

Roteiro de Atividades – Webservices

Procedimentos preliminares - Instalação das ferramentas

- 1 Instalar o Eclipse para desenvolvimento J2EE (disponível em <https://www.eclipse.org/downloads/packages/release/2019-06/r/eclipse-ide-enterprise-java-developers>).
- 2 Instalar o Apache Tomcat (versão recomendável: 9.0.58)
 - 2.1 Fazer o download do arquivo disponível em <https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.58/bin/apache-tomcat-9.0.58.zip>;
 - 2.2 Descompactar em alguma pasta;
 - 2.3 Em um terminal, acessar a pasta em que o arquivo foi descompactado;
(informações sobre navegação em pastas usando terminal em linux: <http://www.ifsc.usp.br/~lattice/Comandos.pdf>)
 - 2.4 Se estiver usando sistema operacional Linux, executar o seguinte comando no terminal : `chmod +x bin/*.sh`
 - 2.5 Executar o arquivo *startup.sh* (Linux) ou *startup.bat* (Windows);
em sistema operacional Linux, digitar `./bin/startup.sh`
em sistema operacional Windows, digitar `bin/startup`
 - 2.6 Em um navegador, acessar <http://localhost:8080> . Se a instalação foi bem sucedida, aparecerá uma página com informações sobre o Apache Tomcat.

Parte 1 – Obtendo dados do servidor

- 1 No eclipse, criar um novo *Dynamic Web Project* (menu *New > Other > Web > Dynamic Web Project*)
 - 1.1 Na janela *New Dynamic Web Project*, que abrirá após selecionar a opção acima, informar um nome para o projeto no campo *Project Name* e clicar em *Finish*.
- 2 No projeto recém criado, verificar se o arquivo *web.xml* existe dentro da pasta *WebContent>WEB-INF*. Se não existir, clicar com o botão direito sobre o projeto e selecionar a opção *Java EE Tools > Generate Deployment Descriptor Stub*.
- 3 Ajustar o conteúdo do arquivo *web.xml* para que o conteúdo fique conforme o seguinte:
(substituir “nome do projeto” pelo nome do projeto dado no passo 1.1)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
  <display-name>hello_world</display-name>
  <servlet>
    <servlet-name>Jersey REST Service</servlet-name>
    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>jersey.config.server.provider.packages</param-name>
      <param-value> nome do projeto </param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Jersey REST Service</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

- 4 Transformar o projeto em um projeto *Maven*: clicar sobre o projeto com o botão direito do *mouse* e selecionar a opção *Configure > Convert to Maven Project*.

4.1 Na janela *Create new POM*, que abrirá ao selecionar a opção acima, clicar em *Finish*.

- 5 Adicionar as dependências necessárias ao arquivo *pom.xml*, que deve ter o seguinte conteúdo:

```
<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
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>hello_world</groupId>
  <artifactId>hello_world</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>
  <build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
      <plugin>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.2.1</version>
        <configuration>
          <warSourceDirectory>WebContent</warSourceDirectory>
        </configuration>
      </plugin>
    </plugins>
  </build>
  <dependencies>
    <dependency>
      <groupId>org.glassfish</groupId>
      <artifactId>javax.json</artifactId>
      <version>1.0.4</version>
    </dependency>
    <dependency>
      <groupId>org.glassfish.jersey.containers</groupId>
      <artifactId>jersey-container-servlet</artifactId>
      <version>2.29.1</version>
    </dependency>
    <dependency>
      <groupId>org.glassfish.jersey.inject</groupId>
      <artifactId>jersey-hk2</artifactId>
      <version>2.29.1</version>
    </dependency>
    <dependency>
      <groupId>org.glassfish.jersey.media</groupId>
      <artifactId>jersey-media-json-jackson</artifactId>
      <version>2.29.1</version>
    </dependency>
    <dependency>
      <groupId>javax.json</groupId>
      <artifactId>javax.json-api</artifactId>
      <version>1.1.4</version>
    </dependency>
  </dependencies>
</project>
```

6 Atualizar as dependências: clicar com o botão direito do *mouse* sobre o projeto e selecionar a opção *Maven>Update Project*.

7 Criar uma classe Java para implementar o webservice: clicar sobre o projeto com o botão direito do *mouse* e selecionar a opção *New>Class*.

7.1 Na janela *New Java Class*, que abrirá ao selecionar a opção acima, preencher o campo *Name* com um nome para a classe Java e clicar em *Finish*.

8 Implementar a classe conforme a seguir, substituindo <nome da classe> pelo nome dado à classe Java no passo anterior:

```
@Path("/hello")
public class <nome da classe> {

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String sayHello(){
        return "Hello world";
    }

}
```

É possível que, neste passo, o Eclipse sinalize advertências nas linhas que contém as anotações *@Path*, *@GET* e *@Produces*. Para resolver as advertências, deve-se clicar sobre elas e selecionar a opção “Import” referente à advertência sinalizada.

9 Exportar o projeto para um arquivo *.war*: clicar sobre o projeto com o botão direito do *mouse* e selecionar a opção *Export>WAR File* (ou, dependendo da versão do Eclipse, *Export>Web>WAR File*).

10 Copiar o arquivo *.war* produzido para a pasta *TOMCAT_HOME/webapps* (onde *TOMCAT_HOME* é a pasta de instalação do Apache Tomcat).

11 Em um navegador *web*, acessar o endereço *http://localhost:8080/<nome do projeto>/hello* e analisar o resultado.

Parte 2 – Enviando parâmetros ao servidor

1. Adicionar o seguinte método à classe criada anteriormente :

```
@GET
@Path("/{name}")
@Produces(MediaType.TEXT_PLAIN)
public String sayHelloName(@PathParam("name") String msg) {
    return "Hello, " + msg;
}
```

2. Exportar o projeto para um arquivo *.war*: clicar sobre o projeto com o botão direito do *mouse* e selecionar a opção *Export>WAR File* (ou, dependendo da versão do Eclipse, *Export>Web>WAR File*).

3. Acessar o endereço *http://localhost:8080/<nome do projeto>/hello/nome* e analisar o resultado.

Parte 3 – Obtendo dados do servidor em formato JSON

1. Adicionar o seguinte método à classe criada anteriormente:

```
@GET
@Path("/hellojson")
@Produces(MediaType.APPLICATION_JSON)
public String sayHelloJson() {
    return "{\"english\": \"hello world\", \"+
        \"portugues\": \"ola, mundo\", \"+
        \"francais\": \"bonjour, tout le monde\" \" +
        \"}\";
}
```

2. Exportar o projeto para um arquivo *.war*: clicar sobre o projeto com o botão direito do *mouse* e selecionar a opção *Export>WAR File* (ou, dependendo da versão do Eclipse, *Export>Web>WAR File*).
3. Acessar o endereço *http://localhost:8080/<nome do projeto>/hellojson*