# 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/2022-06/r/eclipse-ide-enterprise-java-and-web-developers).

2 Instalar o Apache Tomcat (versão recomendável: 10.1.2)

2.1 Fazer o download do arquivo disponível em https://dlcdn.apache.org/tomcat/tomcat-10/v10.1.2/bin/apache-tomcat-10.1.2.zip;

2.2 Descompactar em alguma pasta;

2.3 Em um terminal, acessar a pasta em que o arquivo foi descompactado;

2.4 (informações sobre navegação em pastas usando terminal em linux: http://www.ifsc.usp.br/~lattice/Comandos.pdf)

2.5 Se estiver usando sistema operacional Linux, executar o seguinte comando no terminal : `chmod +x bin/*.sh`

2.6 Executar o arquivo *startup.sh* (Linux) ou *startup.bat* (Windows);

2.6.1 em sistema operacional Linux, digitar `./bin/startup.sh`

2.6.2 em sistema operacional Windows, digitar `bin/startup`

2.7 Em um navegador, acessar http://localhost:8080 . Se a instalação foi bem sucedida, aparecerá uma página com informações sobre o Apache Tomcat.

## Atividades

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 *src>main>webapp>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="http://xmlns.jcp.org/xml/ns/javaee"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
            http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
            id="WebApp_ID" version="3.1">
  <display-name>alunos</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> nome do projeto  </groupId>
        <artifactId> nome do projeto  </artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>war</packaging>
        <build>
                <plugins>
                        <plugin>
                                <artifactId>maven-compiler-plugin</artifactId>
                                <version>3.8.1</version>
                                <configuration>
                                        <release>17</release>
                                </configuration>
                        </plugin>
                        <plugin>
                                <artifactId>maven-war-plugin</artifactId>
                                <version>3.2.3</version>
                        </plugin>
                </plugins>
        </build>
        <dependencies>
                <dependency>
                        <groupId>org.glassfish</groupId>
                        <artifactId>javax.json</artifactId>
                        <version>1.1.4</version>
                </dependency>
                <dependency>
                        <groupId>org.glassfish.jersey.containers</groupId>
                <artifactId>jersey-container-servlet</artifactId>
                        <version> 3.1.0-M3 </version>
                </dependency>
                <dependency>
                        <groupId>org.glassfish.jersey.inject</groupId>
                        <artifactId>jersey-hk2</artifactId>
                        <version> 3.1.0-M3 </version>
                </dependency>
                <dependency>
                        <groupId>org.glassfish.jersey.media</groupId>
                        <artifactId>jersey-media-json-jackson</artifactId>
                        <version> 3.1.0-M3 </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("/alunos")
public class <nome da classe> {

	private static  JsonArray alunos = Json.createArrayBuilder()
			.add(Json.createObjectBuilder()
					.add("id", 1)
					.add("nome", "alice")
					.add("curso","Engenharia de Controle e Automação")
					.add("idade",20))
			.add(Json.createObjectBuilder()
					.add("id", 2)
					.add("nome", "bob")
					.add("curso","Engenharia de Materiais")
					.add("idade",18))
			.build();

	@GET
	@Path("/getalunos")
	@Produces(MediaType.APPLICATION_JSON)
	public String sayHello(){
		return this.alunos.toString();
	}

	@GET
	@Path("/getaluno/{id}")
	@Produces(MediaType.APPLICATION_JSON)
	public String getById(@PathParam("id") int pId) {
		for(int i=0;i<alunos.size();i++) {
			if(alunos.getJsonObject(i).getInt("id")==pId)
				return alunos.getJsonObject(i).toString();
		}

		return null;
	}

	@POST
	@Path("/addalunojson")
	@Consumes(MediaType.APPLICATION_JSON)
	public String addAlunoJson(String aluno) {
		System.out.println("recevido " + aluno);
		JsonReader jsonReader = Json.createReader(new StringReader(aluno));
		JsonObject object = jsonReader.readObject();
		jsonReader.close();
		alunos = addToJsonArray(alunos, object);
		return alunos.toString();
	}

	private JsonArray addToJsonArray(JsonArray array, JsonObject object) {
		JsonArrayBuilder builder = Json.createArrayBuilder();
		for(int i=0;i<array.size();i++)
			builder.add(array.get(i));
		builder.add(object);
		return builder.build();
	}

}
```

Observação: É possível que, neste passo, o Eclipse sinalize advertências nas linhas que contém as anotações @Path, @GET, @Produces. Etc. 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, acessar as URLs a seguir, observando os resultados:
- 11.1 http://localhost:8080/**nome do projeto**/alunos/getalunos
- 11.2 http://localhost:8080/**nome do projeto**/alunos/getaluno/1
- 11.3 http://localhost:8080/**nome do projeto**/alunos/getaluno/2

12 Em um terminal, digitar o seguinte comando (sem quebras de linha)

```
curl --header "Content-Type: application/json" --request POST --data
'{"id":3,"nome":"ana","curso":"textil","idade":25}' http://localhost:8080/nome do
projeto/alunos/addalunojson
```

13 Em um navegador, acessar http://localhost:8080/**nome do projeto**/alunos/getalunos e analisar o resultado.

**Parte 4 – Consumo do webservice**

1. Implementar um programa em que o usuário informe o código (id) de um aluno e que, consultando o webservice criado, imprima o nome, a idade e o curso do aluno correspondente.