# Java Servlets

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

# AGENDA

- Rotas
- Métodos HTTP
- Response
- Parâmetros de URL e Body
- Adicionando dependência externa

# Mapeando rotas

Ao criar um servlet, precisamos mapear a classe para que responda à URL. Podemos fazer esse mapeamento de duas formas:

1) Mapeando pelo web.xml:

```
<servlet>
    <servlet-name>comingsoon</servlet-name>
    <servlet-class>mysite.server.ComingSoonServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>comingsoon</servlet-name>
    <url-pattern>/*</url-pattern>
</servlet-mapping>
```

2) Mapeando pela anotação “@WebServlet” na classe:

```
/**
 * Servlet implementation class FirstServlet
 */
@WebServlet("/FirstServlet/*")
public class FirstServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
```

# Métodos HTTP

Após mapear, podemos declarar os métodos seguindo o prefixo: “do” + método, como por exemplo:

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
protected void doDelete(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
protected void doPut(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
```

# Response

Para responder a requisição, utilizamos o objeto recebido como parâmetro do tipo “HttpServletResponse” chamando o “getWriter()” seguido do “append(valor)”:

```
response.getWriter().append(json);
```

Ou, se necessário, chamando outra função de requisição que receberá o request e response para efetuar a tratativa:

```
doGet(request, response);
```

# Status Code

Podemos responder também informando erro ou o status code devido pelo: “response.sendError(code)” ou “response.setStatus(code)”:

```
// Send Error
response.sendError(404);

// Send Status Code
response.setStatus(204);
```

# Parâmetros na URL

Para resgatar os parâmetros enviados na URL podemos utilizar o método “getParameter” do request, para resgatar parâmetros enviados via “query param” ou o “getPathInfo” para “path param”:

```
//Query param:
request.getParameter("id");

// Path param:
request.getPathInfo();
```

# Body

Quando a requisição envia um “body”, podemos resgatar utilizando o método “getReader” do request. Porém esse método retorna um “BufferedReader” contendo um array das linhas enviadas. Para o parse correto, recomenda ser utilizado uma biblioteca de conversão JSON como por exemplo o Gson:

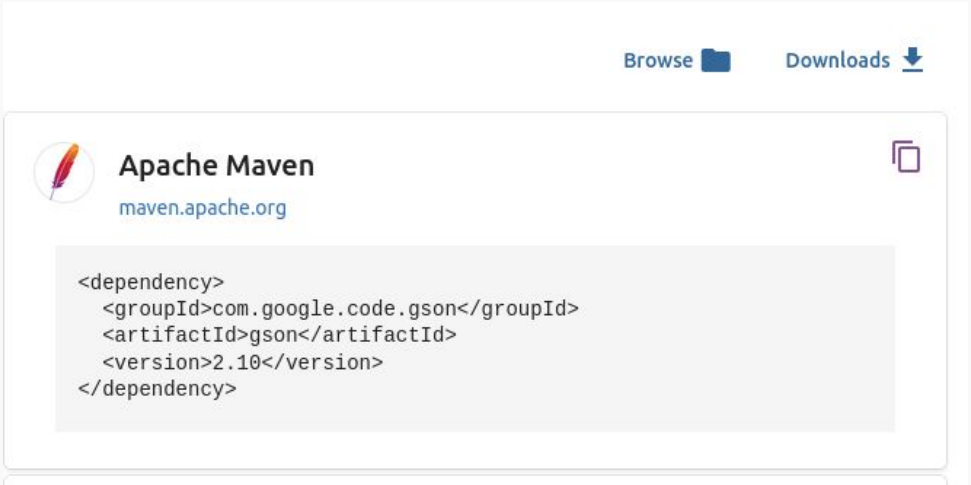
```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    BufferedReader reader = request.getReader();

    Gson gson = new Gson();
    Pessoa pessoa = gson.fromJson(reader, Pessoa.class);
```

# Biblioteca Externa

Como qualquer biblioteca externa, para adicionar você deverá informar no pom.xml (Se for um projeto Maven) ou no “Build Path” do projeto. A dependência ou “jar” dessa biblioteca, pode ser encontrada em:

https://search.maven.org/artifact/com.google.code.gson/gson/2.10/jar



# AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

Clique aqui ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.

# DEVinHouse

Parcerias para desenvolver a sua carreira

## OBRIGADO!

SENAI

<LAB365>