servidor, ele **usa um “método”** para dizer **o que quer fazer**.
- Os mais usados são:
 - **GET** → pedir informações (ler dados)
 - **POST** → enviar informações (criar ou enviar dados)
 - **PUT** → atualizar algo
 - **DELETE** → excluir algo

2. GET — Leitura de dados

```
@GetMapping("/hello")  
public String hello(@RequestParam String name) {  
    return "Hello, " + name;  
}
```

O método `@GetMapping`:

Usa **GET** → leitura.

Espera os dados **na URL**, por exemplo:

2. GET — Leitura de dados

```
@GetMapping("/hello")  
public String hello(@RequestParam String name) {  
    return "Hello, " + name;  
}
```

O método `@GetMapping`:

Usa **GET** → leitura.

Espera os dados **na URL**, por exemplo:

`http://localhost:8080/hello?name=Andre`

O navegador e o Postman enviam os parâmetros como query string.

⚠ Os dados aparecem na URL, por isso não é seguro para senhas ou informações sensíveis.

3. POST — Envio de dados

```
@PostMapping("/hello")
public String hello(@RequestBody Map<String, String> body) {
    return "Hello, " + body.get("name");
}
```

O método `@PostMapping`:

- Usa **POST** → envio.
- Os dados vão **no corpo (body)** da requisição.
- No Postman: **Body** → **raw** → **JSON**
- { "name": "Andre" }
- Mais seguro (dados não aparecem na URL).
- Usado para **criar ou enviar** dados ao servidor.

Métodos HTTP no Spring Boot

Ação	Método	Envio	Segurança	Uso típico
Ler dados	GET	Parâmetros na URL (?name=...)	Aparece na URL	Consultas
Enviar dados	POST	Corpo (JSON, form)	Mais seguro	Envio/criação

5. No Postman

- Se o controller usa `@GetMapping` → selecione **GET** e use **Params**.
- Se usa `@PostMapping` → selecione **POST** e coloque o **JSON no Body**.
- O método HTTP precisa **bater com a anotação** do código.

Resumo final

- `@GetMapping` → consulta, dados simples, URL visível.
- `@PostMapping` → envio, dados estruturados, corpo da requisição.
- É o **Spring Boot** (anotação) que define **qual tipo de requisição** seu endpoint espera.

Tarefa para casa, testar todo esse processo novamente!!

Agora vamos para a parte II.

Referências

HOGAN, Brian P. HTML 5 e CSS3: Desenvolva hoje com o padrão de amanhã . Rio de Janeiro (RJ): Ciência Moderna Ltda., 2012 282 p. ISBN 978-85-399-0260-6

Fábio Flatschart. HTML 5 - Embarque Imediato, 2011. (Biblioteca Digital)

Deitel, Paul J.; Deitel, Harvey M.. Ajax, Rich Internet Applications e Desenvolvimento Web para Programadores, 2008. (Biblioteca Digital)

SILVA, Maurício Samy. HTML 5: a linguagem de marcação que revolucionou a web. 2. ed. São Paulo, SP: Novatec, 2014. 335 p. ISBN 978-85-7522-403-8.

Denilson Bonatti. Desenvolvimento de Jogos em HTML5, 2014. (Biblioteca Digital)

W3SCHOOL. The world's largest web development site. Acessado em 2018. Disponível em: <http://www.w3schools.com/>.

W3C. World Wide Web Consortium. Acessado em 2018. Disponível em: <http://www.w3.org>

MORRISON, Michael. Use a cabeça JavaScript. Rio de Janeiro (RJ): Alta Books, 2008. 606 p. <https://br.freepik.com/fotos-vetores-gratis/desenvolvimento-web>

Material do Professor Fabrício Tonetto Londero, 2023.

Obrigado pela atenção!!



Email: andre.flores@ufn.edu.br

Santa Maria – RS
2025