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: 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.

2. Atividades – Parte 1

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>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:
(substituir “nome do projeto” pelo nome do projeto dado no passo 1.1)

```
<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>
                <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.jersey.containers</groupId>
                <artifactId>jersey-container-servlet</artifactId>
                        <version>2.31</version>
                </dependency>
                <dependency>
                        <groupId>org.glassfish.jersey.media</groupId>
                        <artifactId>jersey-media-json-jackson</artifactId>
                        <version>2.31</version>
                </dependency>
                <dependency>
                        <groupId>org.glassfish.jersey.inject</groupId>
                        <artifactId>jersey-hk2</artifactId>
                        <version>2.31</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.*

8 Criar a classe *Aluno*. Para isso, clicar sobre o projeto com o botão direito do *mouse* e selecionar a opção *New* > *Class*. Na janela *New Java Class*, que abrirá ao selecionar esta opção, preencher o campo *Name* com “Aluno” (sem as aspas) e clicar em *Finish*.

9 Implementar a classe *Aluno* conforme a seguir:

```
public class Aluno {

	private int id;
	private String nome;
	private String curso;
	private int idade;


	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public String getCurso() {
		return curso;
	}

	public void setCurso(String curso) {
		this.curso = curso;
	}

	public int getIdade() {
		return idade;
	}

	public void setIdade(int idade) {
		this.idade = idade;
	}


}
```

10 Criar a classe *AlunoDAO*, seguindo o mesmo processo utilizado no passo 8 e implementá-la conforme abaixo:

```
public class AlunoDAO {

	private static final AlunoDAO instance = new AlunoDAO();
	private static List<Aluno> alunos = new ArrayList<Aluno>();


	private AlunoDAO() {

	}

	public static AlunoDAO getInstance() {
		return instance;
	}

	public void add(Aluno aluno) {
		alunos.add(aluno);
	}

	public List<Aluno> getAlunos() {
		return alunos;
	}

	public Aluno first() {
		return alunos.get(0);
	}


	public static Aluno getById(int id) {
		for(Aluno a:alunos) {
			if(a.getId()==id) {
				return a;
			}
		}
		return null;
	}

}
```

11 Criar a classe *AlunoWS*, seguindo o mesmo processo utilizado no passo 8 e implementá-la conforme abaixo:

```
@Path("/alunos")
public class AlunoWS {

	@POST
	@Path("/addaluno")
	@Consumes("text/plain")
	public void add(String nome) {
		Aluno aluno = new Aluno();
		aluno.setId(1);
		aluno.setNome(nome);
		aluno.setCurso("eca");
		aluno.setIdade(99);
		AlunoDAO.getInstance().add(aluno);
	}

	@GET
	@Path("/getfirstjson")
	@Produces(MediaType.APPLICATION_JSON)
	public Aluno primeiroJson() {
		AlunoDAO alunos = AlunoDAO.getInstance();
		return alunos.first();
	}

	@GET
	@Path("/getalljson")
	@Produces(MediaType.APPLICATION_JSON)
	public List<Aluno> getAllJson() {
		AlunoDAO alunos = AlunoDAO.getInstance();
		return alunos.getAlunos();
	}

	@POST
	@Path("/addalunojson")
	@Consumes(MediaType.APPLICATION_JSON)
	public void addJson(Aluno aluno) {
		AlunoDAO.getInstance().add(aluno);
	}

}
```

12 Fazer o *delpoy* do projeto.

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

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

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

```
curl --header "Content-Type: application/json" --request POST --data '{"id":1,
"nome":"bob","curso":"Engenharia de Controle e Automação","idade":18}'
http://localhost:8080/nome do projeto/alunos/addalunojson
```

14 Em um navegador, acessar a URL

`http://localhost:8080/nome do projeto/alunos/getalljson` e analisar o resultado.

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

```
curl --header "Content-Type: application/json" --request POST --data '{"id":2,
"nome":"alice","curso":"Engenharia de Materiais","idade":21}'
http://localhost:8080/nome do projeto/alunos/addalunojson
```

16 Em um navegador, acessar a URL
`http://localhost:8080/`**`nome do projeto`**`/alunos/getalljson` e analisar o resultado.

17 Em um navegador, acessar a URL
`http://localhost:8080/`**`nome do projeto`**`/alunos/getfirstjson` e analisar o resultado.