

Desenvolvimento de Software para Web



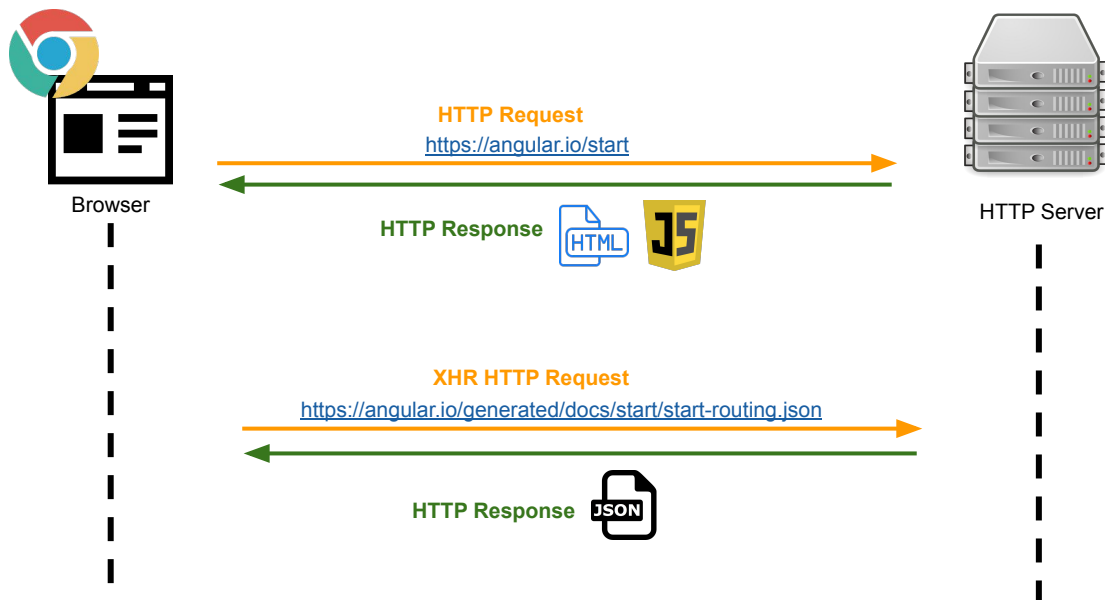
UFC - Universidade Federal do Ceará

André Meireles
andre@crateus.ufc.br

SPA - Rest com Spring Boot & AJAX

Arquitetura Web Single-page App (SPA)

- A experiência do usuário é dirigida pela troca de documentos JSON (site exemplo: angular.io)
- O Javascript processa a resposta do servidor e atualiza os componentes do DOM



Rest + JSON

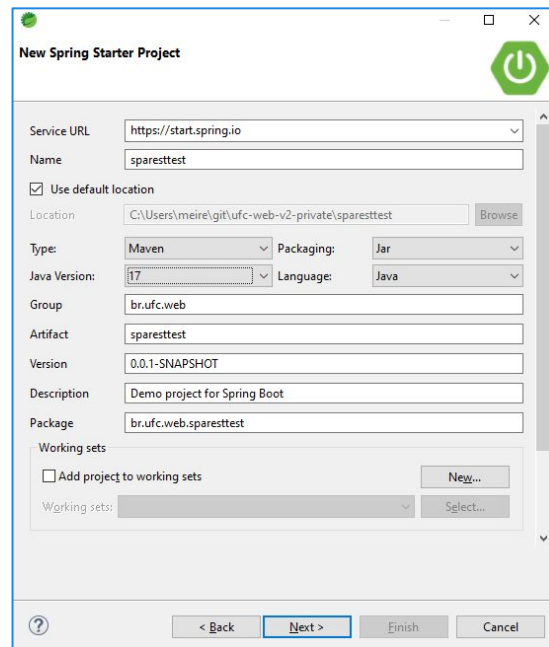
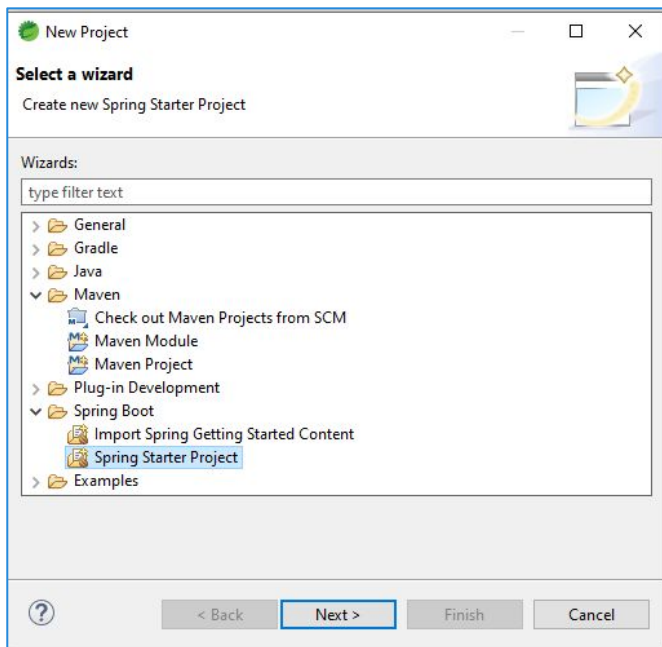
Estude as aulas abaixo antes de prosseguir:

[Aula sobre Rest](#)

[Aula sobre JSON](#)

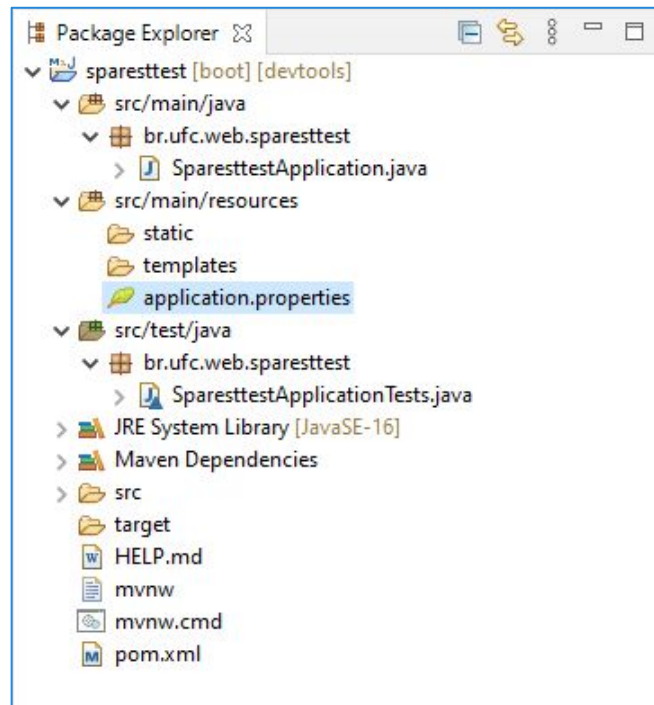
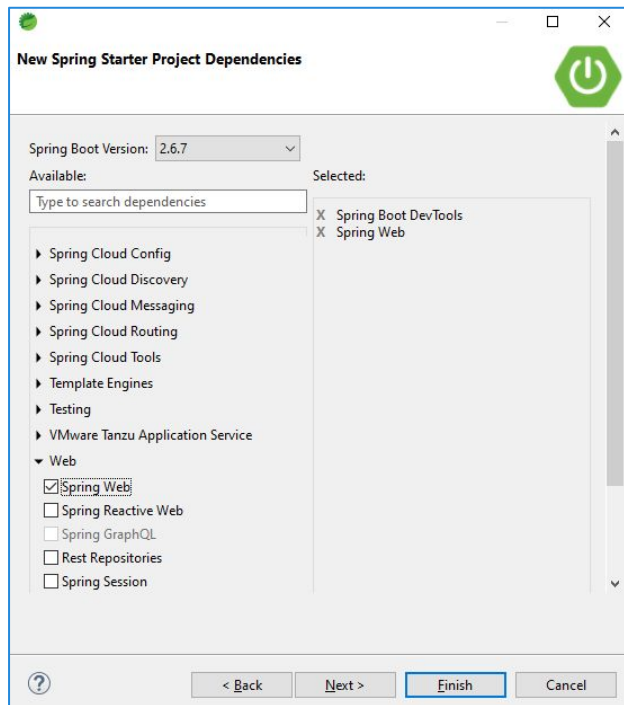
Rest com Spring Boot

Utilizando o [Spring Tool Suite](#) para Eclipse, crie o novo projeto Spring Starter:



Rest com Spring Boot

Escolha os pacotes mínimos necessários (print da esquerda) abaixo e finalize.



Arquivo pom.xml do projeto Spring

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.7</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>br.ufc.web</groupId>
  <artifactId>sparesttest</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>sparesttest</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>16</java.version>
  </properties>
```

Arquivo pom.xml do projeto Spring

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

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```


Arquivo pom.xml do projeto Spring

```
<build>  
  <plugins>  
    <plugin>  
      <groupId>org.springframework.boot </groupId>  
      <artifactId>spring-boot-maven-plugin </artifactId>  
    </plugin>  
  </plugins>  
</build>  
  
</project>
```

Rest com Spring Boot

Execute o projeto usando o **Boot Dashboard** ou execute a classe principal como Java Application. Você deverá ter uma saída no console similar a que é apresentada abaixo:



```
sparesttest - SparesttestApplication [Spring Boot App] C:\Users\meire\sts-4.11.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64.16.0.2.v20210721-1149\jre\bin\java
14:11:18.961 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.spr

:: Spring Boot :: (v2.6.7)

2022-05-16 14:11:19.274 INFO 696 --- [ restartedMain] b.u.w.s.SparesttestApplication : Starting SparesttestApplication using Jav
2022-05-16 14:11:19.276 INFO 696 --- [ restartedMain] b.u.w.s.SparesttestApplication : No active profile set, falling back to 1
2022-05-16 14:11:19.316 INFO 696 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 's
2022-05-16 14:11:19.316 INFO 696 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consic
2022-05-16 14:11:20.068 INFO 696 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (ht
2022-05-16 14:11:20.079 INFO 696 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-05-16 14:11:20.079 INFO 696 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/s
2022-05-16 14:11:20.153 INFO 696 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicati
2022-05-16 14:11:20.153 INFO 696 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initializati
2022-05-16 14:11:20.502 INFO 696 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 3572
2022-05-16 14:11:20.551 INFO 696 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) wi
2022-05-16 14:11:20.561 INFO 696 --- [ restartedMain] b.u.w.s.SparesttestApplication : Started SparesttestApplication in 1.59 se
2022-05-16 14:12:10.774 INFO 696 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'di
2022-05-16 14:12:10.774 INFO 696 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-05-16 14:12:10.775 INFO 696 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 0 ms
```

Rest com Spring Boot

Acesse <http://localhost:8080/> e verifique se o servidor está funcionando



Rest Controllers

Rest Controllers são utilizados para definir endpoints de uma API. Estes métodos são chamados quando uma requisição HTTP é realizada para o caminho e método definidos

```
@RestController << Declara o controller
public class ProductRestController {

    @GetMapping("/api/produto") << Define o método e o caminho
    List<Produto> getProdutos() {
        return Arrays.asList("produto 1", "produto 2", "produto 3");
    }

    @PostMapping("/api/produto")
    void addProduto(@RequestBody Produto produto) {

    }
}
```

<< Define como o objeto será enviado

Services

Services são componentes reusáveis que normalmente implementam as regras de negócio. Costumam ser invocados pelos Controllers ou por outros Services

@Service << Declara a classe como serviço

```
public class ProdutoService {  
  
    private List<Produto> produtos = new ArrayList<Produto>();  
  
    public ProdutoService() {  
        produtos.addAll(Arrays.asList(new Produto(1, "Livro", 200f),  
new Produto(1, "Laptop", 938752.67f), new Produto(3, "Bicicleta", 12400f)));  
    }  
  
    public List<Produto> getProdutos() {  
        return produtos;  
    }  
  
    public void addProduto(Produto produto) {  
        produtos.add(produto);  
    }  
  
}
```

@RestController

```
public class ProductRestController {
```

@Autowired << Instancia o serviço no controller

```
    ProdutoService produtoService;
```

@GetMapping("/api/produto")

```
List<String> getProdutos(){  
    return produtoService.getProdutos();  
}
```

@PostMapping("/api/produto")

```
void addProduto(@RequestBody Produto produto) {  
    produtoService.addProduto(produto);  
}
```

```
}
```

Rest Controller

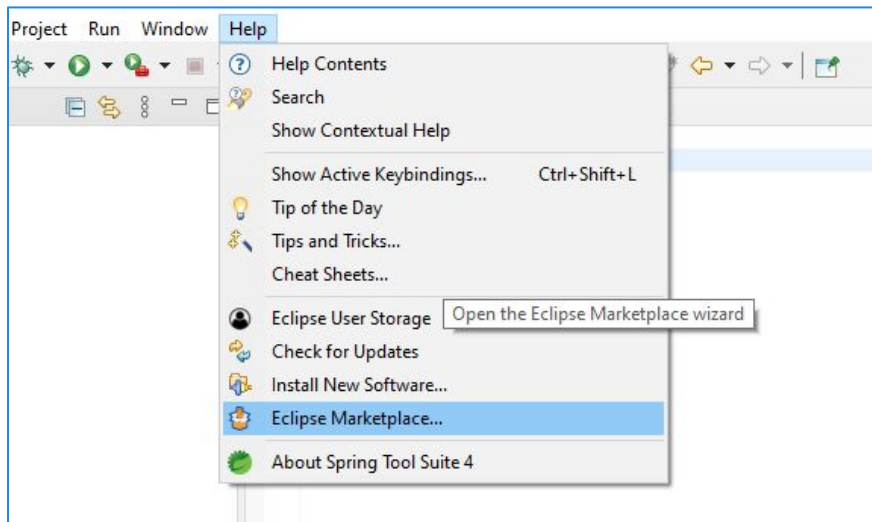
Também é possível definir métodos como PUT e DELETE e receber dados através de parâmetros do path ou parâmetros de consulta, veja abaixo uma exemplo:

```
@GetMapping("/api/product/{id}")
public Product getProductById(@PathVariable int id) {
    return productService.getProductById(id);
}

@PutMapping("/api/product/{id}")
public Product updateProduct(@PathVariable int id, @RequestBody Product product) {
    return productService.updateProduct(id, product);
}


@DeleteMapping("/api/product/{id}")
public void deleteProduct(@PathVariable int id) {
    if (!productService.deleteProduct(id)) throw new ResponseStatusException(HttpStatus.NOT_FOUND);
}
```

Servindo conteúdo estático com Spring



Você pode instalar plugin para facilitar a implementação com HTML e Javascript:

- Acesse o Eclipse Marketplace
- Encontre o plugin desejado
 - Sugestão: Eclipse WDT



Eclipse Web Developer Tools 3.22
Includes the HTML, CSS, and JSON Editors, and JavaScript Development Tools from the Eclipse Web Tools Platform project, aimed at supporting client-side web... [more info](#)
by [The Eclipse Foundation](#), EPL
[xml](#) [html](#) [CSS](#) [js](#) [JSON](#)

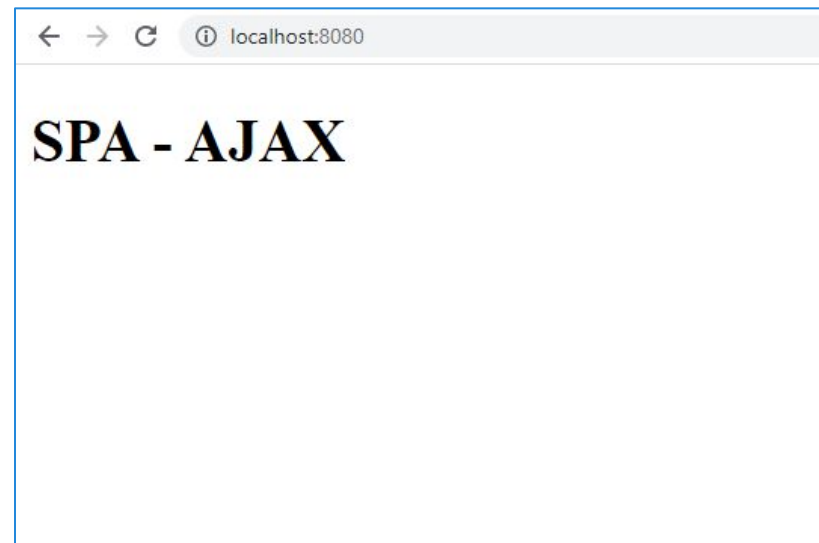
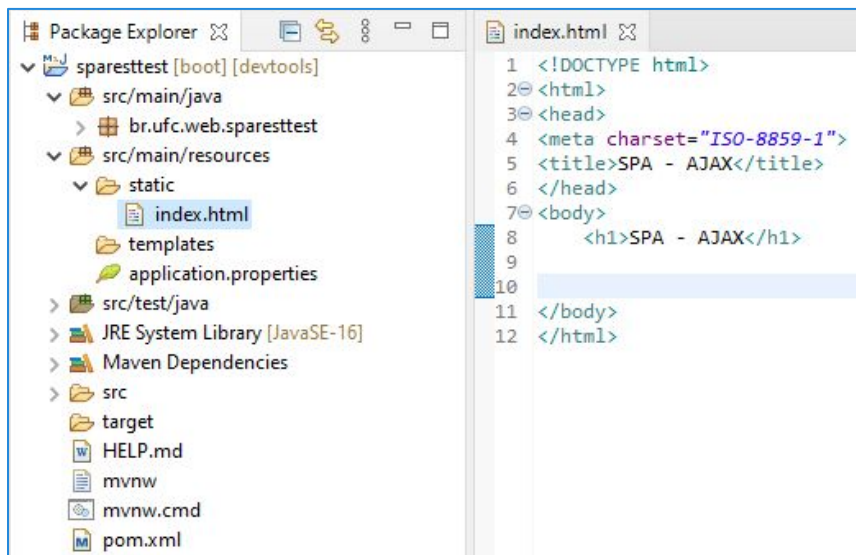
★ 1250

🔄 Installs: **495K** (16,641 last month)

Change

Servindo conteúdo estático com Spring

O diretório padrão para conteúdo estático em aplicações Spring é `src/main/resources/static`



XHR - XML HTTP Request

XHR é utilizado para requisições HTTP assíncronas realizadas pelo Javascript no navegador. O código a seguir consome uma API e altera o conteúdo da página inserindo o conteúdo retornado pelo servidor.

```
<div id="listaProdutos"></div>
<script type="text/javascript">
  var xhttp = new XMLHttpRequest();
  xhttp.open("GET", "/api/product", true);
  xhttp.onload = function() {
    if (this.readyState == 4 && this.status == 200) {
      produtos = JSON.parse(xhttp.responseText);
      produtosDiv = document.getElementById("produtos");

      produtosDiv.innerHTML = JSON.stringify(produtos);

      listaProdutosDiv = document.getElementById("listaProdutos");
      lista = '<ul>';

      for (const i in produtos) {
        let p = produtos[i];
        lista += '<li>' + p.id + ' - ' + p.name + ' - ' + p.price + '</li>';
      }

      lista += '<ul>';
      listaProdutosDiv.innerHTML = lista;
    }
  };
  xhttp.send();
</script>
```

XHR - XML HTTP Request

O código a seguir envia dados de um formulário através de uma API usando POST

```
<h1>SPA - AJAX</h1>
<div>
  Id: <input type="number" id="prodId"> <br/>
  Name: <input type="text" id="prodName"> <br/>
  Price: <input type="number" id="prodPrice"> <br/>
  <input type="button" value="Save" onclick="save()">
</div>
...
```

Continua o no próximo slide...

XHR - XML HTTP Request

O código a seguir envia dados de um formulário através de uma API usando POST

```
<script type="text/javascript">

    function save() {
        let id = document.getElementById("prodId").value;
        let name = document.getElementById("prodName").value;
        let price = document.getElementById("prodPrice").value;
        newprod = {id: parseInt(id), name: name, price: parseFloat(price)};

        var xhr = new XMLHttpRequest();
        xhr.open("POST", '/api/product', true);

        xhr.setRequestHeader("Content-Type", "application/json;charset=UTF-8");

        xhr.send(JSON.stringify(newprod));
    }
    ...

```

Links importantes

- <https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started.html#getting-started.first-application>
- <https://spring.io/guides>
- <https://developer.mozilla.org/pt-BR/docs/Web/API/XMLHttpRequest/send>