

# Web Services - REST

## Sistemas Distribuídos e Mobile

Prof. Me. Gustavo Torres Custódio  
gustavo.custodio@ulife.com.br

# Conteúdo

**REST**

**REST com Netbeans**

**Exercícios**



Web Services - REST

REST

# REST

- Serviços REST possuem uma gramática simples, facilmente compreensível.
- Facilmente escalável.
  - Basta adicionar um novo web service.

# Conceitos de REST

- Resources;
- Representations;
- Operations;
- Hypertext;
- Statelessness.

## Resources

- Todos os recursos recebem um identificador.
- Representados por URLs.
- Exemplo:
  - <http://obeautifulcode.com/game/robots/four-hand-george>

# Representations

- Sistemas RESTful permitem que os clientes perguntem em uma forma que eles possam entender;
  - Exemplo: HTTP Header
    - GET /pages/archiveHTTP/1.1
    - Host: obeautifulcode.com
    - Accept: text/html
- text/html representa o **MIME type**.

# Operations

- REST define 4 operações padrão (invocadas por HTTP).
  - **GET**: recupera algum recurso.
  - **PUT**: cria um recurso ou atualiza um existente.
  - **POST**: criar um recurso, mas deixa o server decidir a URL.
  - **DELETE**: exclui o recurso.



# Hypertext

- No REST, o estado de um aplicativo é transferido e descoberto por meio de mensagens de hipertexto.
- O cliente REST tem menos necessidade de saber como interagir com qualquer serviço.
- Por meio de hipertexto, é possível definir o que os navegadores Web devem fazer.

## Statelessness

- O REST estabelece que o servidor não mantém nenhuma informação sobre o estado da sessão do cliente.
  - A requisição que o cliente faz deve conter toda a informação necessária para entendê-la.
  - O cliente é responsável por mandar informações do estado para o servidor sempre que necessário.

## SOAP X REST

- SOAP é considerado um protocolo, enquanto REST é um estilo arquitetural.
- REST é muito mais simples que SOAP. Criar clients, desenvolver APIs e documentar é muito mais fácil e simples em REST.
- REST permite diferentes formatos para dados (JSON, XML, etc.), enquanto SOAP permite somente XML. JSON por exemplo, é enxuto e tem processamento rápido.

## SOAP X REST

- REST tem uma melhor performance e escalabilidade, além de poder ser cacheado. SOAP não pode ser cacheado.
- Uma das vantagens do SOAP é o suporte à WS-Security que adiciona camadas de segurança extras além das suportadas através de HTTP, SSL, etc. Isso lhe dá algumas vantagens em determinadas finalidades no mundo “Enterprise”.
- SOAP suporta transações ACID (*Atomicity, Consistency, Isolation, Durability*) – mesmo conceito encontrado em transações de banco de dados. REST também suporta



Web Services - REST

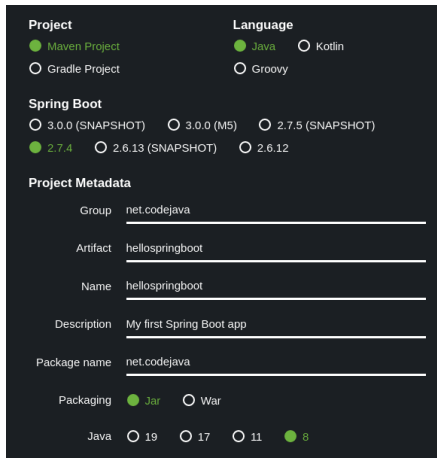
# REST com Netbeans

# REST com Netbeans

- Comece acessando:
  - <https://start.spring.io/>
  - Este site vai ajudar a gerar o projeto inicial que será importado para o Netbeans.

# REST com Netbeans

- Configure o projeto:



The screenshot shows the 'New Project' dialog in NetBeans. The 'Project' section has 'Maven Project' selected. The 'Language' section has 'Java' selected. The 'Spring Boot' section has '2.7.4' selected. The 'Project Metadata' section has the following values: Group (net.codejava), Artifact (hellospringboot), Name (hellospringboot), Description (My first Spring Boot app), Package name (net.codejava), Packaging (Jar), and Java version (8).

**Project**

☒ Maven Project  
☐ Gradle Project

**Language**

☒ Java ☐ Kotlin  
☐ Groovy

**Spring Boot**

☐ 3.0.0 (SNAPSHOT) ☐ 3.0.0 (M5) ☐ 2.7.5 (SNAPSHOT)  
☒ 2.7.4 ☐ 2.6.13 (SNAPSHOT) ☐ 2.6.12

**Project Metadata**

Group

Artifact

Name

Description

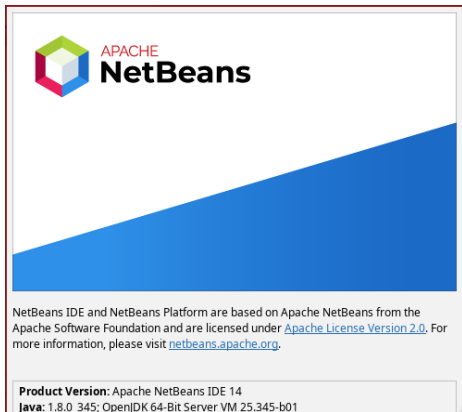
Package name

Packaging ☒ Jar ☐ War

Java ☐ 19 ☐ 17 ☐ 11 ☒ 8

## REST com Netbeans

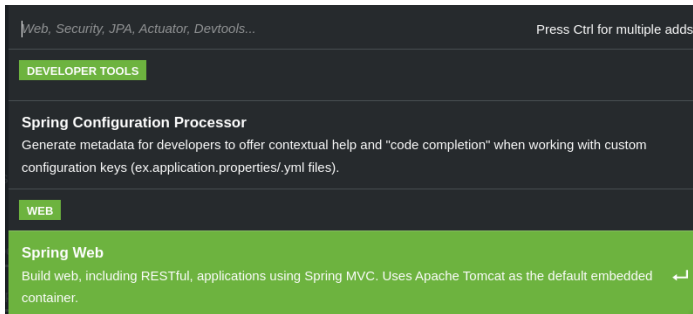
- Escolha a versão do Java correspondente à sua máquina.
  - Para fazer isso no Netbeans, clique em Sobre (about).





## REST com Netbeans

- Adicione a dependência do SpringBoot para o projeto.



- Quando terminar tudo, clique em *Generate* e salve o arquivo.

## REST com Netbeans

- Para importar o arquivo gerado no Netbeans, selecione:
  - Arquivo > Importar Projeto > ZIP
- Se do lado do nome do projeto, a palavra *unloadable* for mostrada, houve um problema na importação do projeto.
  - Clique com o lado direito no projeto e em Propriedades.
  - Clique no botão de resolver problemas no lado direito.
  - Espere o Maven baixar os módulos necessários.

## REST com Netbeans

- Vamos fazer um primeiro exemplo utilizando uma página HTML.
- Na pasta static, em *Other Resources*, crie um arquivo chamado index.html.

```
<html>
  <head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <div>TODO write content</div>
  </body>
</html>
```

## REST com Netbeans

- A aplicação é executada por padrão na porta 8080, então podemos acessá-la pelo navegador:
  - localhost:8080.

## REST com Netbeans

- Agora, no mesmo pacote do arquivo `HellospringbootApplication.java`, adicione um arquivo `HellospringbootController.java`.
- Vamos criar um *endpoint* chamado “hello”.

- Adicionaremos os *endpoints* no arquivo *Controller*.
  - Adicione a anotação `RestController` no topo da classe.

```
@RestController  
public class HellospringbootController {
```

## REST com Netbeans

```
@RequestMapping("/hello")  
public String index() {  
    return "<h1>Olá Mundo!</h1>";  
}
```

- Consumimos a API usando o endereço:
  - localhost:8080/hello

## REST com Netbeans

- Se mudarmos algo, é necessário reiniciar a aplicação.
  - Para não ser necessário reiniciá-la a cada mudança, adicionamos a dependência devtools em pom.xml.

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



# REST com Netbeans

- Anotação PathVariable:
  - Podemos utilizá-la para trabalhar com variáveis de *template*.
  - São passadas como parâmetros de método.

```
@RequestMapping("/cadastro/{nome}")  
public String dizernome(@PathVariable String nome) {  
    return "Olá, meu nome é" + nome;  
}
```

# REST com Netbeans

- Anotação RequestParam:
  - Podemos utilizá-la para extrair parâmetros da *query* da requisição realizada.

```
@RequestMapping("/info")
public String apresentar(@RequestParam String nome,
                        @RequestParam int idade)
{
    return "<h1>Olá pessoal, meu nome é" + nome + " e eu tenho " + idade + "
        anos</h1>";
}
```

- Acesse com localhost:8080/info?nome=<nome>&idade=<idade>

## REST com Netbeans

- Até agora, só fizemos requisições HTTP do tipo GET (*RequestMapping*).
  - Verbo padrão HTTP.
  - Utilizado com mais frequência.

## REST com Netbeans

- Podemos utilizar outros verbos HTTP
  - Como o POST
- Vamos utilizar uma aplicação para nos ajudar a simular requisições de outros tipos além de GET.

## REST com Netbeans

- Postman
- Disponível em:
  - <https://www.postman.com/>.
- Ferramenta utilizada para testar APIs.

# REST com Netbeans

Home Workspaces ▾ API Network ▾ Explore

🔍 Search Postman



Upgrade



## Postman works best with teams

Collaborate in real-time and establish a single source of truth for all API workflows.

Create Team

Workspaces >

Integrations >

Reports >

## Let's burn some midnight oil, Gustavo

Take a look around, get started when you're ready.

### Recently visited workspaces

👤 My Workspace



### Get started with Postman

#### Start with something new

Create a new request, collection, or API in a workspace

[Create New →](#)

#### Import an existing file

Import any API schema file from your local drive or Github

[Import file →](#)

### Product Updates

[Hide](#)



#### Postman v10 is here! NEW

Read about Postman v10 and learn about new features including Native Git, gRPC, Postman CLI, A

[Learn More](#)

### Activity Feed



#### Your team's activity will show up here

Get started by inviting people to your team.

Create Team

Explore popular APIs [Explore all →](#)

## REST com Netbeans

- Vamos criar um *endpoint* para requisições POST
  - Utilizamos a anotação `PostMapping` para atender requisições POST.

```
@PostMapping("/postar")  
public String postar() {  
    return "Objeto postado com sucesso.";  
}
```

- No Postman, verifique a resposta da URL para requisições POST:
  - `localhost:8080/postar`.

# REST com Netbeans

localhost:8080/hello

Save

</>

GET

localhost:8080/hello

Send

Params

Authorization

Headers (6)

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: 12 ms

Size: 184 B

Save Response

Pretty

Raw

Preview

Visualize

Text

1

<h1>Olá Mundo!</h1>





Web Services - REST

# Exercícios

## Exercício 1

- Faça um web service para validar um CPF.
  - Utilize Java e REST.

## Exercício 2

- Faça um Webservice que receba dois números:
  - um número mínimo.
  - um número máximo.
- Sorteie um número inteiro aleatório entre esses dois.

## Referências

- <https://www.infoq.com/minibooks/emag-03-2010-rest>
- <http://blog.obautifulcode.com/API/Learn-REST-In-18-Slides/>
- <http://courses.ischool.berkeley.edu/i290-rmm/s12/slides/Lecture3%20REST.pdf>

## Conteúdo



<https://gustavotcustodio.github.io/sdmobile.html>

Obrigado

[gustavo.custodio@ulife.com.br](mailto:gustavo.custodio@ulife.com.br)