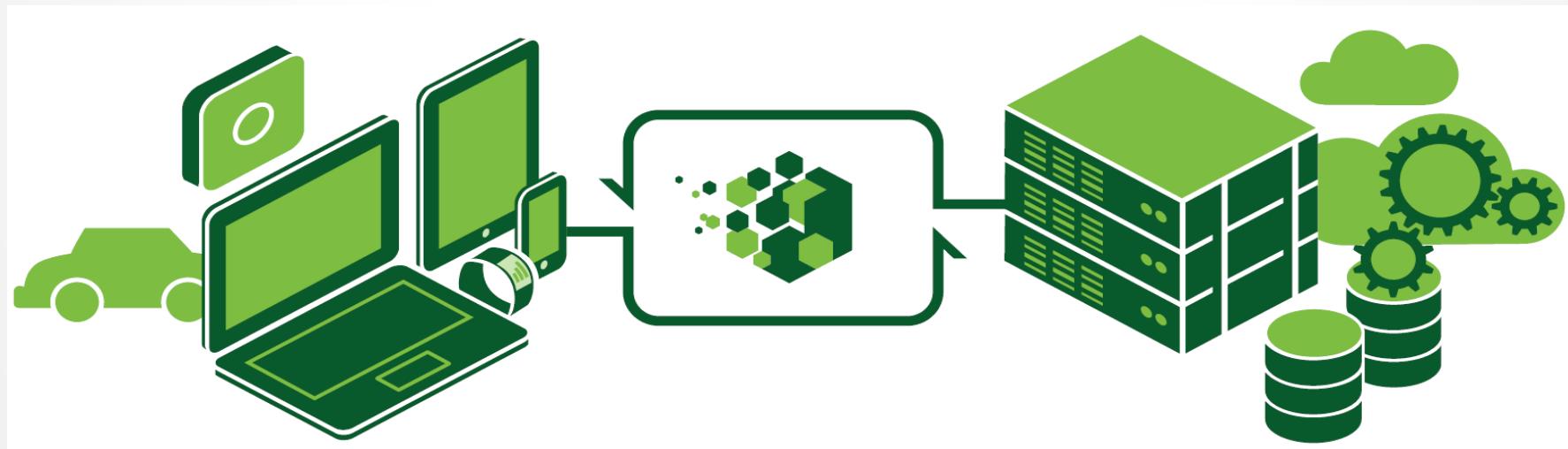


# Tutorial: Criação de um Web Service usando Java 8 + Jersey + hibernate 3 e JPA





Nome: Bruno Ábia  
e-mail: bruno.abia@gmail.com

# Objetivo

- O objetivo desse material é apenas ensinar como fazer um “Hello World” com um WS em java, mas nada impede que o desenvolvedor aprimore o código mostrado ao longo desse curso.

# Ementa

- Ambiente de Desenvolvimento
- Apresentação das APIs
- Criação do projeto
- Desenvolvimento do caso de teste
- Aperfeiçoar o projeto

# Ambiente de Desenvolvimento

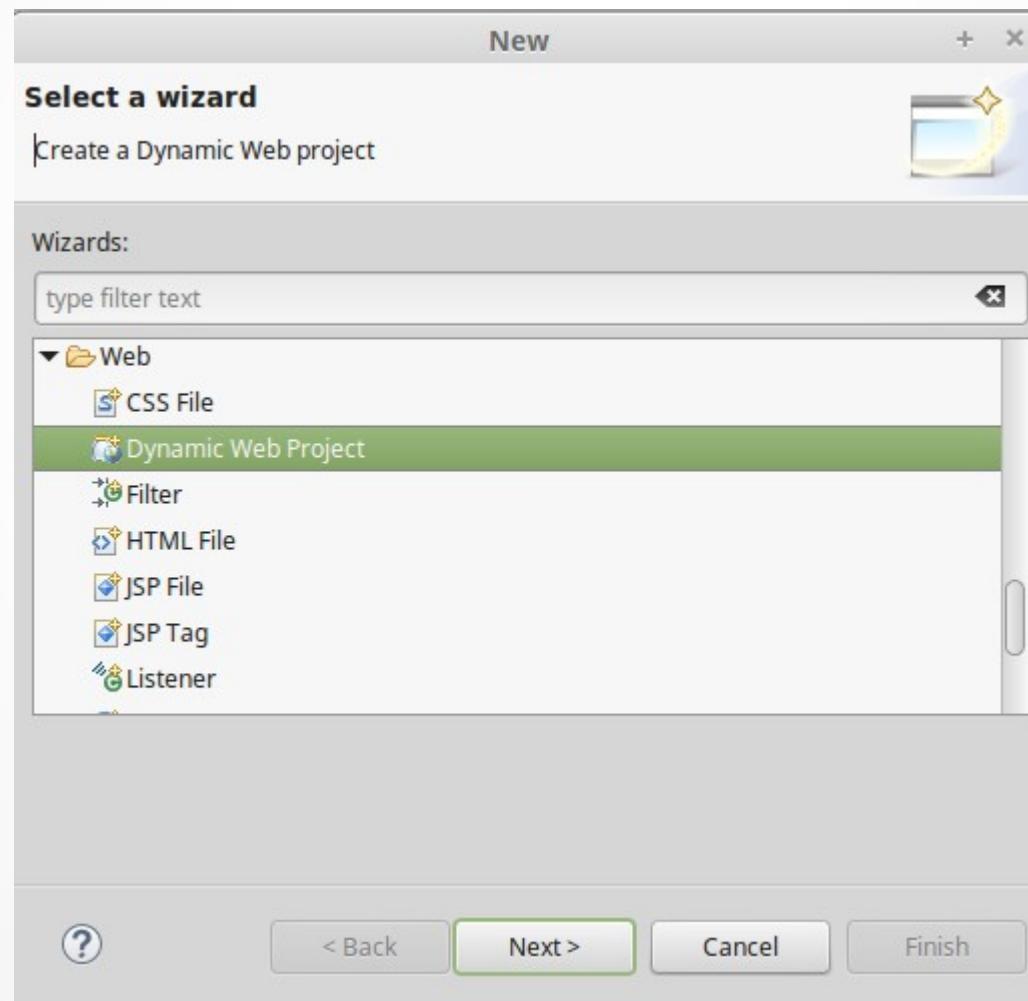
- <https://eclipse.org/downloads/> - Eclipse
- <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html?ssSourceSiteId=otnpt> - JDK 8
- <https://tomcat.apache.org/download-80.cgi> - tomcat 8
- [https://www.apachefriends.org/pt\\_br/index.html](https://www.apachefriends.org/pt_br/index.html) – xampp

# Apresentação das APIs

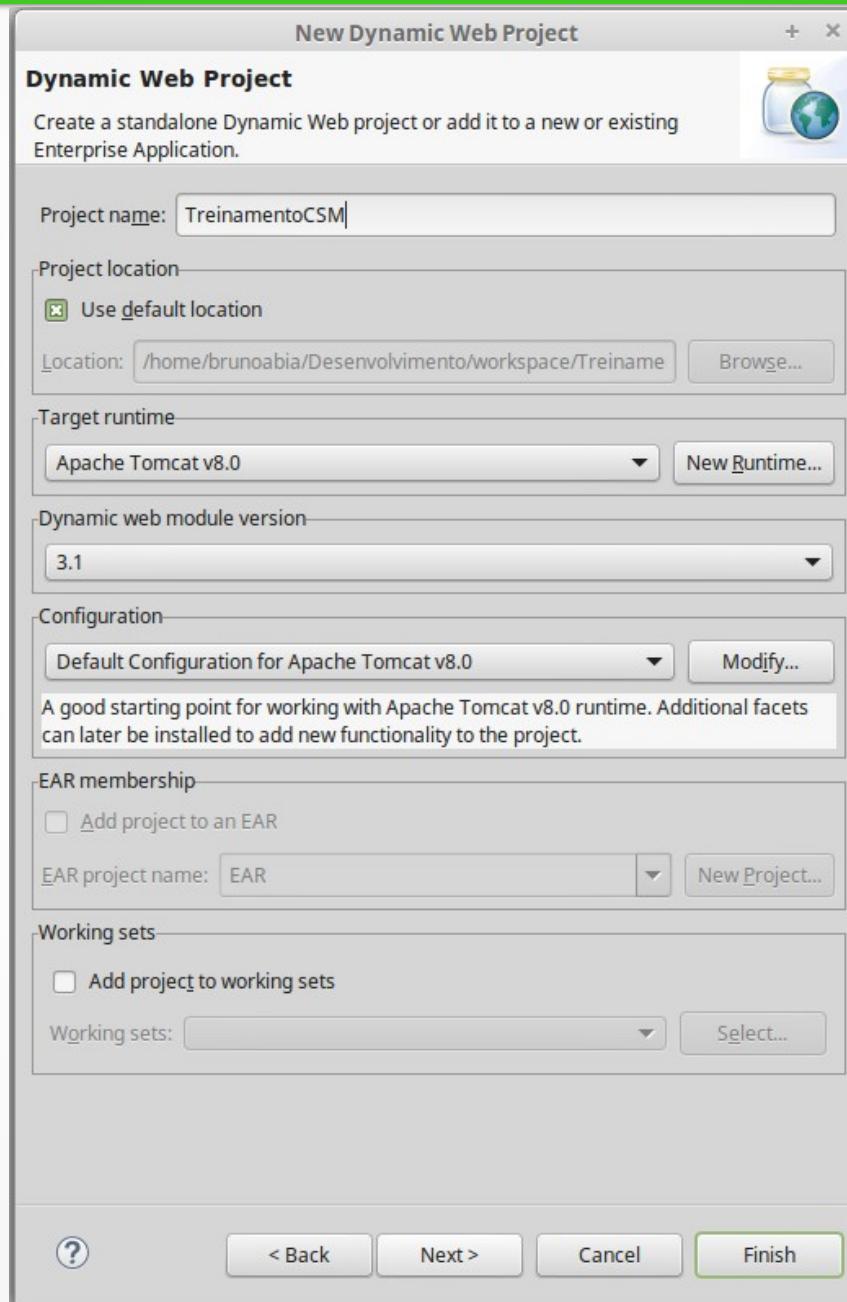


# Criação do projeto

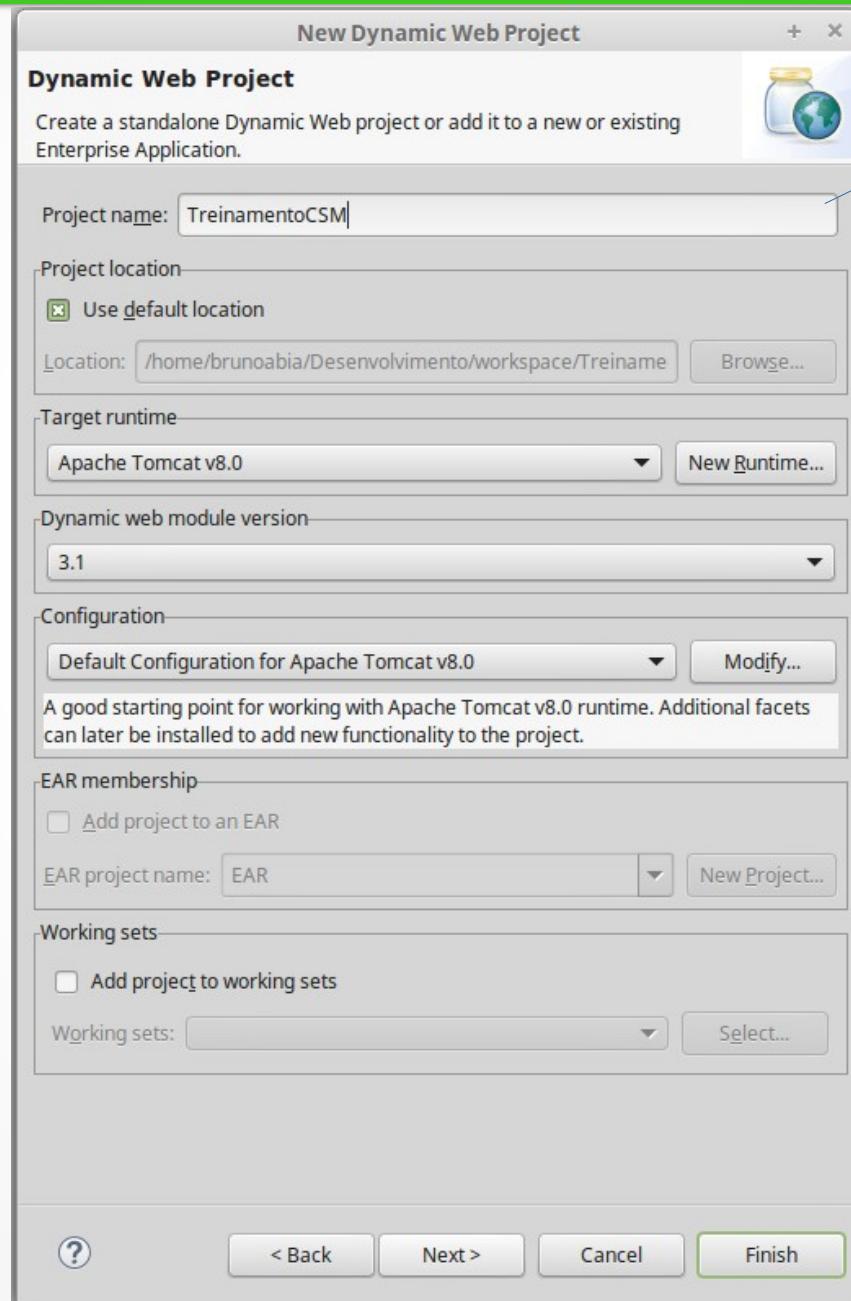
- Create a Dynamic Web project



# Criação do Projeto



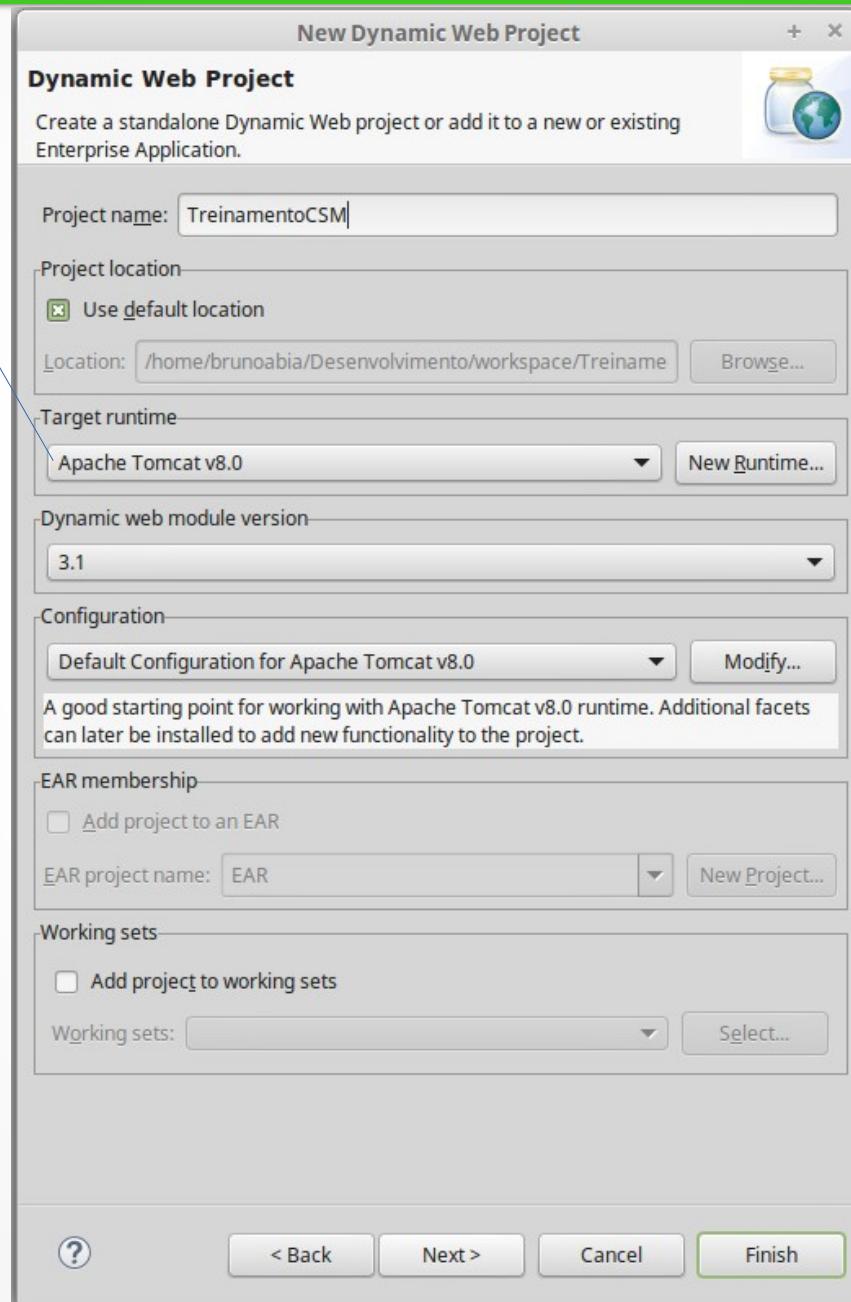
# Criação do Projeto



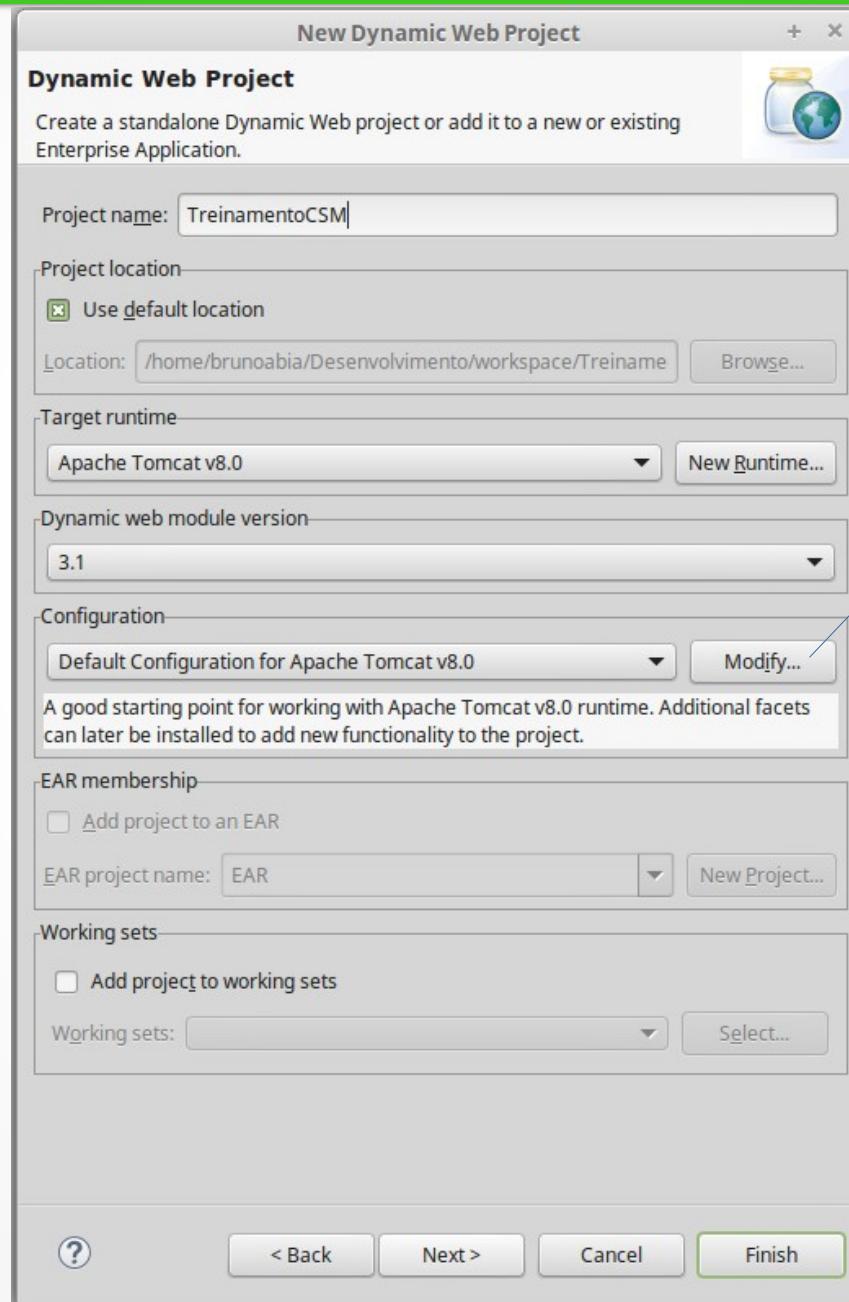
Informe o nome  
do projeto

# Criação do Projeto

O target marca  
o tomcat

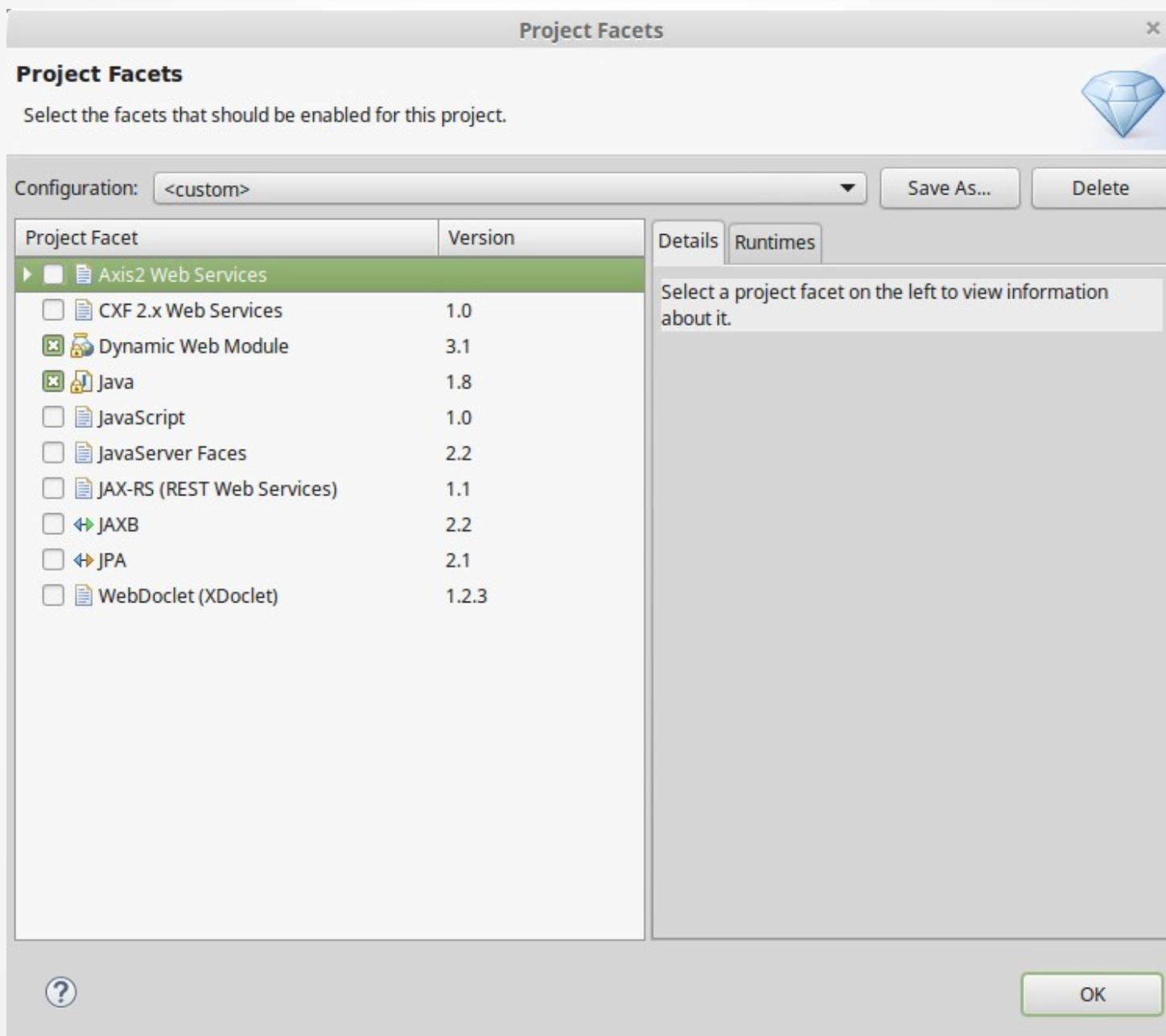


# Criação do Projeto

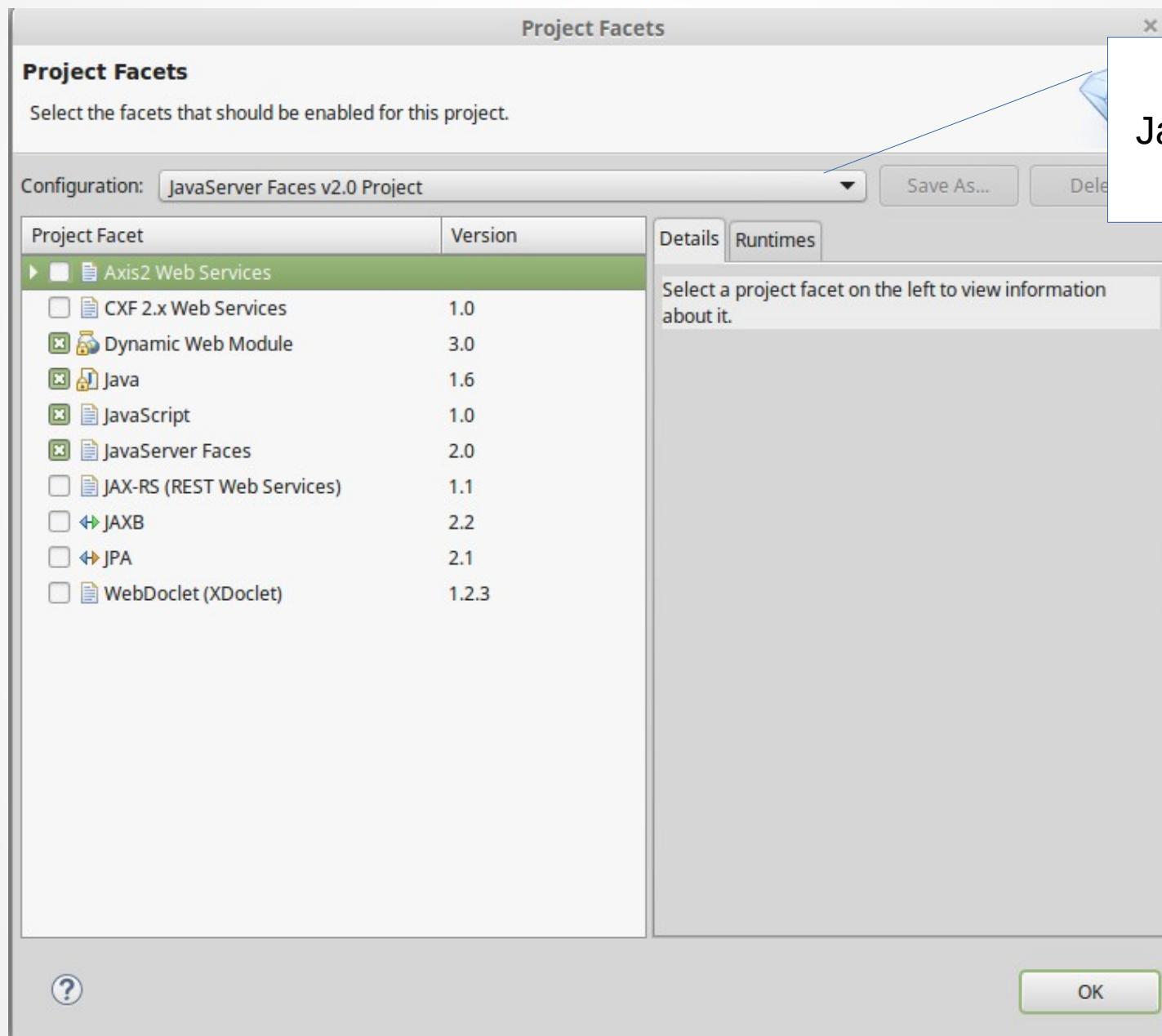


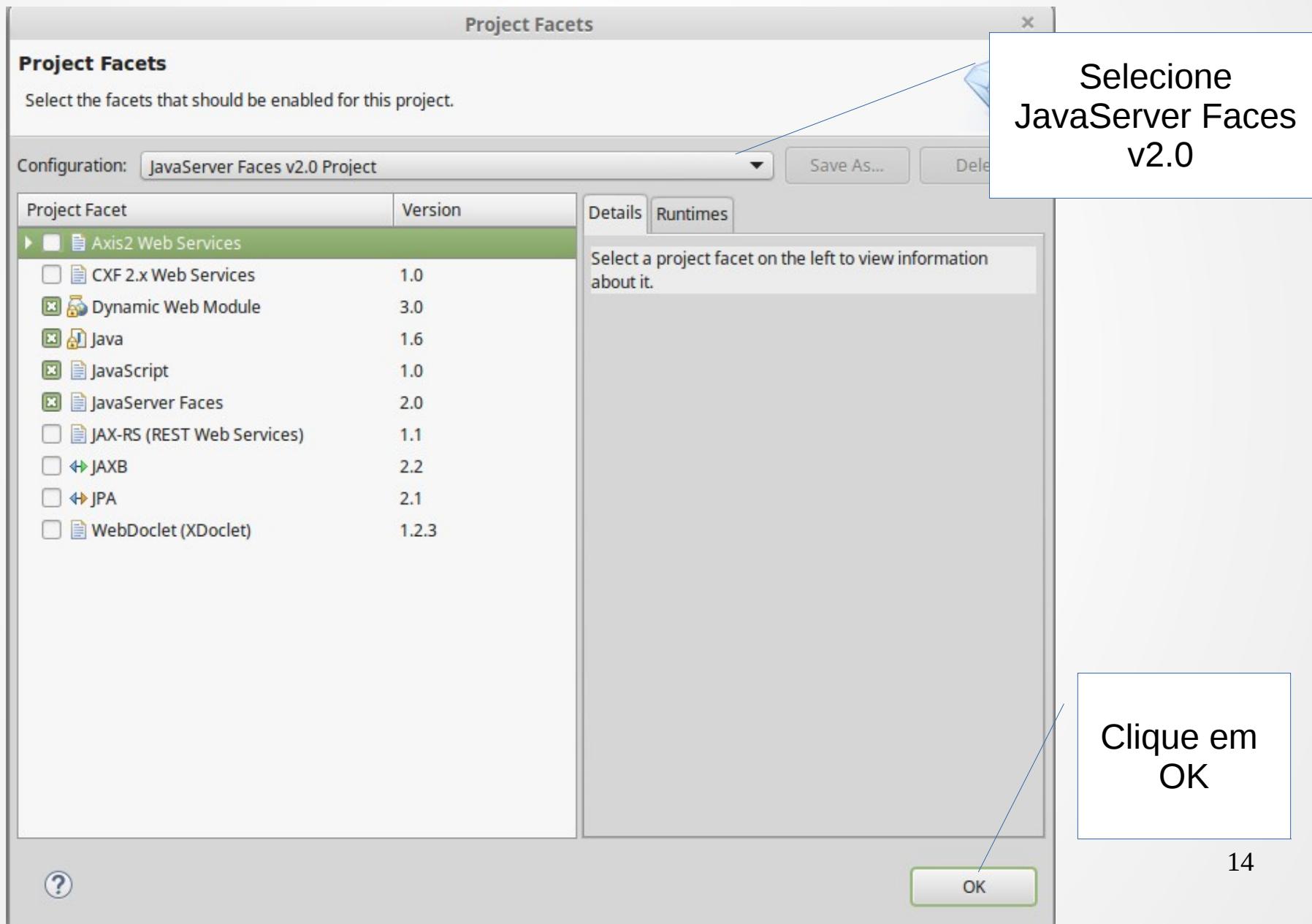
Clique em modify

# Criação do Projeto

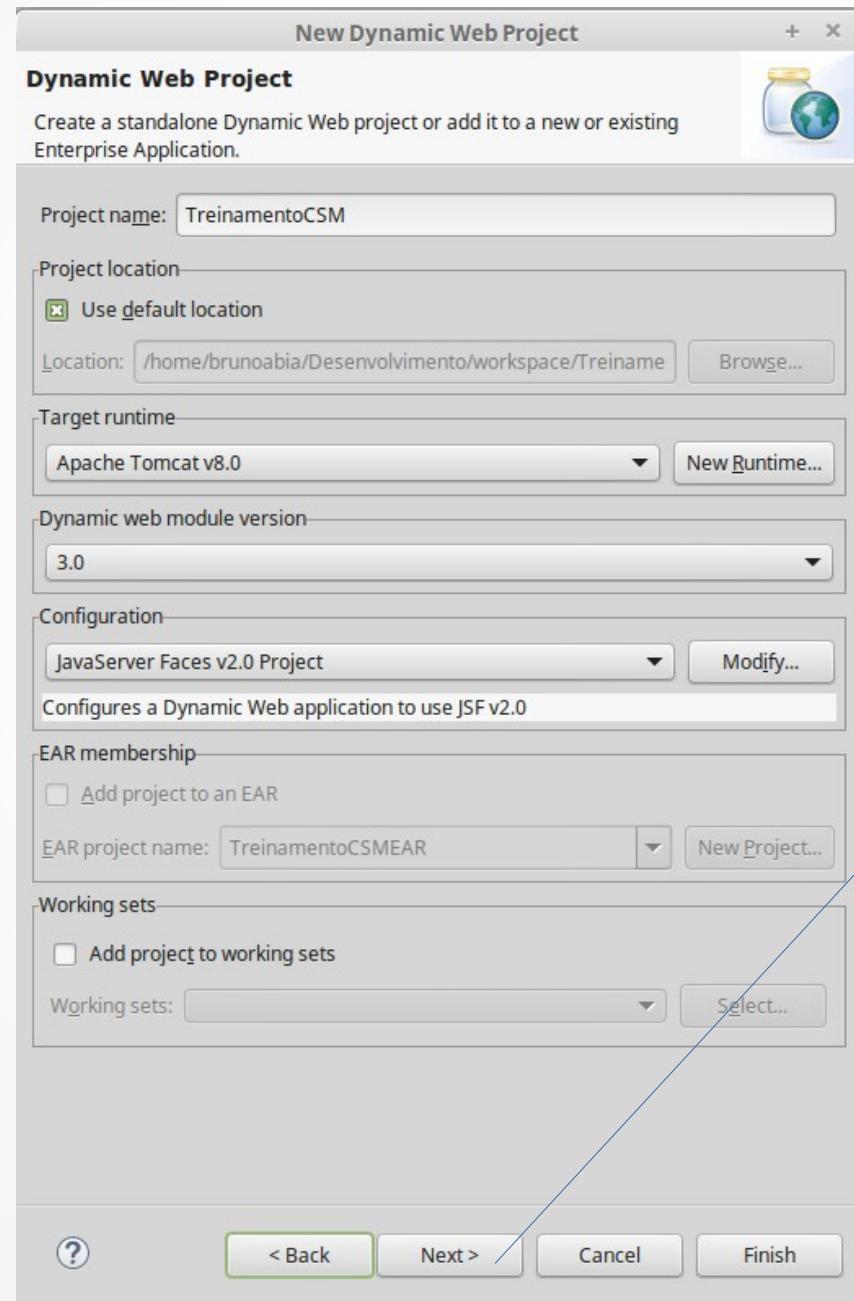


Seleccione  
JavaServer Faces  
v2.0



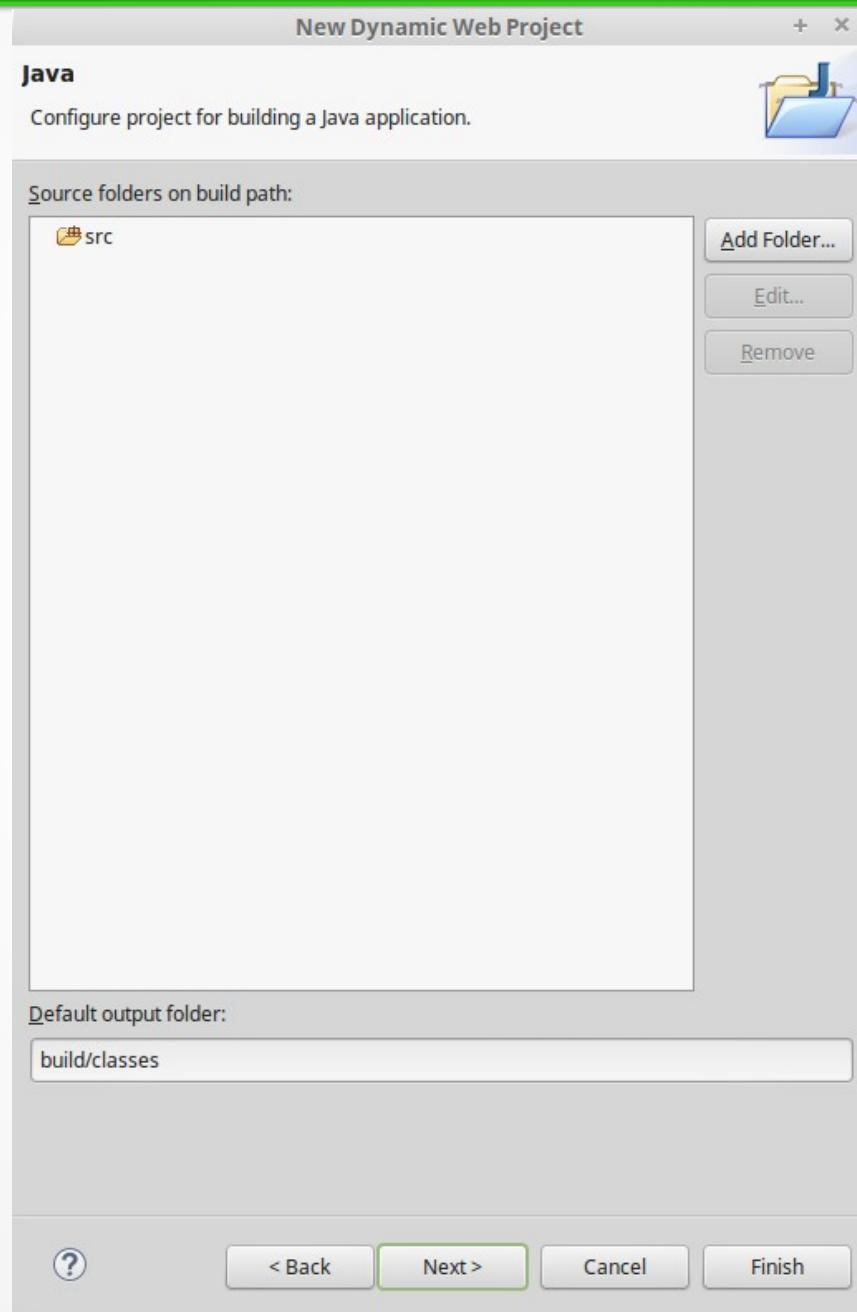


# Criação do Projeto

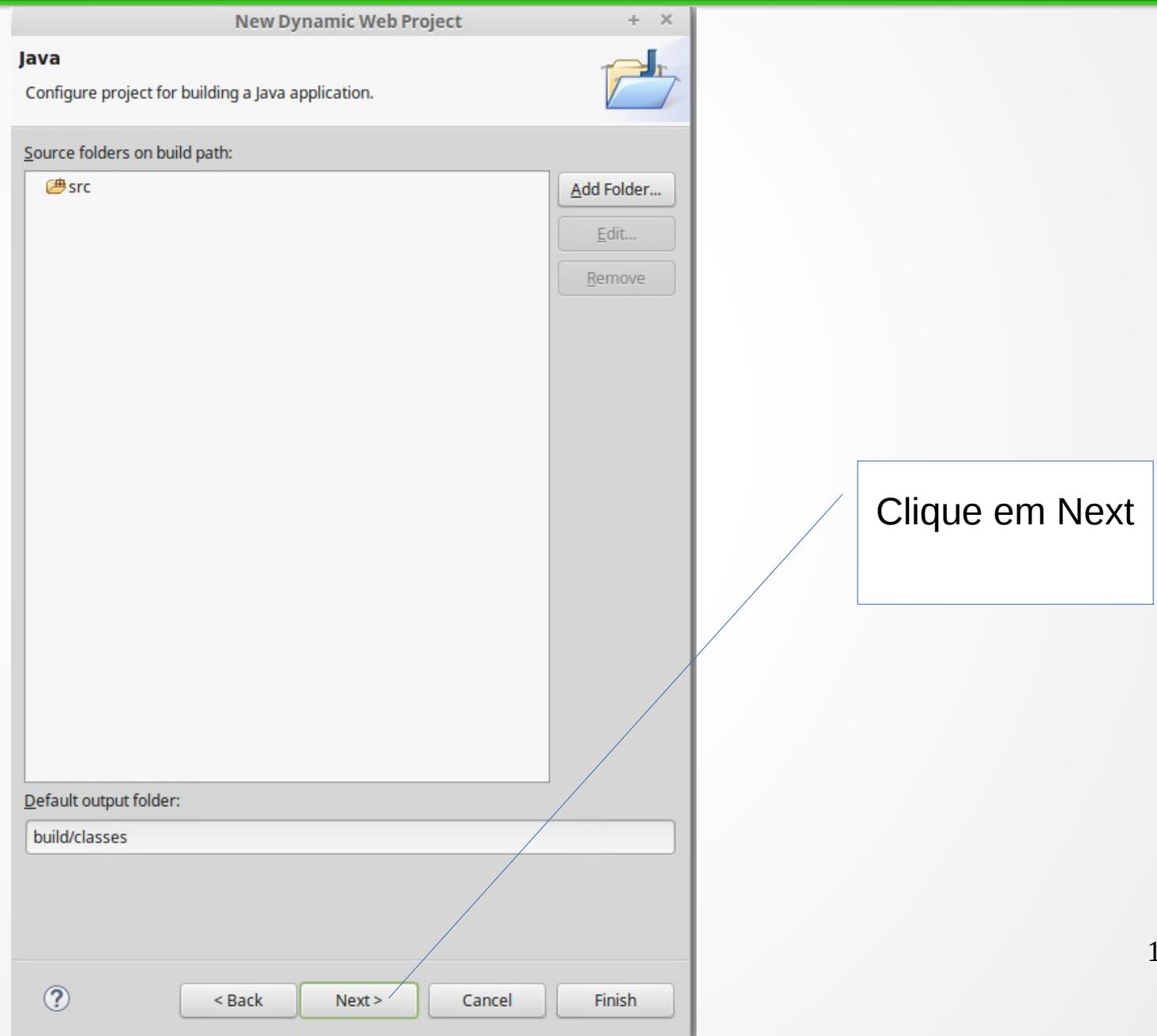


Clique em NEXT

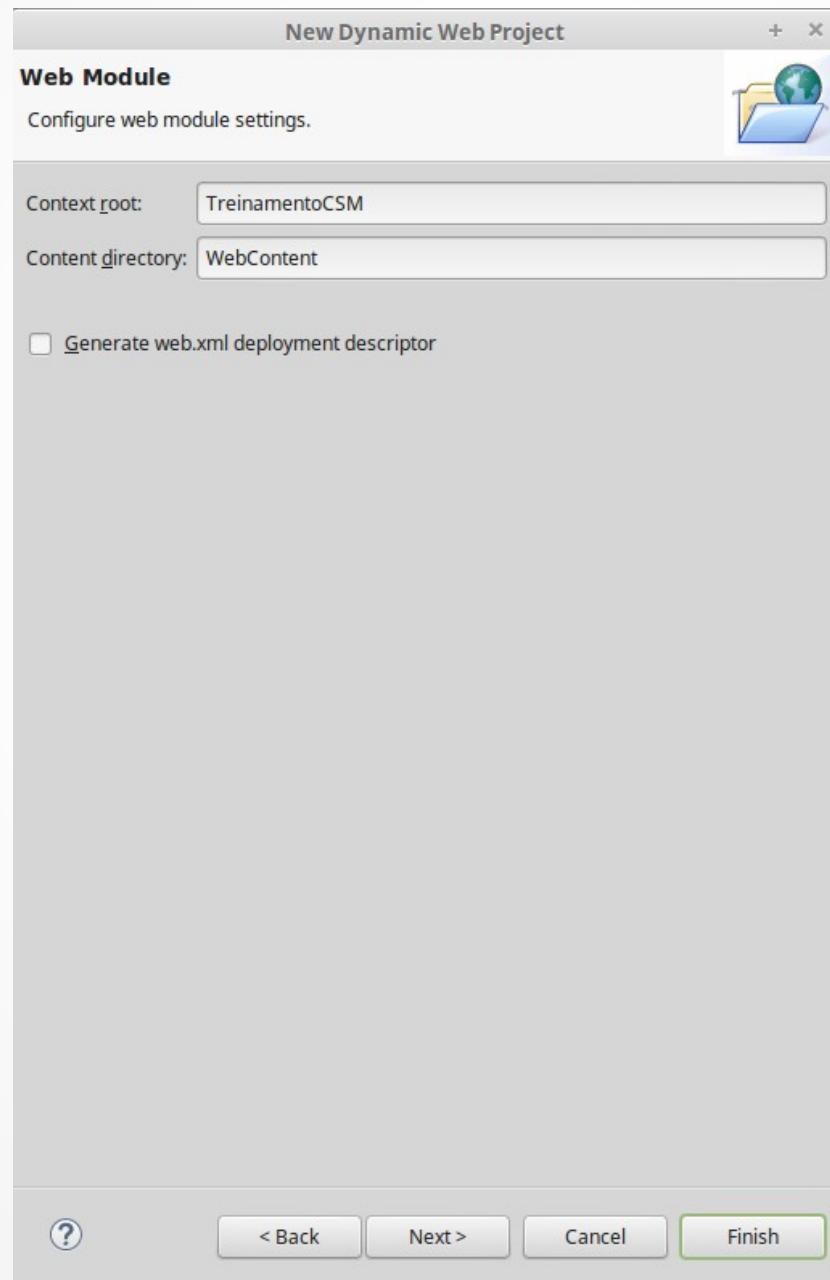
# Criação do Projeto



# Criação do Projeto

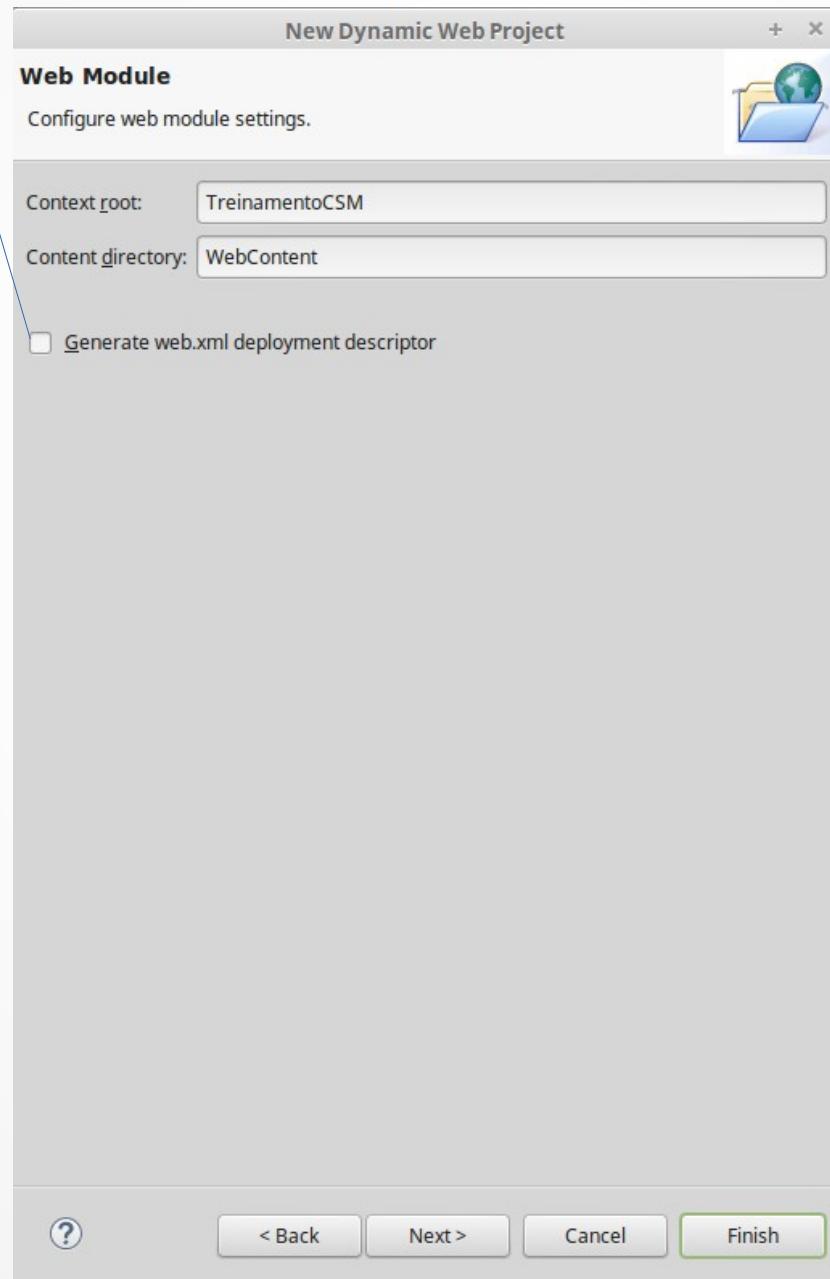


# Criação do Projeto



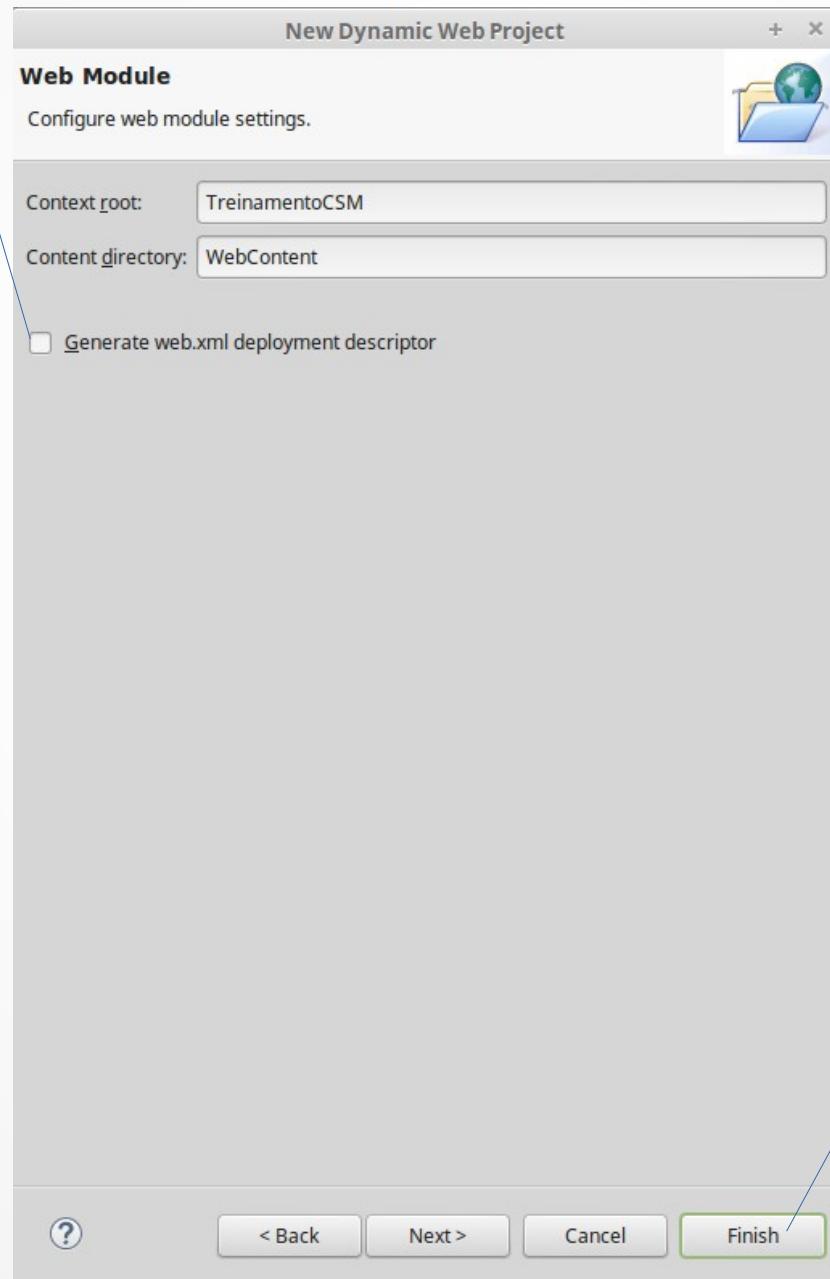
# Criação do Projeto

Marque a opção para gerar web.xml



# Criação do Projeto

Marque a opção para gerar web.xml



Clique em finish

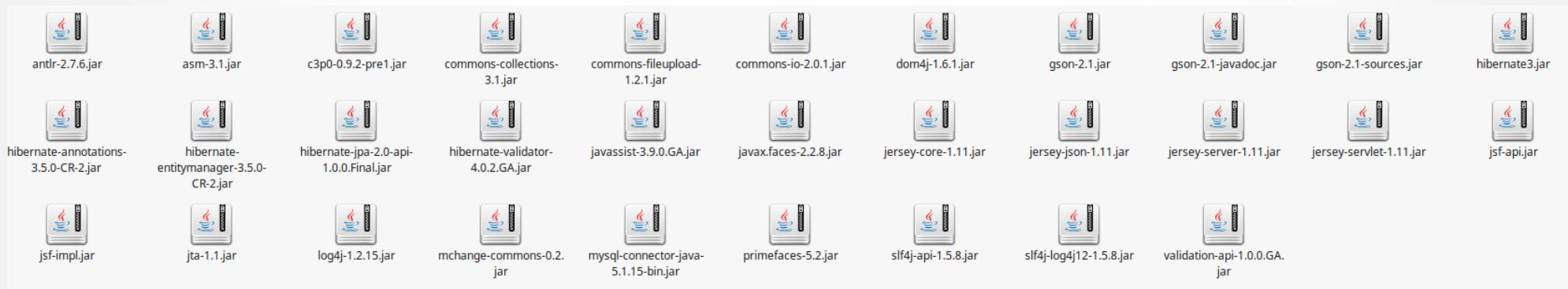
# Projeto Criado !!!!!



# Agora vamos começar de verdade...



# 1º vamos copiar os .jars para seu projeto...





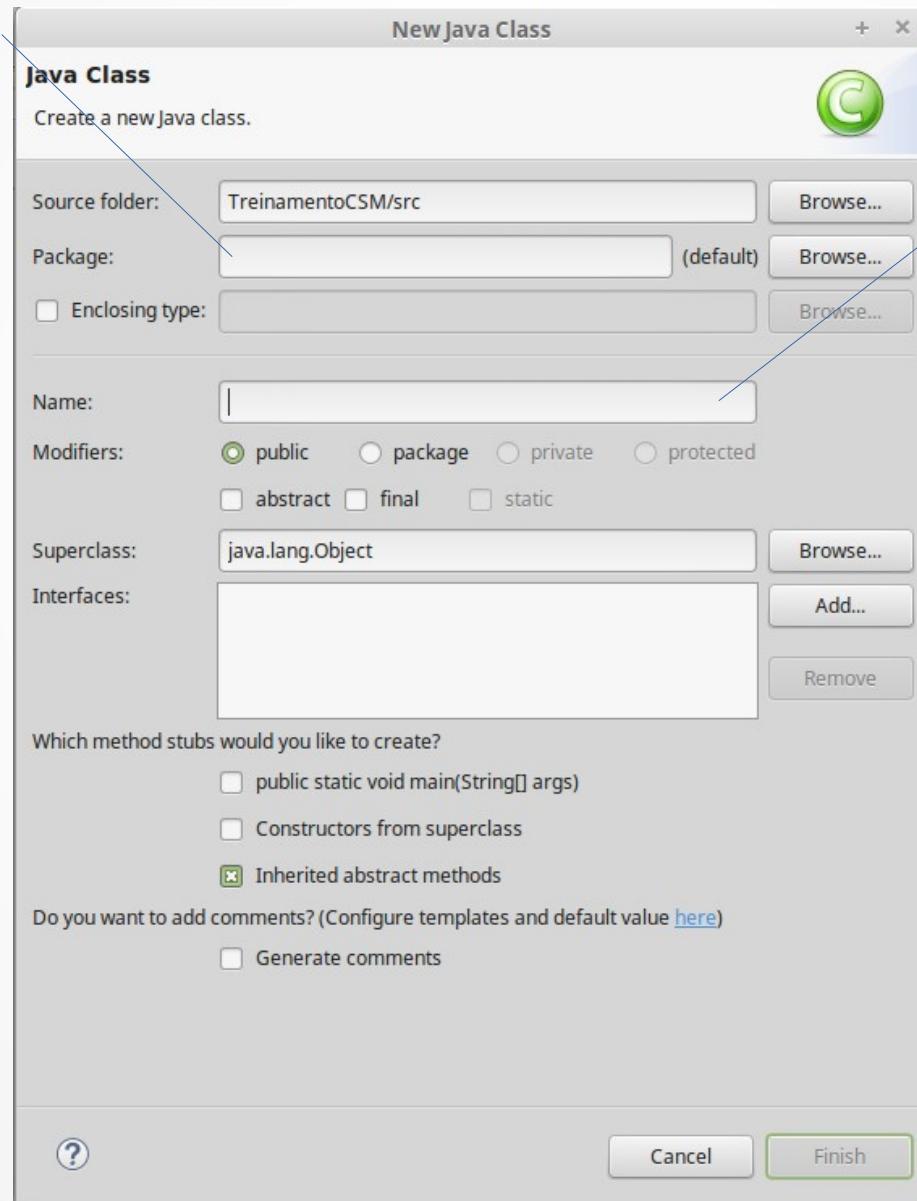
Selecione todos os jars e adicione .build path.

Clicando com o botão Direito e selecionando a opção build path

# Vamos para os códigos

Vamos criar  
um package para  
organizar  
nossa código:

br.com.csm.bean

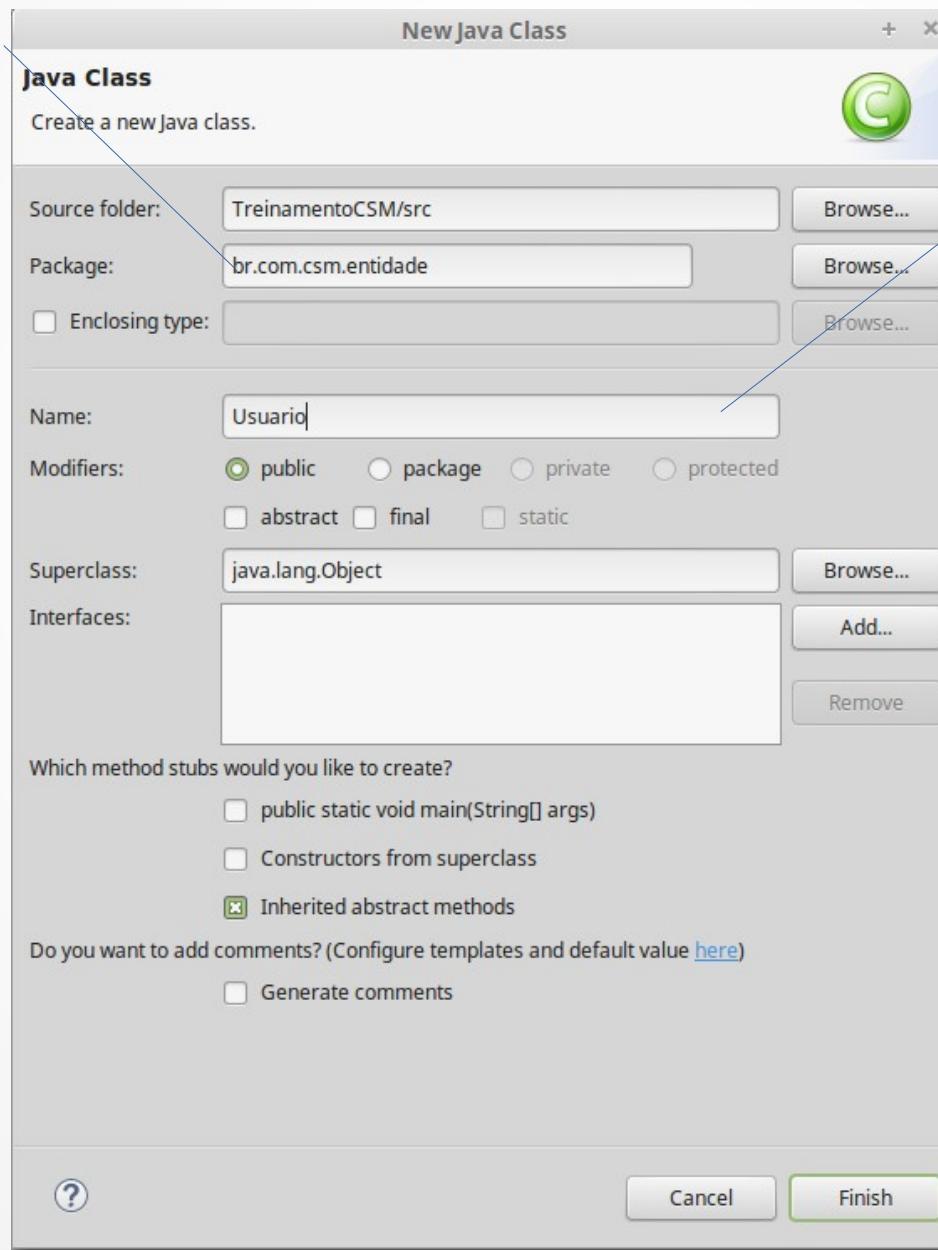


Vamos criar a  
classe Usuario

# Vamos para os códigos

Vamos criar  
um package para  
organizar  
nossa código:

br.com.csm.bean

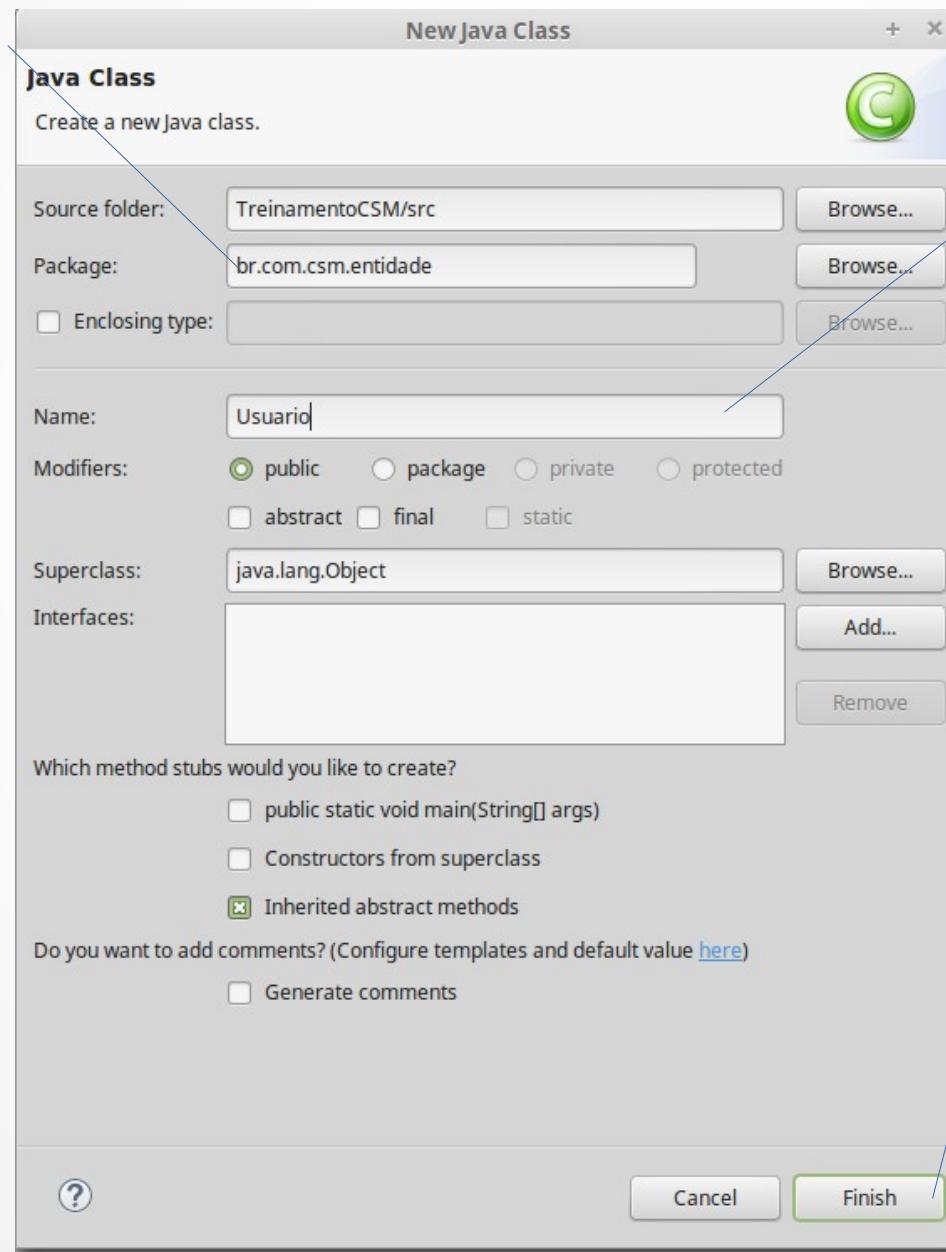


Vamos criar a  
classe Usuario

# Vamos para os códigos

Vamos criar  
um package para  
organizar  
nossa código:

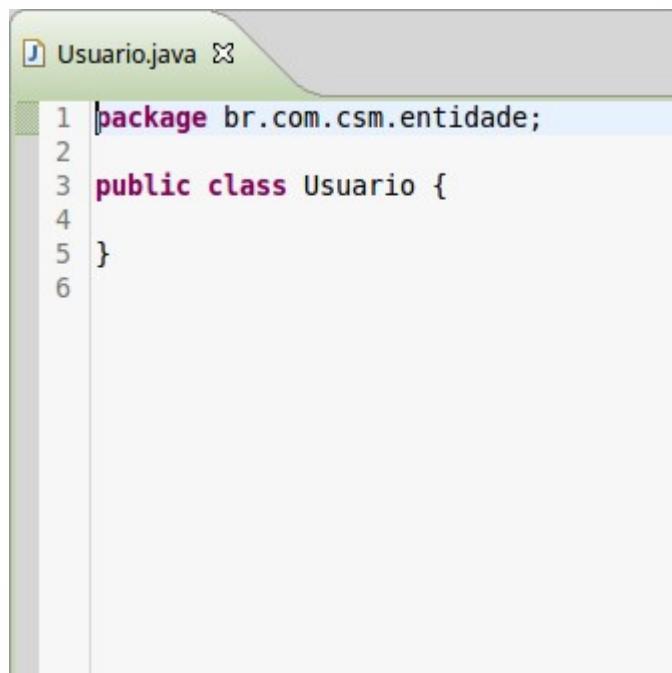
br.com.csm.bean



Vamos criar a  
classe Usuario

Clique  
em Finish

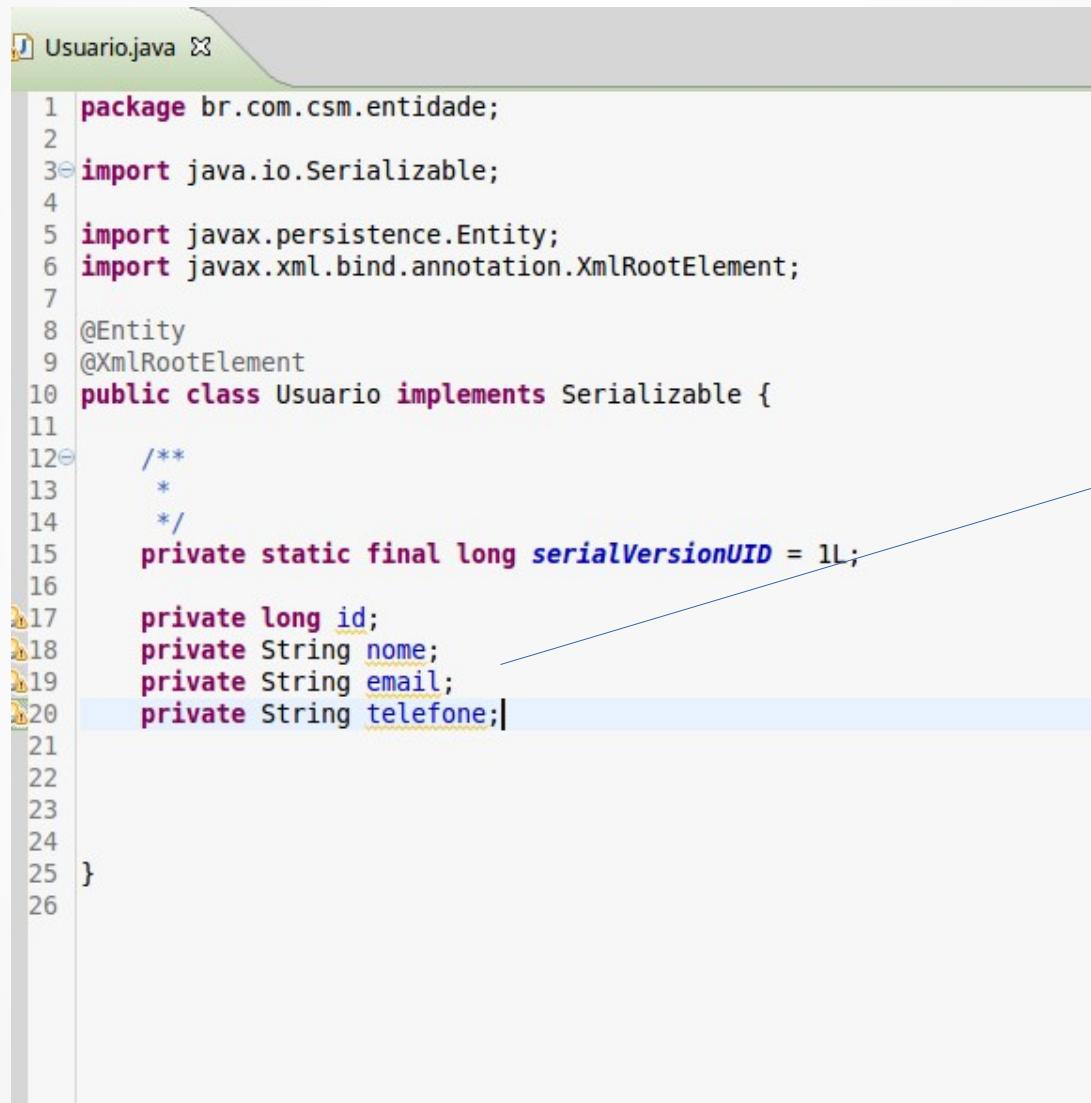
# Vamos começar a definir os atributos



A screenshot of a Java code editor window titled "Usuario.java". The code shown is:

```
1 package br.com.csm.entidade;
2
3 public class Usuario {
4
5 }
6
```

# Criando atributos

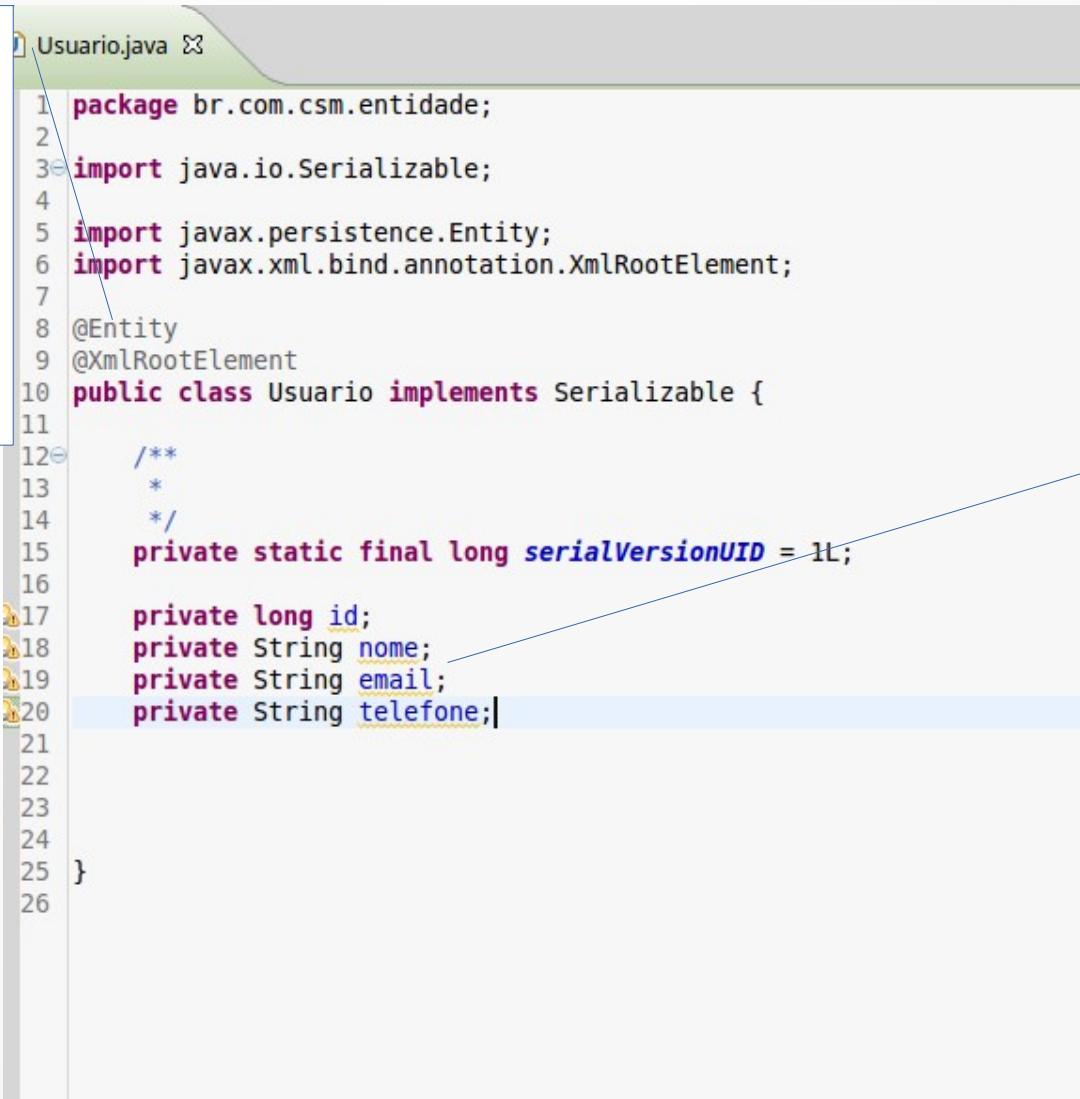


```
1 package br.com.csm.entidade;
2
3 import java.io.Serializable;
4
5 import javax.persistence.Entity;
6 import javax.xml.bind.annotation.XmlRootElement;
7
8 @Entity
9 @XmlRootElement
10 public class Usuario implements Serializable {
11
12     /**
13      *
14      */
15     private static final long serialVersionUID = 1L;
16
17     private long id;
18     private String nome;
19     private String email;
20     private String telefone;
21
22
23
24
25 }
26
```

Para nosso exemplo usaremos esses atributos

# Criando atributos

O Entity é um annotation do Hibernate que simboliza que o mesmo nome da classe sera o da tabela no banco



```
1 package br.com.csm.entidade;
2
3 import java.io.Serializable;
4
5 import javax.persistence.Entity;
6 import javax.xml.bind.annotation.XmlRootElement;
7
8 @Entity
9 @XmlRootElement
10 public class Usuario implements Serializable {
11
12     /**
13      *
14      */
15     private static final long serialVersionUID = 1L;
16
17     private long id;
18     private String nome;
19     private String email;
20     private String telefone;
21
22
23
24
25
26 }
```

Para nosso exemplo usaremos esses atributos

# Criando atributos

O Entity é um annotation do Hibernate que simboliza que o mesmo nome da classe sera o da tabela no banco

O XMLRootElement é um annotation que usaremos no Jersey

```
1 package br.com.csm.entidade;
2
3 import java.io.Serializable;
4
5 import javax.persistence.Entity;
6 import javax.xml.bind.annotation.XmlRootElement;
7
8 @Entity
9 @XmlRootElement
10 public class Usuario implements Serializable {
11
12     /**
13      *
14      */
15     private static final long serialVersionUID = 1L;
16
17     private long id;
18     private String nome;
19     private String email;
20     private String telefone;
21
22
23
24
25
26 }
```

Para nosso exemplo usaremos esses atributos

# Criando atributos

O Entity é um annotation

do Hibernate que simboliza que o mesmo nome da classe sera o da tabela no banco

O XMLRootElement é um annotation que usaremos no Jersey

```
1 package br.com.csm.entidade;
2
3 import java.io.Serializable;
4
5 import javax.persistence.Entity;
6 import javax.xml.bind.annotation.XmlRootElement;
7
8 @Entity
9 @XmlRootElement
10 public class Usuario implements Serializable {
11
12     /**
13      *
14      */
15     private static final long serialVersionUID = 1L;
16
17     private long id;
18     private String nome;
19     private String email;
20     private String telefone;
21
22
23
24
25
26 }
```

Atenção nos imports

Para nosso exemplo usaremos esses atributos

# Criando atributos

O Entity é um annotation

do Hibernate que simboliza que o mesmo nome da classe sera o da tabela no banco

O XMLRootElement é um annotation que usaremos no Jersey

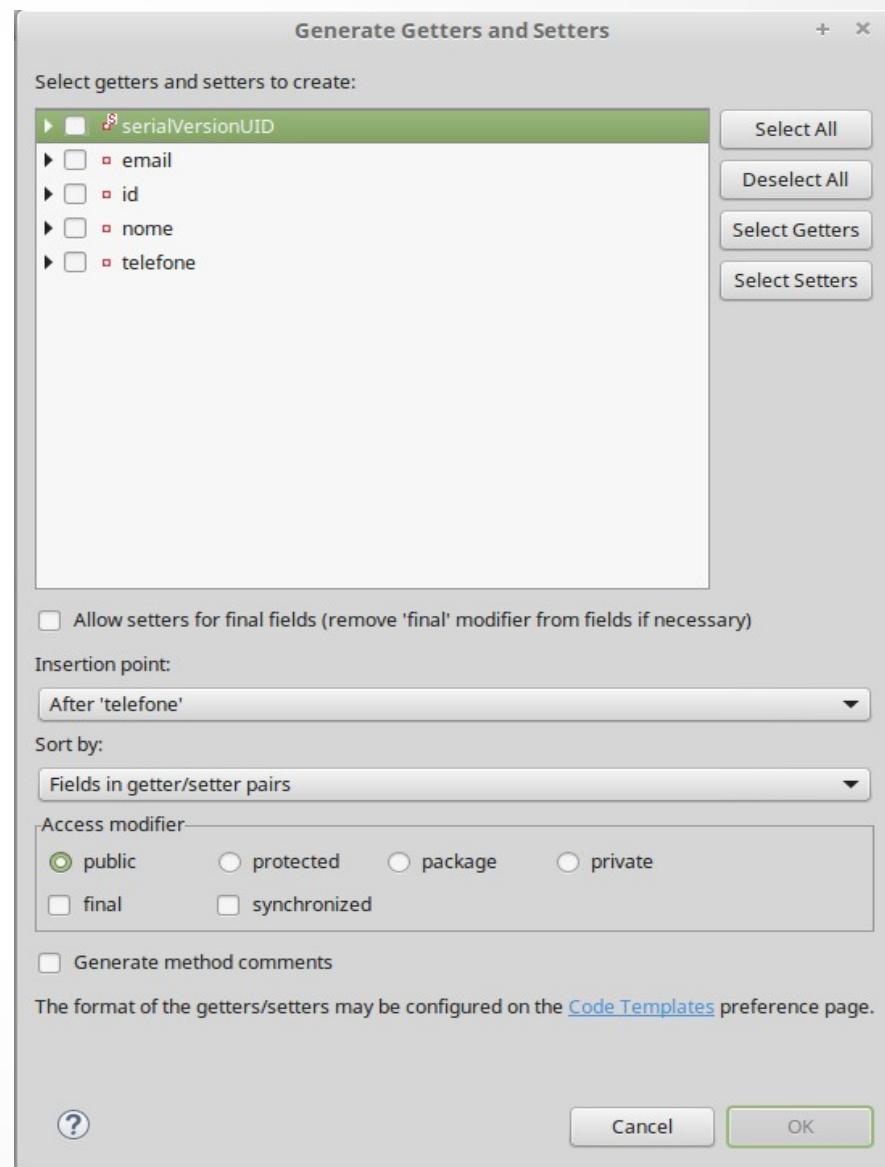
```
1 package br.com.csm.entidade;
2
3 import java.io.Serializable;
4
5 import javax.persistence.Entity;
6 import javax.xml.bind.annotation.XmlRootElement;
7
8 @Entity
9 @XmlRootElement
10 public class Usuario implements Serializable {
11
12     /**
13      *
14      */
15     private static final long serialVersionUID = 1L;
16
17     private long id;
18     private String nome;
19     private String email;
20     private String telefone;
21
22
23
24
25
26 }
```

Atenção nos imports

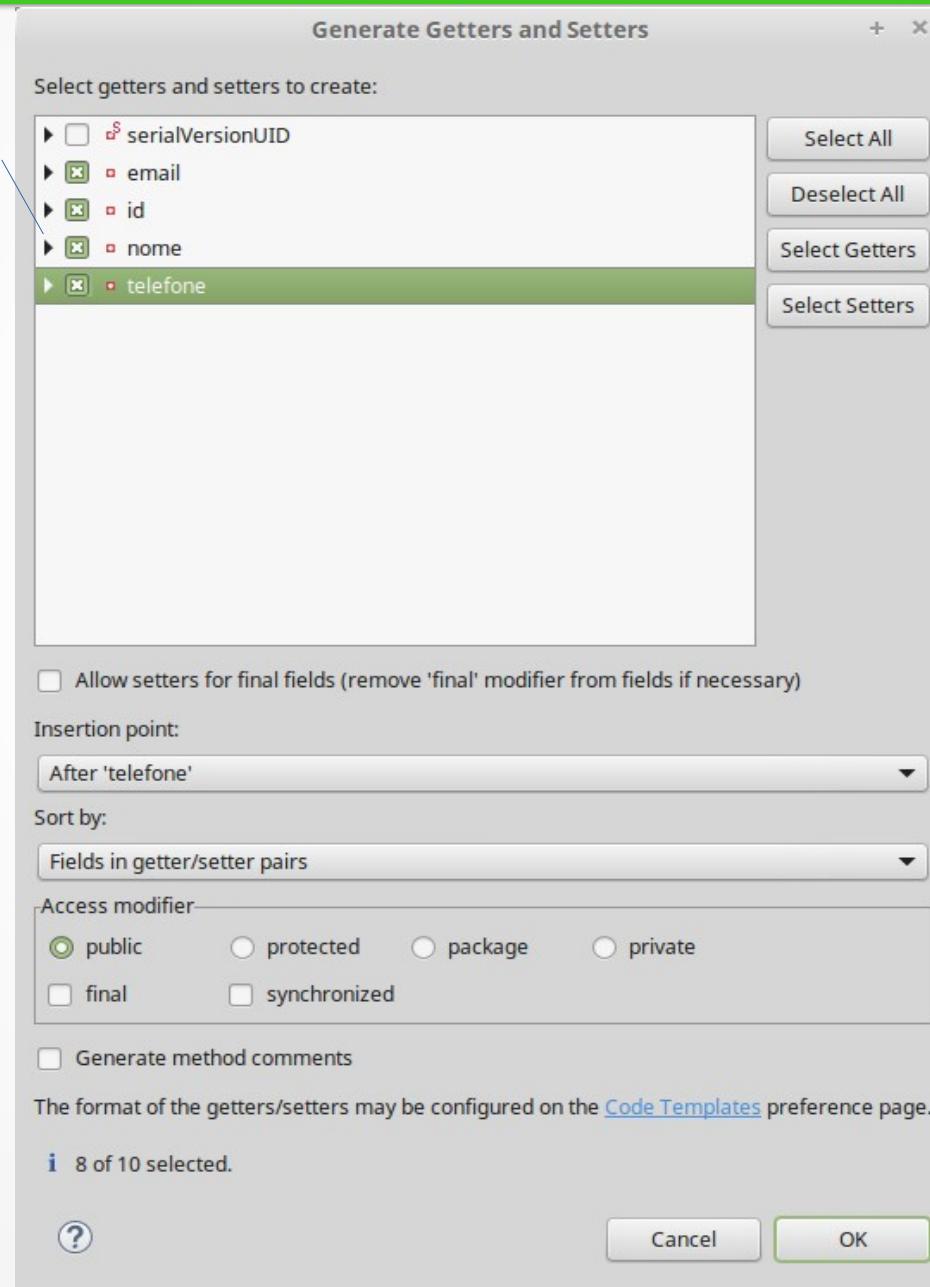
Para nosso exemplo usaremos esses atributos

Vamos criar os gets e sets

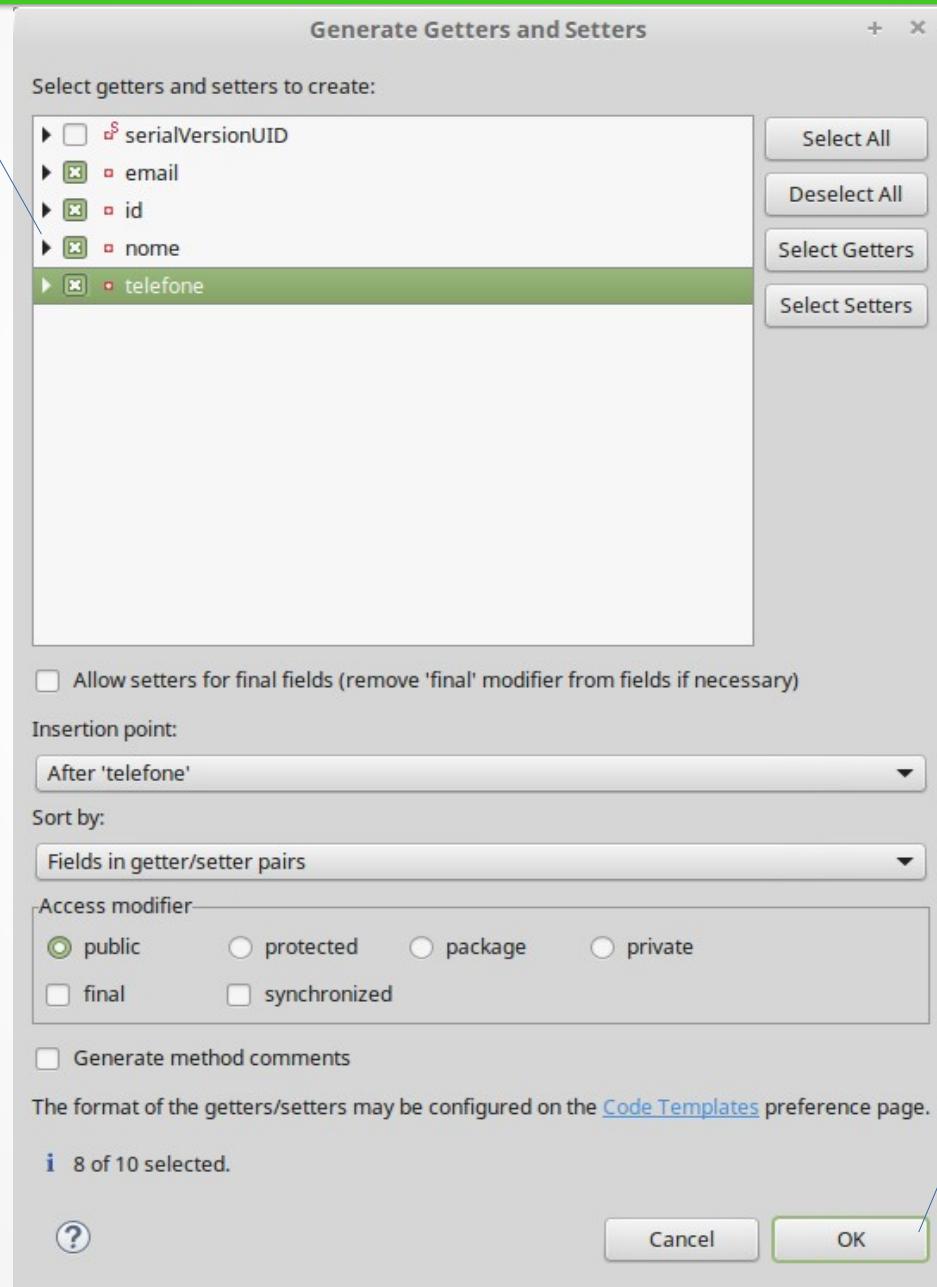
Pressione ctrl+3 e  
digite ggas



Marque as seguintes opções



Marque as seguintes opções



Clique em OK

```
3 | @Entity
4 | @XmlRootElement
5 | public class Usuario implements Serializable {
6 |
7 |     /**
8 |      *
9 |      */
10 |     private static final long serialVersionUID = 1L;
11 |
12 |     private long id;
13 |     private String nome;
14 |     private String email;
15 |     private String telefone;
16 |
17 |     /**
18 |      * 
19 |      */
20 |     public long getId() {
21 |         return id;
22 |     }
23 |     public void setId(long id) {
24 |         this.id = id;
25 |     }
26 |     public String getNome() {
27 |         return nome;
28 |     }
29 |     public void setNome(String nome) {
30 |         this.nome = nome;
31 |     }
32 |     public String getEmail() {
33 |         return email;
34 |     }
35 |     public void setEmail(String email) {
36 |         this.email = email;
37 |     }
38 |     public String getTelefone() {
39 |         return telefone;
40 |     }
41 |     public void setTelefone(String telefone) {
42 |         this.telefone = telefone;
43 |     }
44 |
45 |
46 |
47 |
48 | }
```

```
3 @Entity
4 @XmlRootElement
5 public class Usuario implements Serializable {
6
7     /**
8      *
9      */
10    private static final long serialVersionUID = 1L;
11
12    private long id;
13    private String nome;
14    private String email;
15    private String telefone;
16
17    /**
18     * 
19     */
20    public long getId() {
21        return id;
22    }
23    public void setId(long id) {
24        this.id = id;
25    }
26    public String getNome() {
27        return nome;
28    }
29    public void setNome(String nome) {
30        this.nome = nome;
31    }
32    public String getEmail() {
33        return email;
34    }
35    public void setEmail(String email) {
36        this.email = email;
37    }
38    public String getTelefone() {
39        return telefone;
40    }
41    public void setTelefone(String telefone) {
42        this.telefone = telefone;
43    }
44
45
46}
```

Precisamos agora inserir as notações do hibernate, para que os atributos vierem colunas no banco

```
2  
3 import java.io.Serializable;  
4  
5 import javax.persistence.Column;  
6 import javax.persistence.Entity;  
7 import javax.persistence.GeneratedValue;  
8 import javax.persistence.Id;  
9 import javax.xml.bind.annotation.XmlRootElement;  
10  
11 @Entity  
12 @XmlRootElement  
13 public class Usuario implements Serializable {  
14  
15     /**  
16      *  
17      */  
18     private static final long serialVersionUID = 1L;  
19  
20     @Id  
21     @GeneratedValue  
22     private long id;  
23     @Column  
24     private String nome;  
25     @Column  
26     private String email;  
27     @Column  
28     private String telefone;  
29  
30  
31  
32     //gets and setters....  
33  
34 |
```

A notação @Id é para dizer que esse atributo será nosso indentificador

```
2  
3  import java.io.Serializable;  
4  
5  import javax.persistence.Column;  
6  import javax.persistence.Entity;  
7  import javax.persistence.GeneratedValue;  
8  import javax.persistence.Id;  
9  import javax.xml.bind.annotation.XmlRootElement;  
10  
11 @Entity  
12 @XmlRootElement  
13 public class Usuario implements Serializable {  
14  
15  /**  
16  *  
17  */  
18  private static final long serialVersionUID = 1L;  
19  
20  @Id  
21  @GeneratedValue  
22  private long id;  
23  @Column  
24  private String nome;  
25  @Column  
26  private String email;  
27  @Column  
28  private String telefone;  
29  
30  
31  
32  //gets and setters....  
33  
34 |
```

A notação @Id é para dizer que esse atributo será nosso indentificador

Esse atributo será auto-incremento

```
2  
3  import java.io.Serializable;  
4  
5  import javax.persistence.Column;  
6  import javax.persistence.Entity;  
7  import javax.persistence.GeneratedValue;  
8  import javax.persistence.Id;  
9  import javax.xml.bind.annotation.XmlRootElement;  
10  
11 @Entity  
12 @XmlRootElement  
13 public class Usuario implements Serializable {  
14  
15  /**  
16  *  
17  */  
18  private static final long serialVersionUID = 1L;  
19  
20  @Id  
21  @GeneratedValue  
22  private long id;  
23  @Column  
24  private String nome;  
25  @Column  
26  private String email;  
27  @Column  
28  private String telefone;  
29  
30  
31  
32  //gets and setters....  
33  
34 |
```

A notação @Id é para dizer que esse atributo será nosso indentificador

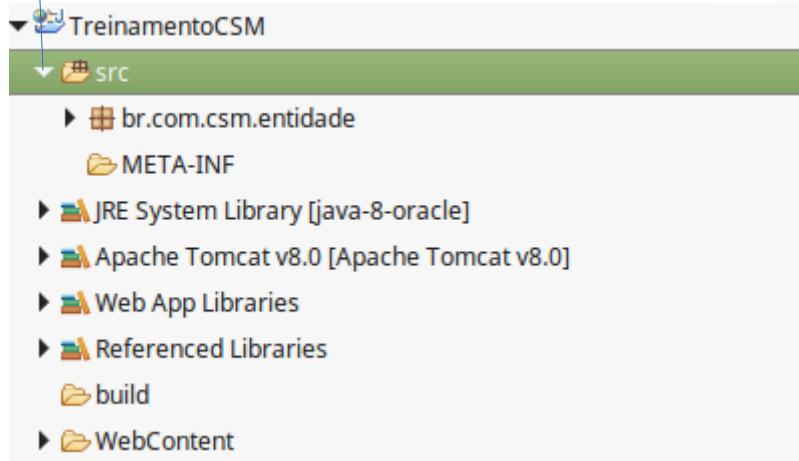
Esse atributo será auto-incremento

```
2  
3  import java.io.Serializable;  
4  
5  import javax.persistence.Column;  
6  import javax.persistence.Entity;  
7  import javax.persistence.GeneratedValue;  
8  import javax.persistence.Id;  
9  import javax.xml.bind.annotation.XmlRootElement;  
10  
11 @Entity  
12 @XmlRootElement  
13 public class Usuario implements Serializable {  
14  
15  /**  
16  *  
17  */  
18  private static final long serialVersionUID = 1L;  
19  
20  @Id  
21  @GeneratedValue  
22  private long id;  
23  @Column  
24  private String nome;  
25  @Column  
26  private String email;  
27  @Column  
28  private String telefone;  
29  
30  
31  
32  //gets and setters....  
33  
34 |
```

@Column e para dizer que Os demais campos São colunas

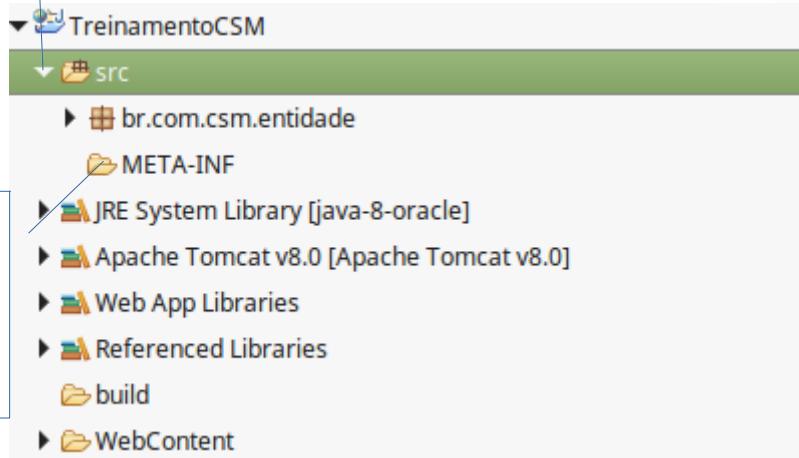
# Antes de prosseguir vamos configurar nossa conexão...

Dentro da pasta src crie uma pasta chamada META-INF



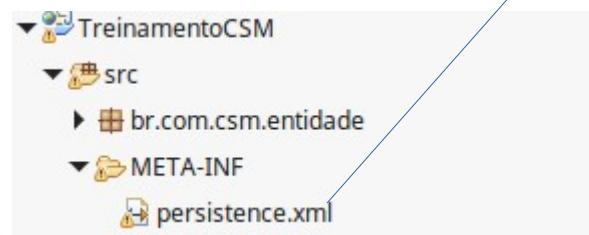
# Antes de prosseguir vamos configurar nossa conexão...

Dentro da pasta src crie uma pasta chamada META-INF



E dentro da pasta META-INF crie um arquivo .xml chamado persistence.xml

Abra o arquivo e preencha com  
seguinte conteúdo





The screenshot shows a Java IDE interface with two tabs open: 'Usuario.java' and 'persistence.xml'. The 'persistence.xml' tab is currently active, displaying the XML configuration for a persistence unit named 'DB'. The code is color-coded, with tags in blue and values in black. A warning icon is visible next to the first line of the XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence">
    <persistence-unit name="DB" transaction-type="RESOURCE_LOCAL">
        <provider>org.hibernate.ejb.HibernatePersistence</provider>
        <properties>
            <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
            <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost/NOME_DO_SEU_BANCO_DE_DADOS"/>
            <property name="javax.persistence.jdbc.user" value="USUARIO_DE_CONEXAO_COM_O_BANCO"/>
            <property name="javax.persistence.jdbc.password" value="SENHA_DE_CONEXAO_COM_O_BANCO"/>
            <property name="hibernate.hbm2ddl.auto" value="update"/>
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.format_sql" value="true" />
            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5InnoDBDialect"/>
        </properties>
    </persistence-unit>
</persistence>
```

Esse nome será  
usado pelo  
JPA mais tarde

Coloque o nome do seu  
banco de dados

```
1<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence">
2  <persistence-unit name="DB" transaction-type="RESOURCE_LOCAL">
3
4    <provider>org.hibernate.ejb.HibernatePersistence</provider>
5
6    <properties>
7      <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
8      <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost/NOME_DO_SEU_BANCO_DE_DADOS"/>
9      <property name="javax.persistence.jdbc.user" value="USUARIO_DE_CONEXAO_COM_O_BANCO"/>
10     <property name="javax.persistence.jdbc.password" value="SENHA_DE_CONEXAO_COM_O_BANCO"/>
11
12     <property name="hibernate.hbm2ddl.auto" value="update"/>
13     <property name="hibernate.show_sql" value="true" />
14     <property name="hibernate.format_sql" value="true" />
15     <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5InnoDBDialect"/>
16   </properties>
17 </persistence-unit>
18 </persistence>
```

Usuário de acesso  
Ao banco

Senha de acesso  
Ao banco

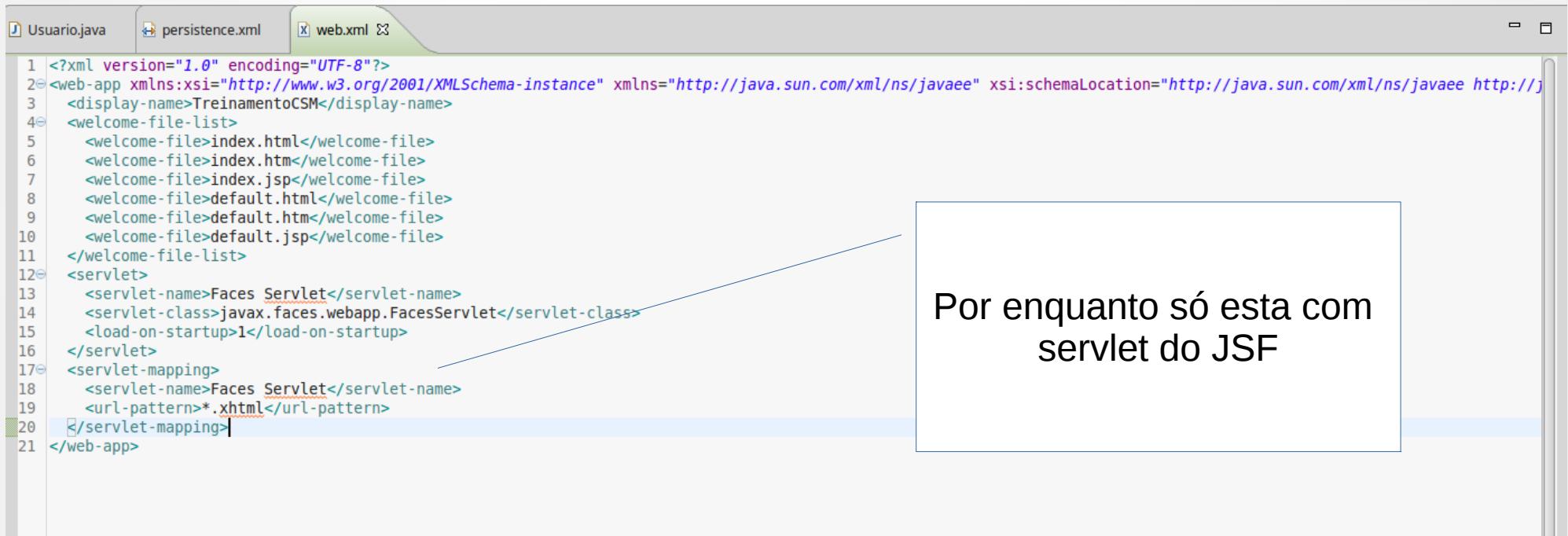
# Agora vamos para o arquivo web.xml



The screenshot shows a code editor window with three tabs at the top: 'Usuario.java', 'persistence.xml', and 'web.xml'. The 'web.xml' tab is active, displaying XML configuration code. The code defines a web application with various welcome files, a servlet named 'Faces Servlet' with its class set to 'javax.faces.webapp.FacesServlet', and a servlet mapping for '.xhtml' URLs.

```
<?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_2_5.xsd" version="2.5">
    <display-name>TreinamentoCSM</display-name>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>default.html</welcome-file>
        <welcome-file>default.htm</welcome-file>
        <welcome-file>default.jsp</welcome-file>
    </welcome-file-list>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.xhtml</url-pattern>
    </servlet-mapping>
</web-app>
```

# Agora vamos para o arquivo web.xml

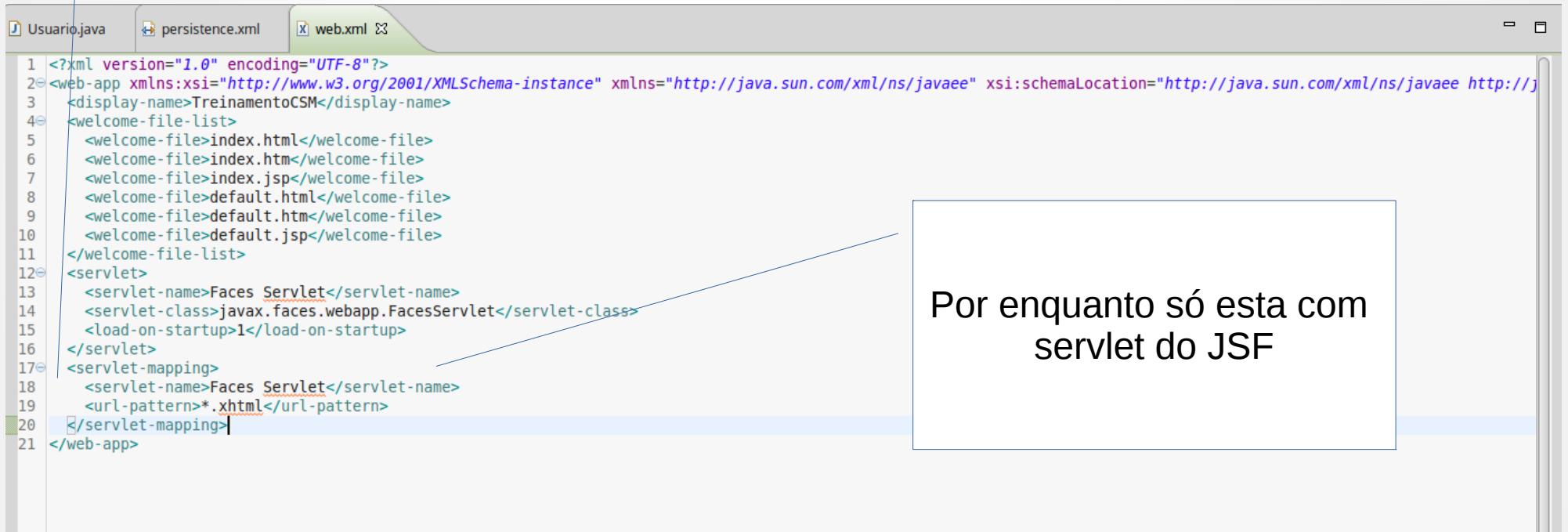


```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://j
3   <display-name>TreinamentoCSM</display-name>
4   <welcome-file-list>
5     <welcome-file>index.html</welcome-file>
6     <welcome-file>index.htm</welcome-file>
7     <welcome-file>index.jsp</welcome-file>
8     <welcome-file>default.html</welcome-file>
9     <welcome-file>default.htm</welcome-file>
10    <welcome-file>default.jsp</welcome-file>
11  </welcome-file-list>
12  <servlet>
13    <servlet-name>Faces Servlet</servlet-name>
14    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
15    <load-on-startup>1</load-on-startup>
16  </servlet>
17  <servlet-mapping>
18    <servlet-name>Faces Servlet</servlet-name>
19    <url-pattern>*.xhtml</url-pattern>
20  </servlet-mapping>
21 </web-app>
```

Por enquanto só esta com servlet do JSF

# Agora vamos para o arquivo web.xml

Vamos adicionar a servlet do Jersey



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <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">
3   <display-name>TreinamentoCSM</display-name>
4   <welcome-file-list>
5     <welcome-file>index.html</welcome-file>
6     <welcome-file>index.htm</welcome-file>
7     <welcome-file>index.jsp</welcome-file>
8     <welcome-file>default.html</welcome-file>
9     <welcome-file>default.htm</welcome-file>
10    <welcome-file>default.jsp</welcome-file>
11  </welcome-file-list>
12  <servlet>
13    <servlet-name>Faces Servlet</servlet-name>
14    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
15    <load-on-startup>1</load-on-startup>
16  </servlet>
17  <servlet-mapping>
18    <servlet-name>Faces Servlet</servlet-name>
19    <url-pattern>*.xhtml</url-pattern>
20  </servlet-mapping>
21</web-app>
```

Por enquanto só esta com  
servlet do JSF

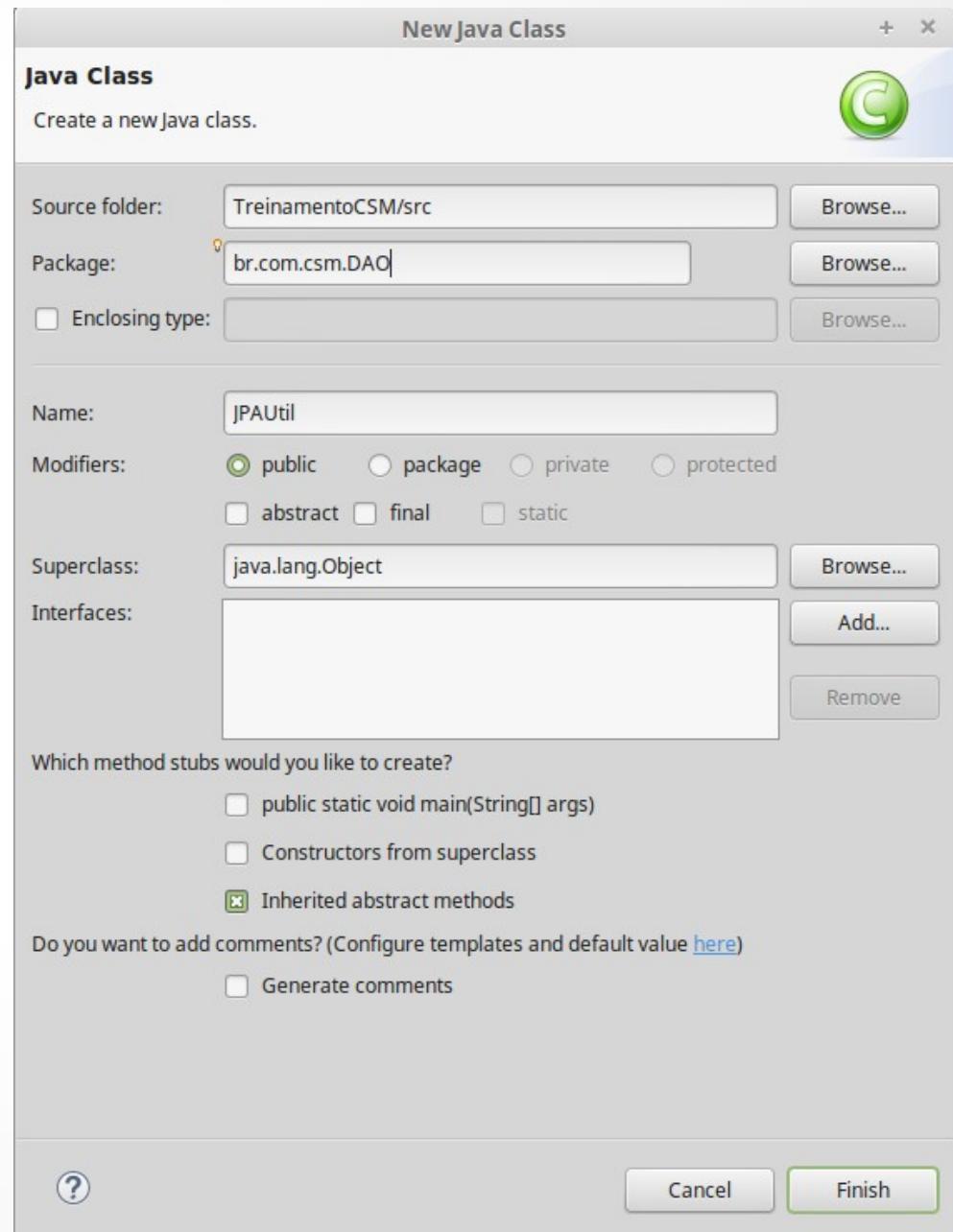
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://
3 <display-name>TreinamentoCSM</display-name>
4 <welcome-file-list>
5   <welcome-file>index.html</welcome-file>
6   <welcome-file>index.htm</welcome-file>
7   <welcome-file>index.jsp</welcome-file>
8   <welcome-file>default.html</welcome-file>
9   <welcome-file>default.htm</welcome-file>
10  <welcome-file>default.jsp</welcome-file>
11 </welcome-file-list>
12 <servlet>
13   <servlet-name>Faces Servlet</servlet-name>
14   <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
15   <load-on-startup>1</load-on-startup>
16 </servlet>
17 <servlet-mapping>
18   <servlet-name>Faces Servlet</servlet-name>
19   <url-pattern>*.xhtml</url-pattern>
20 </servlet-mapping>
21
22
23 <servlet>
24   <servlet-name>Jersey REST Service</servlet-name>
25   <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
26   <init-param>
27     <param-name>com.sun.jersey.config.property.packages</param-name>
28     <param-value>br.com.csm.resources</param-value>
29   </init-param>
30   <load-on-startup>1</load-on-startup>
31 </servlet>
32 <servlet-mapping>
33   <servlet-name>Jersey REST Service</servlet-name>
34   <url-pattern>/ws/*</url-pattern>
35 </servlet-mapping>
36 </web-app>
```

Adicione o  
código abaixo

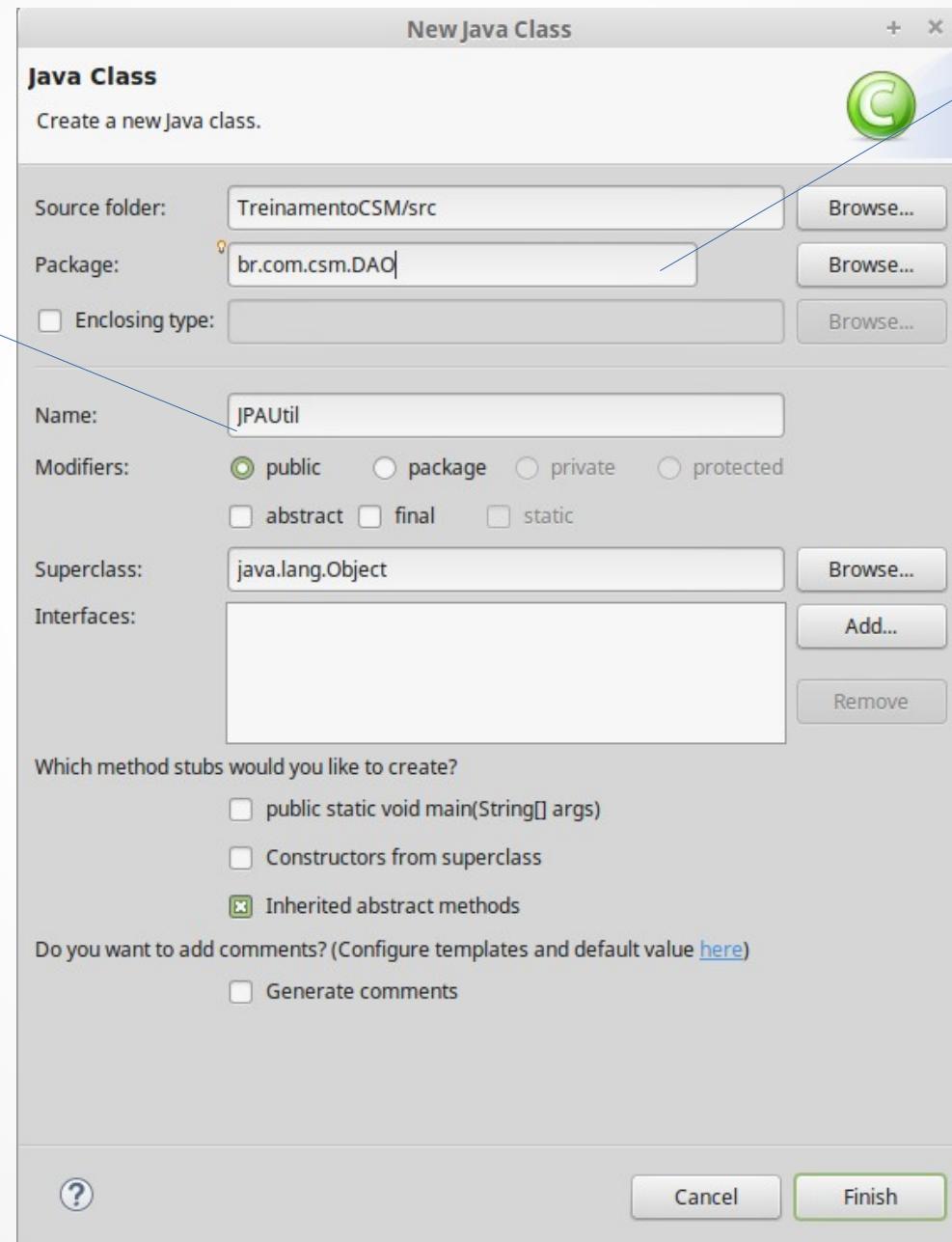
Esse br.com.csm.resources  
Será explicado mais tarde

# Vamos criar nossa JPAUtil

- Crie uma classe chamada JPAUtil

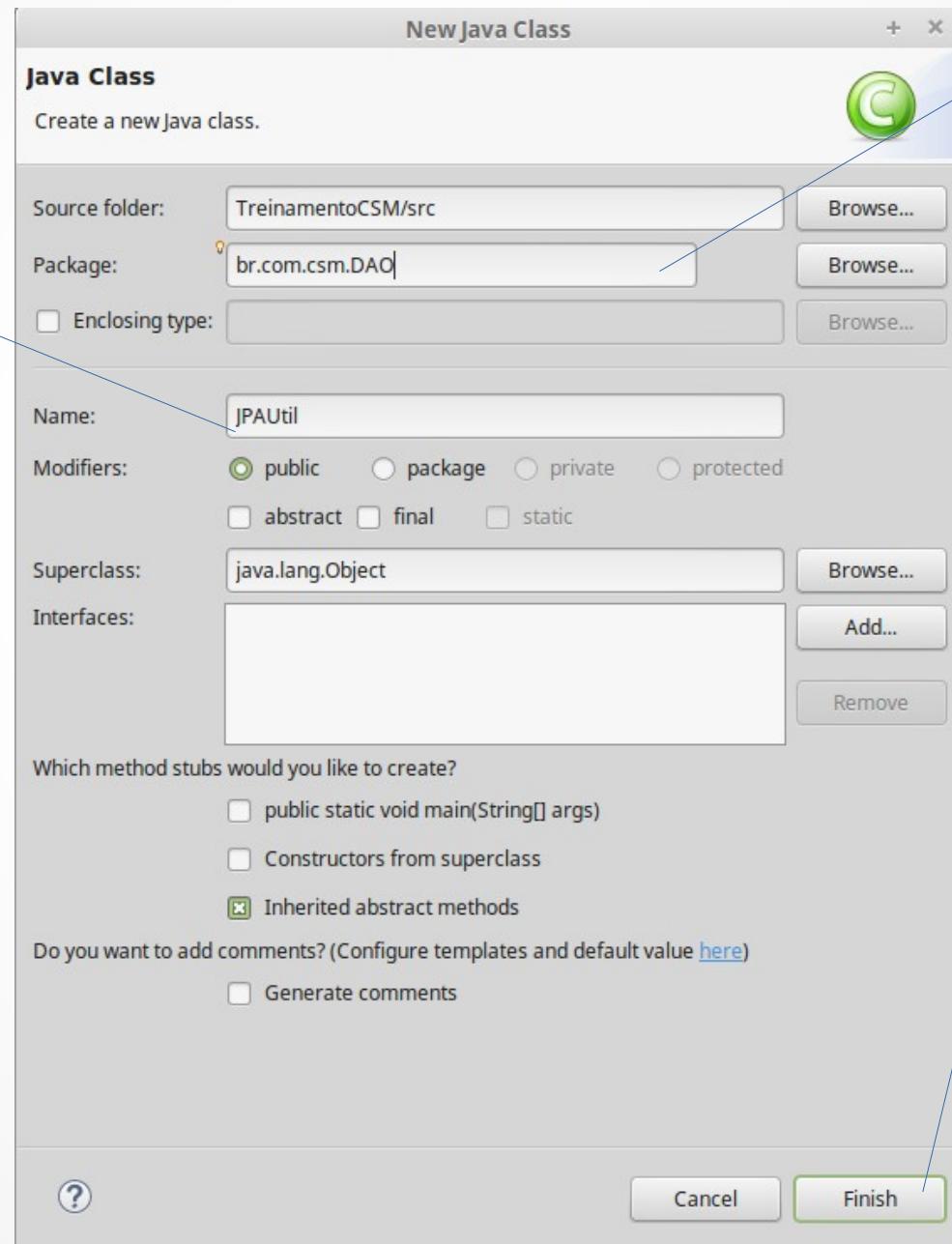


Nome da Classe



Nome do package

Nome da Classe



Nome do package

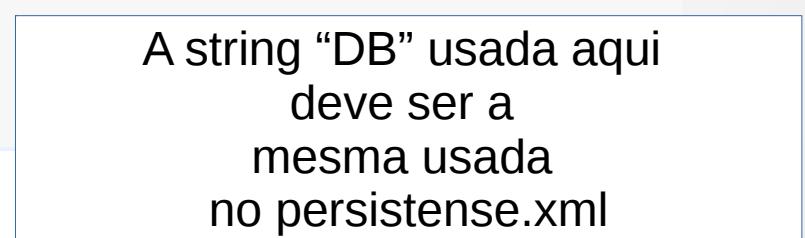
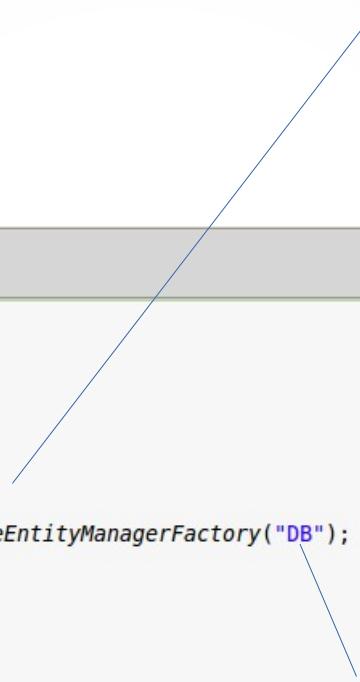
Clique em  
Finish

Preencha a classe JPAUtil  
Com o seguinte conteúdo



```
1 package br.com.csm.DAO;
2
3 import javax.persistence.EntityManager;
4 import javax.persistence.EntityManagerFactory;
5 import javax.persistence.Persistence;
6
7 public class JPAUtil {
8
9     private static EntityManagerFactory emf = Persistence.createEntityManagerFactory("DB");
10
11
12     public static EntityManager getEntityManager(){
13         return emf.createEntityManager();
14     }
15
16
17 }
```

Preencha a classe JPAUtil  
Com o seguinte conteúdo



```
1 package br.com.csm.DAO;
2
3 import javax.persistence.EntityManager;
4 import javax.persistence.EntityManagerFactory;
5 import javax.persistence.Persistence;
6
7 public class JPAUtil {
8
9     private static EntityManagerFactory emf = Persistence.createEntityManagerFactory("DB");
10
11
12     public static EntityManager getEntityManager(){
13         return emf.createEntityManager();
14     }
15
16
17 }
```

A string “DB” usada aqui  
deve ser a  
mesma usada  
no persistense.xml



A screenshot of a Java IDE showing the `persistence.xml` file. A blue circle highlights the `name="DB"` attribute in the `<persistence-unit>` element. The code is as follows:

```
1<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence">
2<persistence-unit name="DB" transaction-type="RESOURCE_LOCAL">
3
```

Preencha a classe JPAUtil  
Com o seguinte conteúdo



A screenshot of a Java IDE showing the `JPAUtil.java` code. A blue line points from the circled `name="DB"` in the XML to the string `"DB"` in the Java code. The code is as follows:

```
1 package br.com.csm.DAO;
2
3 import javax.persistence.EntityManager;
4 import javax.persistence.EntityManagerFactory;
5 import javax.persistence.Persistence;
6
7 public class JPAUtil {
8
9     private static EntityManagerFactory emf = Persistence.createEntityManagerFactory("DB");
10
11
12     public static EntityManager getEntityManager(){
13         return emf.createEntityManager();
14     }
15
16
17 }
```

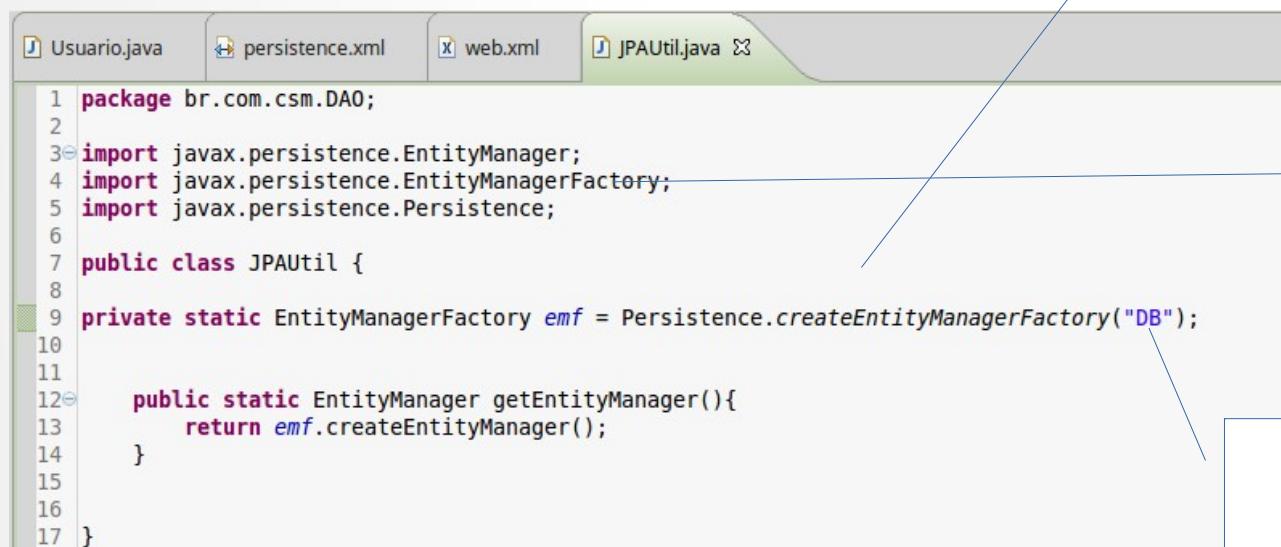
A string “DB” usada aqui  
deve ser a  
mesma usada  
no persistense.xml



A screenshot of a Java IDE showing the `persistence.xml` file. A blue circle highlights the `name="DB"` attribute in the `<persistence-unit>` element. The code is as follows:

```
1<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence">
2<persistence-unit name="DB" transaction-type="RESOURCE_LOCAL">
3
```

Preencha a classe JPAUtil  
Com o seguinte conteúdo



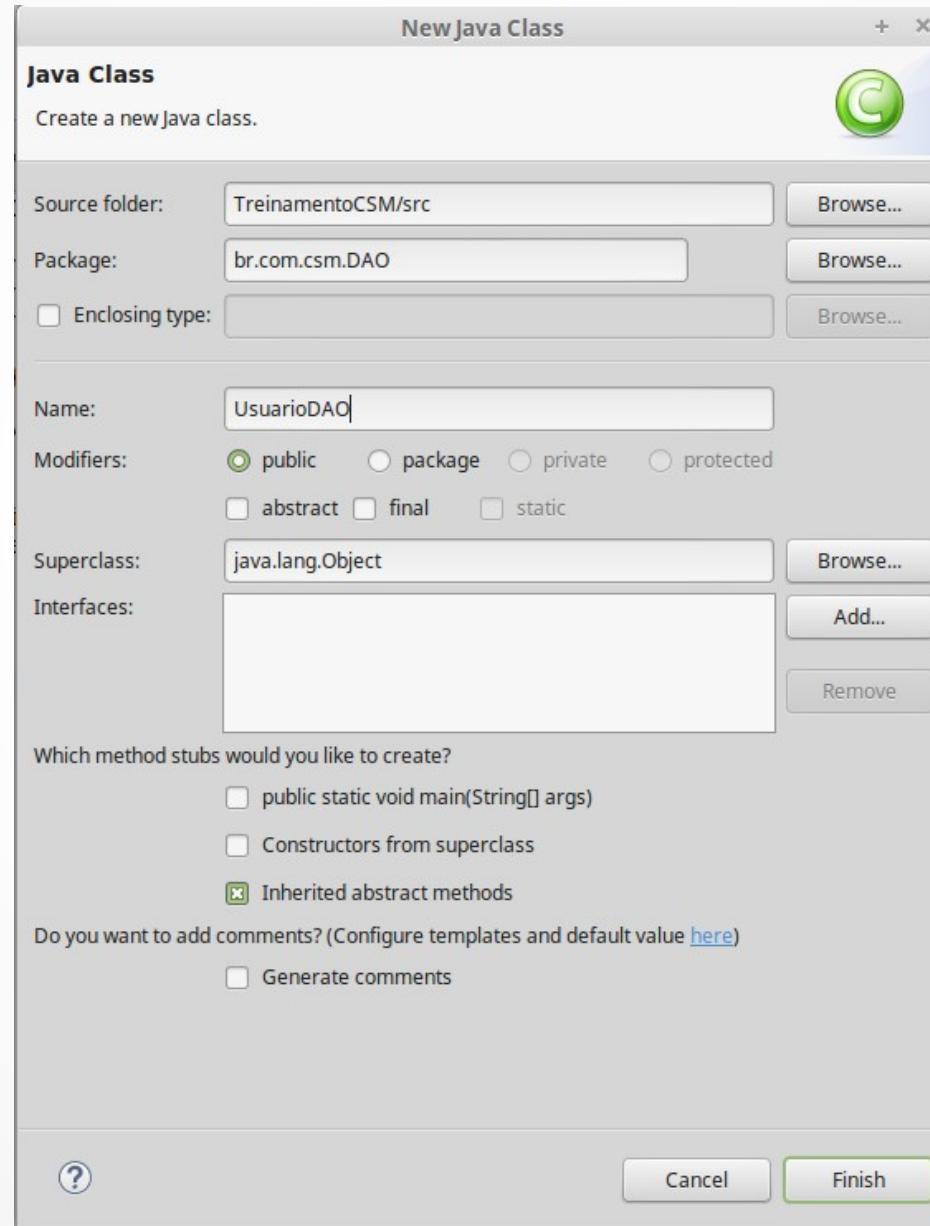
A screenshot of a Java IDE showing the `JPAUtil.java` code. The code defines a static factory method to create an EntityManagerFactory. A blue line points from the circled `name="DB"` in the XML to the string "DB" in the Java code. The code is as follows:

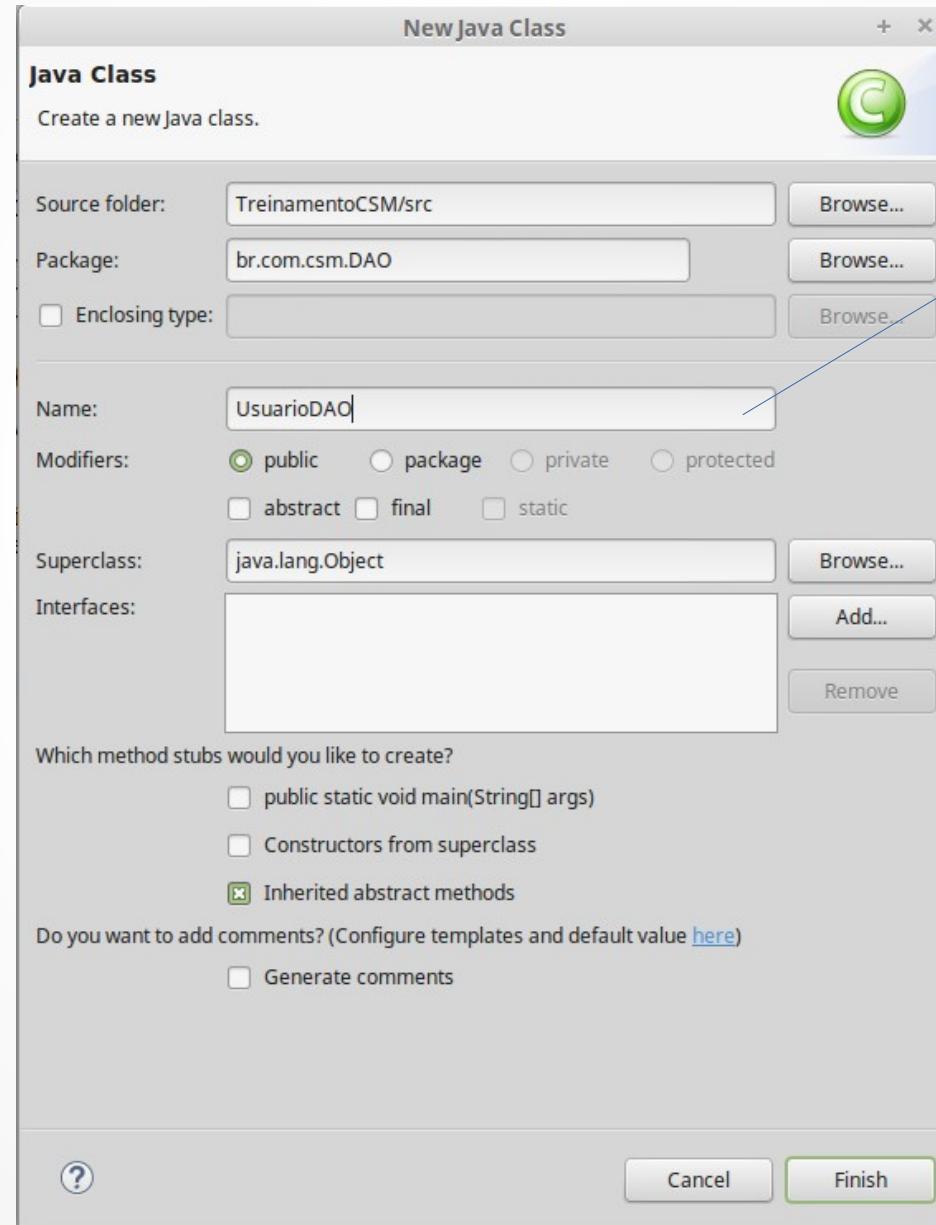
```
1 package br.com.csm.DAO;
2
3 import javax.persistence.EntityManager;
4 import javax.persistence.EntityManagerFactory;
5 import javax.persistence.Persistence;
6
7 public class JPAUtil {
8
9     private static EntityManagerFactory emf = Persistence.createEntityManagerFactory("DB");
10
11
12     public static EntityManager getEntityManager(){
13         return emf.createEntityManager();
14     }
15
16
17 }
```

ATENÇÃO COM  
OS IMPORTS

A string “DB” usada aqui  
deve ser a  
mesma usada  
no persistense.xml

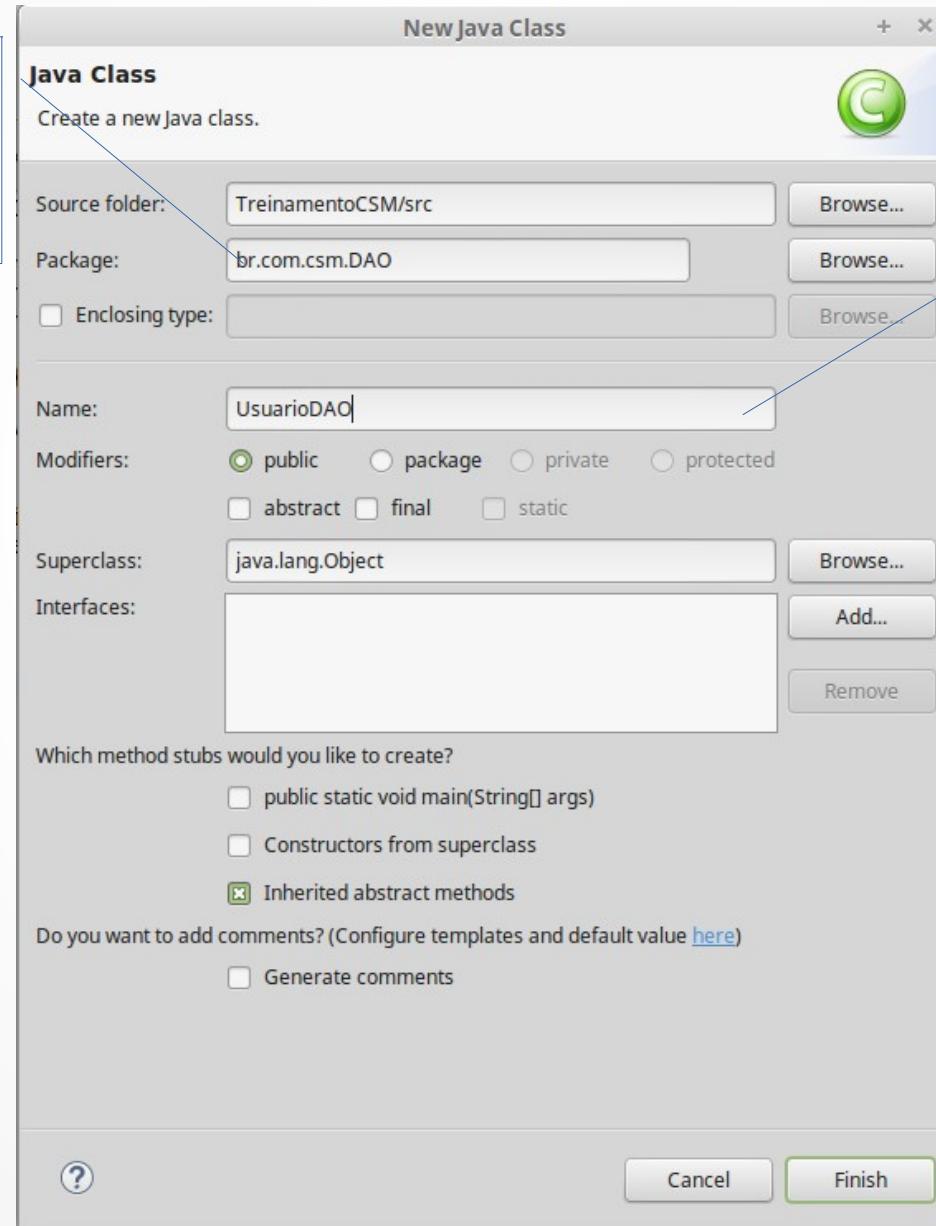
# Vamos criar uma classe chamada UsuarioDAO





Nome da classe

No mesmo package que  
já tinha sido criado



Nome da  
classe

```
1 package br.com.csm.DAO;  
2  
3 public class UsuarioDAO {  
4  
5 }  
6
```

Essa classe é responsável  
Por ter métodos que  
acessam o banco

```
1 package br.com.csm.DAO;  
2  
3 public class UsuarioDAO {  
4  
5 }  
6
```

Vamos criar os  
Métodos...

Essa classe é responsável  
Por ter métodos que  
acessam o banco

```
1 package br.com.csm.DAO;  
2  
3 public class UsuarioDAO {  
4  
5 }  
6
```

```
1 package br.com.csm.DAO;
2
3 import java.util.List;
4
5 import javax.persistence.EntityManager;
6 import javax.persistence.Query;
7
8 import br.com.csm.entidade.Usuario;
9
10 public class UsuarioDAO {
11
12     private EntityManager entityManager;
13
14     public UsuarioDAO(EntityManager entityManager){
15         this.entityManager= entityManager;
16     }
17
18     public void cadastrar(Usuario usuario){
19         entityManager.persist(usuario);
20     }
21
22     public void alterar(Usuario usuario){
23         entityManager.merge(usuario);
24     }
25
26     public void excluir(Usuario usuario){
27         entityManager.remove(entityManager.merge(usuario));
28     }
29
30     public Usuario consultar(Long id){
31         return entityManager.getReference(Usuario.class, id);
32     }
33
34     public List<Usuario> listar(){
35
36         String sql = "Select usu from Usuario usu order by nome";
37         Query query = entityManager.createQuery(sql);
38
39         return query.getResultList();
40     }
41
42 }
43
```

```
1 package br.com.csm.DAO;
2
3 import java.util.List;
4
5 import javax.persistence.EntityManager;
6 import javax.persistence.Query;
7
8 import br.com.csm.entidade.Usuario;
9
10 public class UsuarioDAO {
11
12     private EntityManager entityManager;
13
14     public UsuarioDAO(EntityManager entityManager){
15         this.entityManager= entityManager;
16     }
17
18     public void cadastrar(Usuario usuario){
19         entityManager.persist(usuario);
20     }
21
22     public void alterar(Usuario usuario){
23         entityManager.merge(usuario);
24     }
25
26     public void excluir(Usuario usuario){
27         entityManager.remove(entityManager.merge(usuario));
28     }
29
30     public Usuario consultar(Long id){
31         return entityManager.getReference(Usuario.class, id);
32     }
33
34     public List<Usuario> listar(){
35
36         String sql = "Select usu from Usuario usu order by nome";
37         Query query = entityManager.createQuery(sql);
38
39         return query.getResultList();
40     }
41
42 }
43
```

Temos os métodos básicos fornecidos pelo hibernate

Precisamos de um construtor da classe para passarmos a instância da conexão

```
1 package br.com.csm.DAO;
2
3 import java.util.List;
4
5 import javax.persistence.EntityManager;
6 import javax.persistence.Query;
7
8 import br.com.csm.entidade.Usuario;
9
10 public class UsuarioDAO {
11
12     private EntityManager entityManager;
13
14     public UsuarioDAO(EntityManager entityManager){
15         this.entityManager= entityManager;
16     }
17
18     public void cadastrar(Usuario usuario){
19         entityManager.persist(usuario);
20     }
21
22     public void alterar(Usuario usuario){
23         entityManager.merge(usuario);
24     }
25
26     public void excluir(Usuario usuario){
27         entityManager.remove(entityManager.merge(usuario));
28     }
29
30     public Usuario consultar(Long id){
31         return entityManager.getReference(Usuario.class, id);
32     }
33
34     public List<Usuario> listar(){
35
36         String sql = "Select usu from Usuario usu order by nome";
37         Query query = entityManager.createQuery(sql);
38
39         return query.getResultList();
40     }
41
42 }
43
```

Temos os métodos básicos fornecidos pelo hibernate

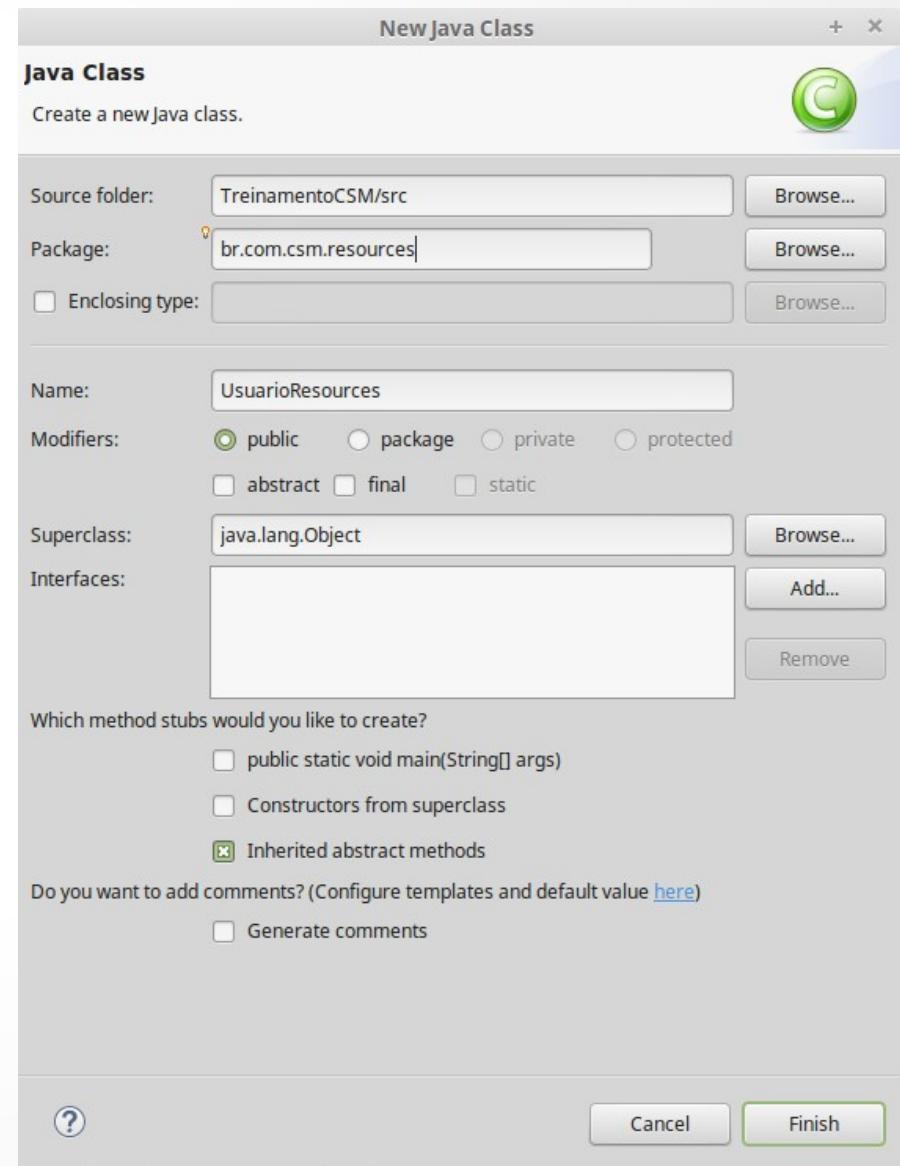
Precisamos de um construtor da classe para passarmos a instância da conexão

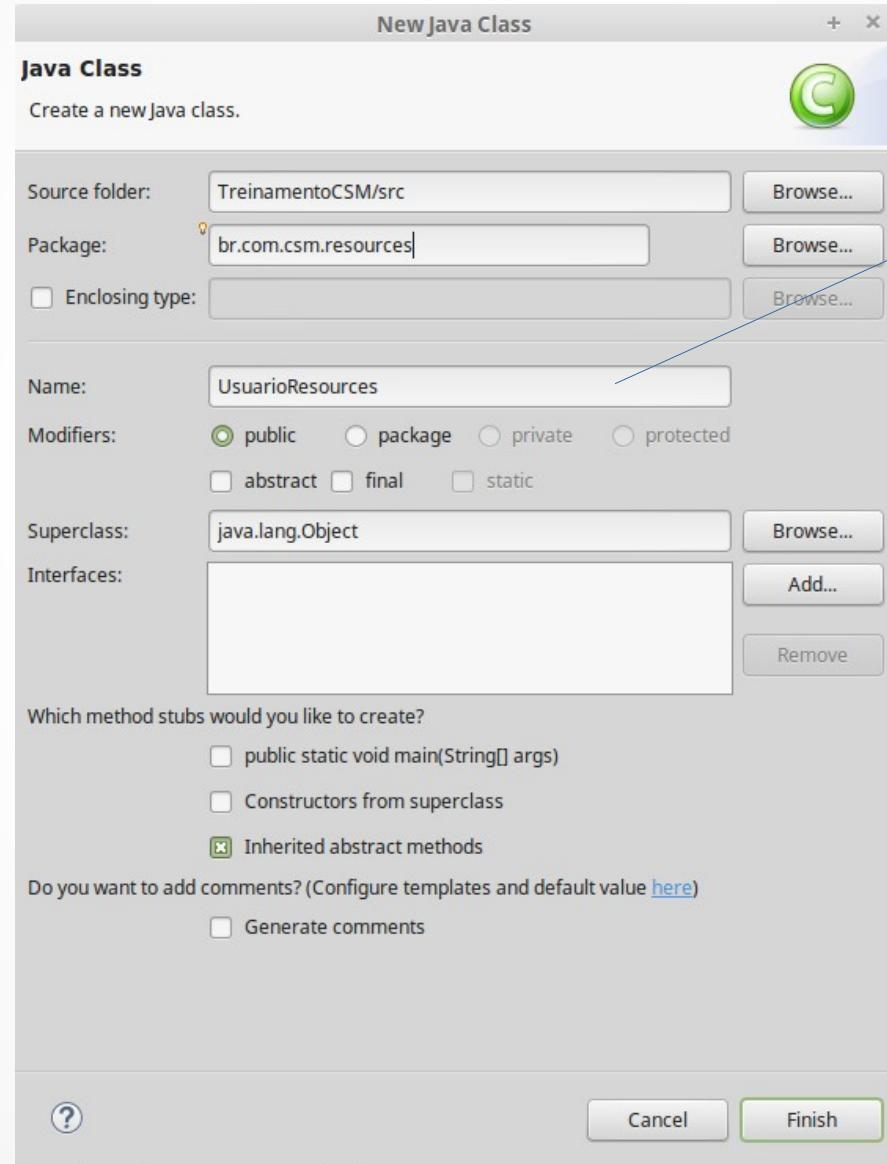
```
1 package br.com.csm.DAO;
2
3 import java.util.List;
4
5 import javax.persistence.EntityManager;
6 import javax.persistence.Query;
7
8 import br.com.csm.entidade.Usuario;
9
10 public class UsuarioDAO {
11
12     private EntityManager entityManager;
13
14     public UsuarioDAO(EntityManager entityManager){
15         this.entityManager= entityManager;
16     }
17
18     public void cadastrar(Usuario usuario){
19         entityManager.persist(usuario);
20     }
21
22     public void alterar(Usuario usuario){
23         entityManager.merge(usuario);
24     }
25
26     public void excluir(Usuario usuario){
27         entityManager.remove(entityManager.merge(usuario));
28     }
29
30     public Usuario consultar(Long id){
31         return entityManager.getReference(Usuario.class, id);
32     }
33
34     public List<Usuario> listar(){
35
36         String sql = "Select usu from Usuario usu order by nome";
37         Query query = entityManager.createQuery(sql);
38
39         return query.getResultList();
40     }
41
42 }
43
```

Temos os métodos básicos fornecidos pelo hibernate

ATENÇÃO NOS IMPORTS

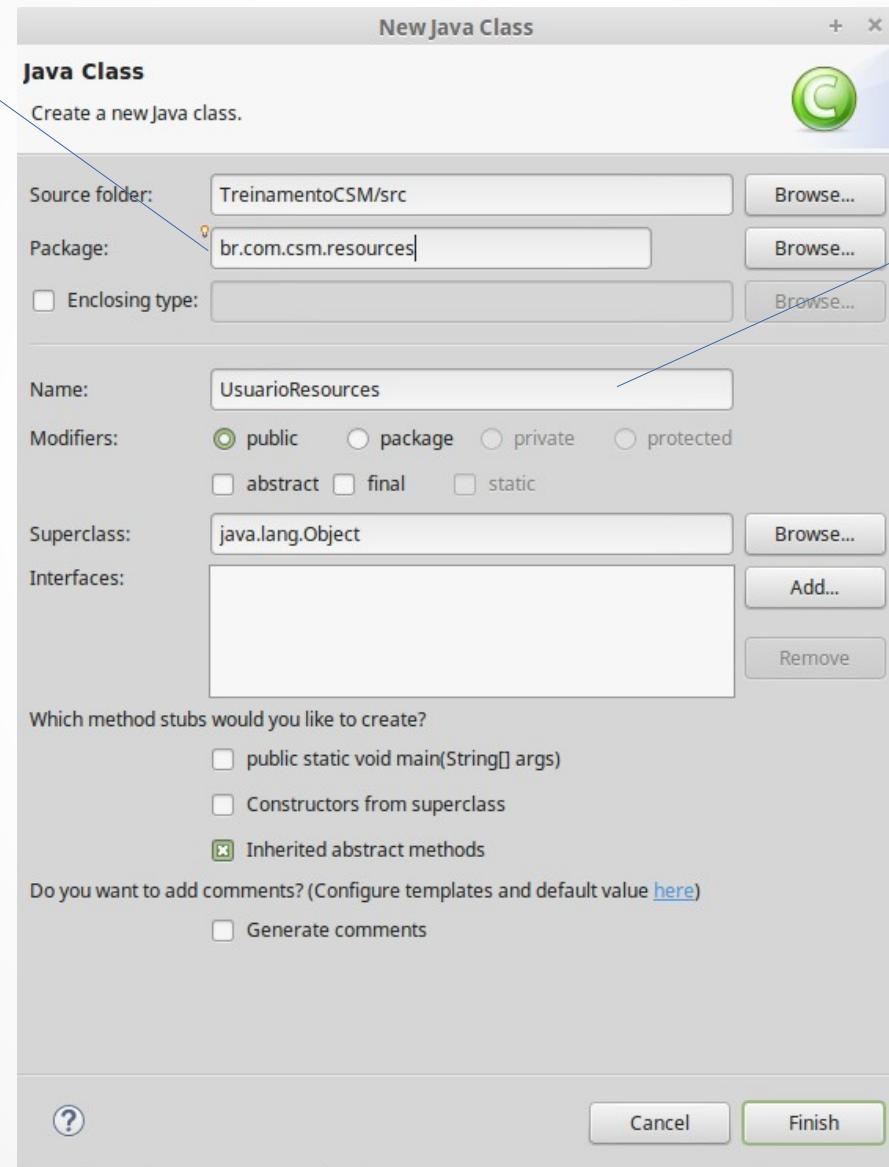
- Vamos criar uma classe chamada UsuarioResources





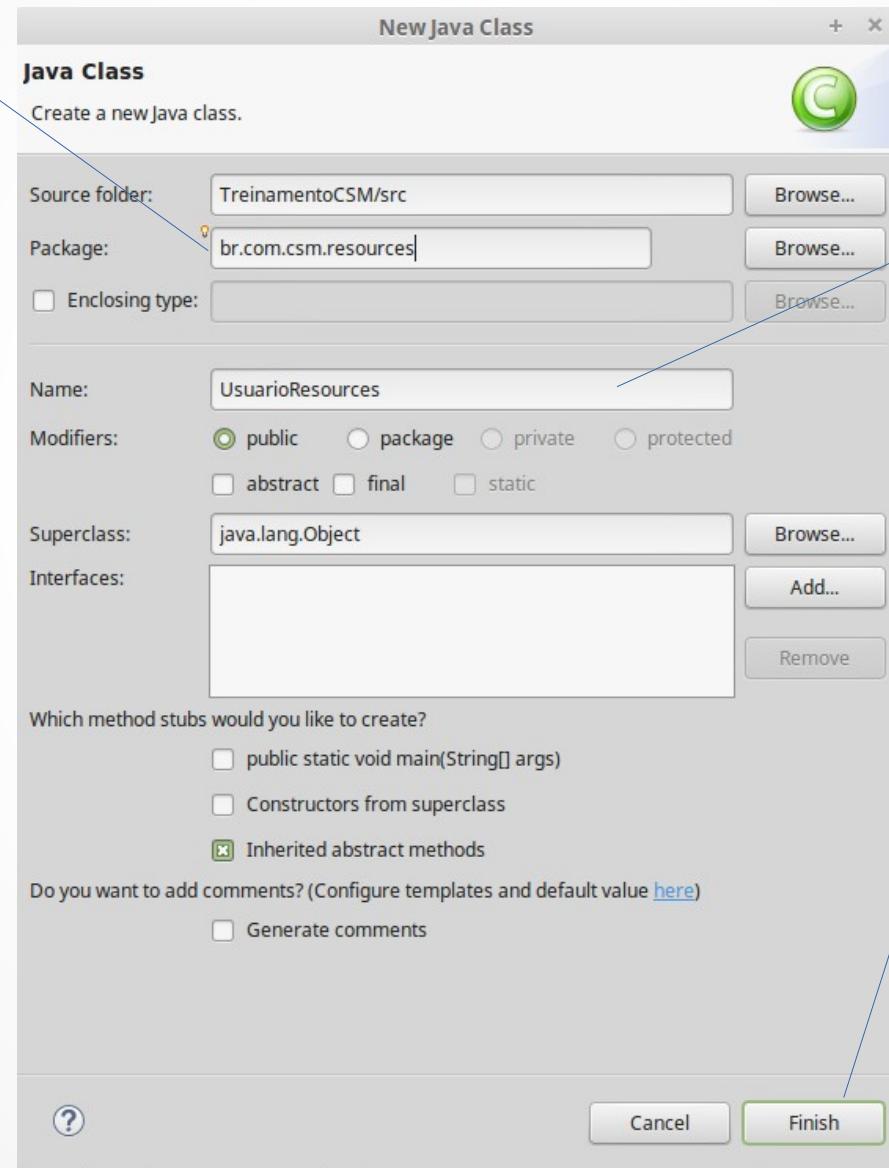
Nome da classe

Nome do package



Nome da classe

Nome do package



Nome da classe

Clique em Finish

```
1 package br.com.csm.resources;
2
3 public class UsuarioResources {
4
5 }
6
```

Essa classe é  
responsável  
por receber  
as requisições de Rest

```
1 package br.com.csm.resources;  
2  
3 public class UsuarioResources {  
4  
5 }  
6
```

Essa classe é responsável por receber as requisições de Rest

Precisaremos adicionar as notações do Jersey

```
1 package br.com.csm.resources;  
2  
3 public class UsuarioResources {  
4  
5 }  
6
```

Essa classe é responsável por receber as requisições de Rest

Precisaremos adicionar as notações do Jersey

```
1 package br.com.csm.resources;  
2  
3 public class UsuarioResources {  
4  
5 }  
6
```

Uma vantagem é que o Jersey deixa o código bem simples

```
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import javax.persistence.EntityManager;
7 import javax.ws.rs.Consumes;
8 import javax.ws.rs.GET;
9 import javax.ws.rs.POST;
10 import javax.ws.rs.Path;
11 import javax.ws.rs.Produces;
12 import javax.ws.rs.core.MediaType;
13
14 import br.com.csm.DAO.JPAUtil;
15 import br.com.csm.DAO.UsuarioDAO;
16 import br.com.csm.entidade.Usuario;
17
18 @Path("/usuarios")
19 public class UsuarioResources {
20
21
22     @GET
23     @Path("/listarUsuarios")
24     @Consumes(MediaType.APPLICATION_JSON)
25     @Produces(MediaType.APPLICATION_JSON)
26     public List<Usuario> listarUsuarios(){
27         List<Usuario> list = new ArrayList<Usuario>();
28         EntityManager em = JPAUtil.getEntityManager();
29         UsuarioDAO dao = new UsuarioDAO(em);
30         list=dao.listar();
31
32         return list;
33     }
34
35
36     @POST
37     @Path("/consultarUsuario")
38     @Consumes(MediaType.APPLICATION_JSON)
39     @Produces(MediaType.APPLICATION_JSON)
40     public Usuario buscarUsuario(long id){
41         Usuario usuario = new Usuario();
42         EntityManager em = JPAUtil.getEntityManager();
43         UsuarioDAO dao = new UsuarioDAO(em);
44         usuario = dao.consultar(id);
45
46         return usuario;
47     }
48
49
50 }
```

```
2  
3 import java.util.ArrayList;  
4 import java.util.List;  
5  
6 import javax.persistence.EntityManager;  
7 import javax.ws.rs.Consumes;  
8 import javax.ws.rs.GET;  
9 import javax.ws.rs.POST;  
10 import javax.ws.rs.Path;  
11 import javax.ws.rs.Produces;  
12 import javax.ws.rs.core.MediaType;  
13  
14 import br.com.csm.DAO.JPAUtil;  
15 import br.com.csm.DAO.UsuarioDAO;  
16 import br.com.csm.entidade.Usuario;  
17  
18 @Path("/usuarios")  
19 public class UsuarioResources {  
20  
21  
22     @GET  
23     @Path("/listarUsuarios")  
24     @Consumes(MediaType.APPLICATION_JSON)  
25     @Produces(MediaType.APPLICATION_JSON)  
26     public List<Usuario> listarUsuarios(){  
27         List<Usuario> list = new ArrayList<Usuario>();  
28         EntityManager em = JPAUtil.getEntityManager();  
29         UsuarioDAO dao = new UsuarioDAO(em);  
30         list=dao.listar();  
31  
32         return list;  
33     }  
34  
35  
36     @POST  
37     @Path("/consultarUsuario")  
38     @Consumes(MediaType.APPLICATION_JSON)  
39     @Produces(MediaType.APPLICATION_JSON)  
40     public Usuario buscarUsuario(long id){  
41         Usuario usuario = new Usuario();  
42         EntityManager em = JPAUtil.getEntityManager();  
43         UsuarioDAO dao = new UsuarioDAO(em);  
44         usuario = dao.consultar(id);  
45  
46         return usuario;  
47     }  
48  
49  
50 }
```

Essa notação serve para referenciar qual classe resource iremos trabalhos

```

2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import javax.persistence.EntityManager;
7 import javax.ws.rs.Consumes;
8 import javax.ws.rs.GET;
9 import javax.ws.rs.POST;
10 import javax.ws.rs.Path;
11 import javax.ws.rs.Produces;
12 import javax.ws.rs.core.MediaType;
13
14 import br.com.csm.DAO.JPAUtil;
15 import br.com.csm.DAO.UsuarioDAO;
16 import br.com.csm.entidade.Usuario;
17
18 @Path("/usuarios")
19 public class UsuarioResources {
20
21
22     @GET
23     @Path("/listarUsuarios")
24     @Consumes(MediaType.APPLICATION_JSON)
25     @Produces(MediaType.APPLICATION_JSON)
26     public List<Usuario> listarUsuarios(){
27         List<Usuario> list = new ArrayList<Usuario>();
28         EntityManager em = JPAUtil.getEntityManager();
29         UsuarioDAO dao = new UsuarioDAO(em);
30         list=dao.listar();
31
32         return list;
33     }
34
35
36     @POST
37     @Path("/consultarUsuario")
38     @Consumes(MediaType.APPLICATION_JSON)
39     @Produces(MediaType.APPLICATION_JSON)
40     public Usuario buscarUsuario(long id){
41         Usuario usuario = new Usuario();
42         EntityManager em = JPAUtil.getEntityManager();
43         UsuarioDAO dao = new UsuarioDAO(em);
44         usuario = dao.consultar(id);
45
46         return usuario;
47     }
48
49
50 }

```

Essa notação serve para referenciar qual classe resource iremos trabalhos

Usando a notação GET

Essa notação para informamos qual métodos iremos acessar

```
2 import java.util.ArrayList;
3 import java.util.List;
4
5 import javax.persistence.EntityManager;
6 import javax.ws.rs.Consumes;
7 import javax.ws.rs.GET;
8 import javax.ws.rs.POST;
9 import javax.ws.rs.Path;
10 import javax.ws.rs.Produces;
11 import javax.ws.rs.core.MediaType;
12
13 import br.com.csm.DAO.JPAUtil;
14 import br.com.csm.DAO.UsuarioDAO;
15 import br.com.csm.entidade.Usuario;
16
17 @Path("/usuarios")
18 public class UsuarioResources {
19
20
21     @GET
22     @Path("/listarUsuarios")
23     @Consumes(MediaType.APPLICATION_JSON)
24     @Produces(MediaType.APPLICATION_JSON)
25     public List<Usuario> listarUsuarios(){
26         List<Usuario> list = new ArrayList<Usuario>();
27         EntityManager em = JPAUtil.getEntityManager();
28         UsuarioDAO dao = new UsuarioDAO(em);
29         list=dao.listar();
30
31         return list;
32     }
33
34 }
35
36     @POST
37     @Path("/consultarUsuario")
38     @Consumes(MediaType.APPLICATION_JSON)
39     @Produces(MediaType.APPLICATION_JSON)
40     public Usuario buscarUsuario(long id){
41         Usuario usuario = new Usuario();
42         EntityManager em = JPAUtil.getEntityManager();
43         UsuarioDAO dao = new UsuarioDAO(em);
44         usuario = dao.consultar(id);
45
46         return usuario;
47     }
48
49
50 }
```

Essa notação serve para referenciar qual classe resource iremos trabalhos

Usando a notação GET

Essa notação para informamos qual métodos iremos acessar

Esse notação é a mágica do Jersey

```
2 import java.util.ArrayList;
3 import java.util.List;
4
5 import javax.persistence.EntityManager;
6 import javax.ws.rs.Consumes;
7 import javax.ws.rs.GET;
8 import javax.ws.rs.POST;
9 import javax.ws.rs.Path;
10 import javax.ws.rs.Produces;
11 import javax.ws.rs.core.MediaType;
12
13 import br.com.csm.DAO.JPAUtil;
14 import br.com.csm.DAO.UsuarioDAO;
15 import br.com.csm.entidade.Usuario;
16
17 @Path("/usuarios")
18 public class UsuarioResources {
19
20
21     @GET
22     @Path("/listarUsuarios")
23     @Consumes(MediaType.APPLICATION_JSON)
24     @Produces(MediaType.APPLICATION_JSON)
25     public List<Usuario> listarUsuarios(){
26         List<Usuario> list = new ArrayList<Usuario>();
27         EntityManager em = JPAUtil.getEntityManager();
28         UsuarioDAO dao = new UsuarioDAO(em);
29         list=dao.listar();
30
31         return list;
32     }
33
34
35
36     @POST
37     @Path("/consultarUsuario")
38     @Consumes(MediaType.APPLICATION_JSON)
39     @Produces(MediaType.APPLICATION_JSON)
40     public Usuario buscarUsuario(long id){
41         Usuario usuario = new Usuario();
42         EntityManager em = JPAUtil.getEntityManager();
43         UsuarioDAO dao = new UsuarioDAO(em);
44         usuario = dao.consultar(id);
45
46         return usuario;
47     }
48
49
50 }
```

Essa notação serve para referenciar qual classe resource iremos trabalhos

Usando a notação GET

Essa notação para informamos qual métodos iremos acessar

Esse notação é a mágica do Jersey

```
2 import java.util.ArrayList;
3 import java.util.List;
4
5 import javax.persistence.EntityManager;
6 import javax.ws.rs.Consumes;
7 import javax.ws.rs.GET;
8 import javax.ws.rs.POST;
9 import javax.ws.rs.Path;
10 import javax.ws.rs.Produces;
11 import javax.ws.rs.core.MediaType;
12
13 import br.com.csm.DAO.JPAUtil;
14 import br.com.csm.DAO.UsuarioDAO;
15 import br.com.csm.entidade.Usuario;
16
17 @Path("/usuarios")
18 public class UsuarioResources {
19
20
21     @GET
22     @Path("/listarUsuarios")
23     @Consumes(MediaType.APPLICATION_JSON)
24     @Produces(MediaType.APPLICATION_JSON)
25     public List<Usuario> listarUsuarios(){
26         List<Usuario> list = new ArrayList<Usuario>();
27         EntityManager em = JPAUtil.getEntityManager();
28         UsuarioDAO dao = new UsuarioDAO(em);
29         list=dao.listar();
30
31         return list;
32     }
33
34 }
35
36     @POST
37     @Path("/consultarUsuario")
38     @Consumes(MediaType.APPLICATION_JSON)
39     @Produces(MediaType.APPLICATION_JSON)
40     public Usuario buscarUsuario(long id){
41         Usuario usuario = new Usuario();
42         EntityManager em = JPAUtil.getEntityManager();
43         UsuarioDAO dao = new UsuarioDAO(em);
44         usuario = dao.consultar(id);
45
46         return usuario;
47     }
48
49
50 }
```

Essa notação serve para referenciar qual classe resource iremos trabalhos

Usando a notação GET

Ele irá transformar o retorno desse método em conteúdo JSON

Essa notação para informamos qual métodos iremos acessar

Esse notação é a mágica do Jersey

Não precisa se preocupar com nada de retorno de Conteúdo, pois isso é função JERSEY

```
2 import java.util.ArrayList;
3 import java.util.List;
4
5 import javax.persistence.EntityManager;
6 import javax.ws.rs.Consumes;
7 import javax.ws.rs.GET;
8 import javax.ws.rs.POST;
9 import javax.ws.rs.Path;
10 import javax.ws.rs.Produces;
11 import javax.ws.rs.core.MediaType;
12
13 import br.com.csm.DAO.JPAUtil;
14 import br.com.csm.DAO.UsuarioDAO;
15 import br.com.csm.entidade.Usuario;
16
17 @Path("/usuarios")
18 public class UsuarioResources {
19
20
21     @GET
22     @Path("/listarUsuarios")
23     @Consumes(MediaType.APPLICATION_JSON)
24     @Produces(MediaType.APPLICATION_JSON)
25     public List<Usuario> listarUsuarios(){
26         List<Usuario> list = new ArrayList<Usuario>();
27         EntityManager em = JPAUtil.getEntityManager();
28         UsuarioDAO dao = new UsuarioDAO(em);
29         list=dao.listar();
30
31         return list;
32     }
33
34
35
36     @POST
37     @Path("/consultarUsuario")
38     @Consumes(MediaType.APPLICATION_JSON)
39     @Produces(MediaType.APPLICATION_JSON)
40     public Usuario buscarUsuario(long id){
41         Usuario usuario = new Usuario();
42         EntityManager em = JPAUtil.getEntityManager();
43         UsuarioDAO dao = new UsuarioDAO(em);
44         usuario = dao.consultar(id);
45
46         return usuario;
47     }
48
49
50 }
```

Essa notação serve para referenciar qual classe resource iremos trabalhos

Usando a notação GET

Ele irá transformar o retorno desse método em conteúdo JSON

## Atenção nos imports !!!

Essa notação para informamos qual métodos iremos acessar

Esse notação é a mágica do Jersey

Não precisa se preocupar com nada de retorno de Conteúdo, pois isso é função JERSEY

```
2 import java.util.ArrayList;
3 import java.util.List;
4
5 import javax.persistence.EntityManager;
6 import javax.ws.rs.Consumes;
7 import javax.ws.rs.GET;
8 import javax.ws.rs.POST;
9 import javax.ws.rs.Path;
10 import javax.ws.rs.Produces;
11 import javax.ws.rs.core.MediaType;
12
13 import br.com.csm.DAO.JPAUtil;
14 import br.com.csm.DAO.UsuarioDAO;
15 import br.com.csm.entidade.Usuario;
16
17 @Path("/usuarios")
18 public class UsuarioResources {
19
20
21     @GET
22     @Path("/listarUsuarios")
23     @Consumes(MediaType.APPLICATION_JSON)
24     @Produces(MediaType.APPLICATION_JSON)
25     public List<Usuario> listarUsuarios(){
26         List<Usuario> list = new ArrayList<Usuario>();
27         EntityManager em = JPAUtil.getEntityManager();
28         UsuarioDAO dao = new UsuarioDAO(em);
29         list=dao.listar();
30
31         return list;
32     }
33
34
35     @POST
36     @Path("/consultarUsuario")
37     @Consumes(MediaType.APPLICATION_JSON)
38     @Produces(MediaType.APPLICATION_JSON)
39     public Usuario buscarUsuario(long id){
40         Usuario usuario = new Usuario();
41         EntityManager em = JPAUtil.getEntityManager();
42         UsuarioDAO dao = new UsuarioDAO(em);
43         usuario = dao.consultar(id);
44
45         return usuario;
46     }
47
48
49
50 }
```

Essa notação serve para referenciar qual classe resource iremos trabalhos

Usando a notação GET

Ele irá transformar o retorno desse método em conteúdo JSON

## Atenção nos imports !!!

Essa notação para informamos qual métodos iremos acessar

Esse notação é a mágica do Jersey

Não precisa se preocupar com nada de retorno de Conteúdo, pois isso é função JERSEY

```
2 import java.util.ArrayList;
3 import java.util.List;
4
5 import javax.persistence.EntityManager;
6 import javax.ws.rs.Consumes;
7 import javax.ws.rs.GET;
8 import javax.ws.rs.POST;
9 import javax.ws.rs.Path;
10 import javax.ws.rs.Produces;
11 import javax.ws.rs.core.MediaType;
12
13 import br.com.csm.DAO.JPAUtil;
14 import br.com.csm.DAO.UsuarioDAO;
15 import br.com.csm.entidade.Usuario;
16
17 @Path("/usuarios")
18 public class UsuarioResources {
19
20
21     @GET
22     @Path("/listarUsuarios")
23     @Consumes(MediaType.APPLICATION_JSON)
24     @Produces(MediaType.APPLICATION_JSON)
25     public List<Usuario> listarUsuarios(){
26         List<Usuario> list = new ArrayList<Usuario>();
27         EntityManager em = JPAUtil.getEntityManager();
28         UsuarioDAO dao = new UsuarioDAO(em);
29         list=dao.listar();
30
31         return list;
32     }
33
34
35     @POST
36     @Path("/consultarUsuario")
37     @Consumes(MediaType.APPLICATION_JSON)
38     @Produces(MediaType.APPLICATION_JSON)
39     public Usuario buscarUsuario(long id){
40         Usuario usuario = new Usuario();
41         EntityManager em = JPAUtil.getEntityManager();
42         UsuarioDAO dao = new UsuarioDAO(em);
43         usuario = dao.consultar(id);
44
45         return usuario;
46     }
47
48
49
50 }
```

Essa notação serve para referenciar qual classe resource iremos trabalhos

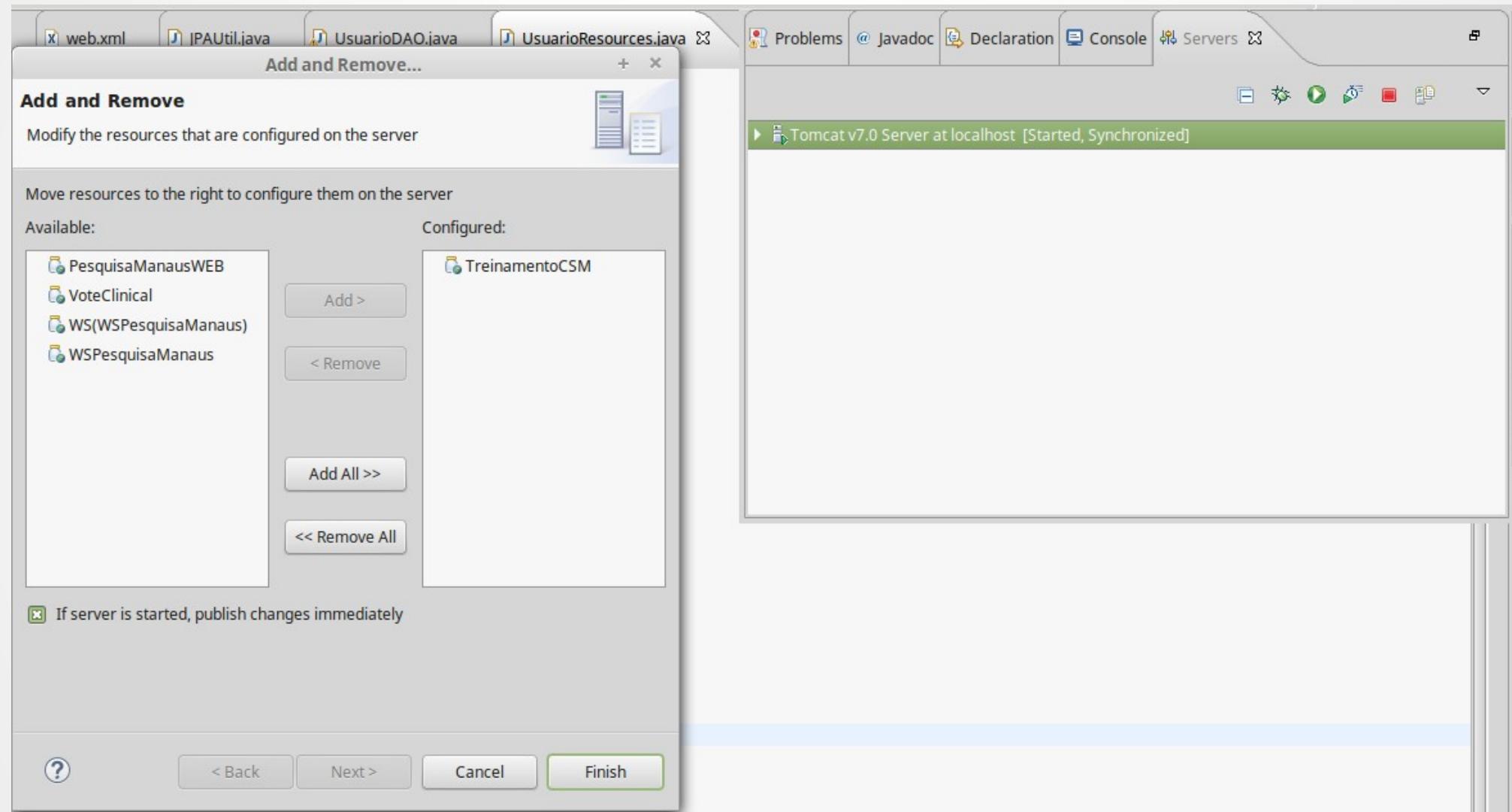
Usando a notação GET

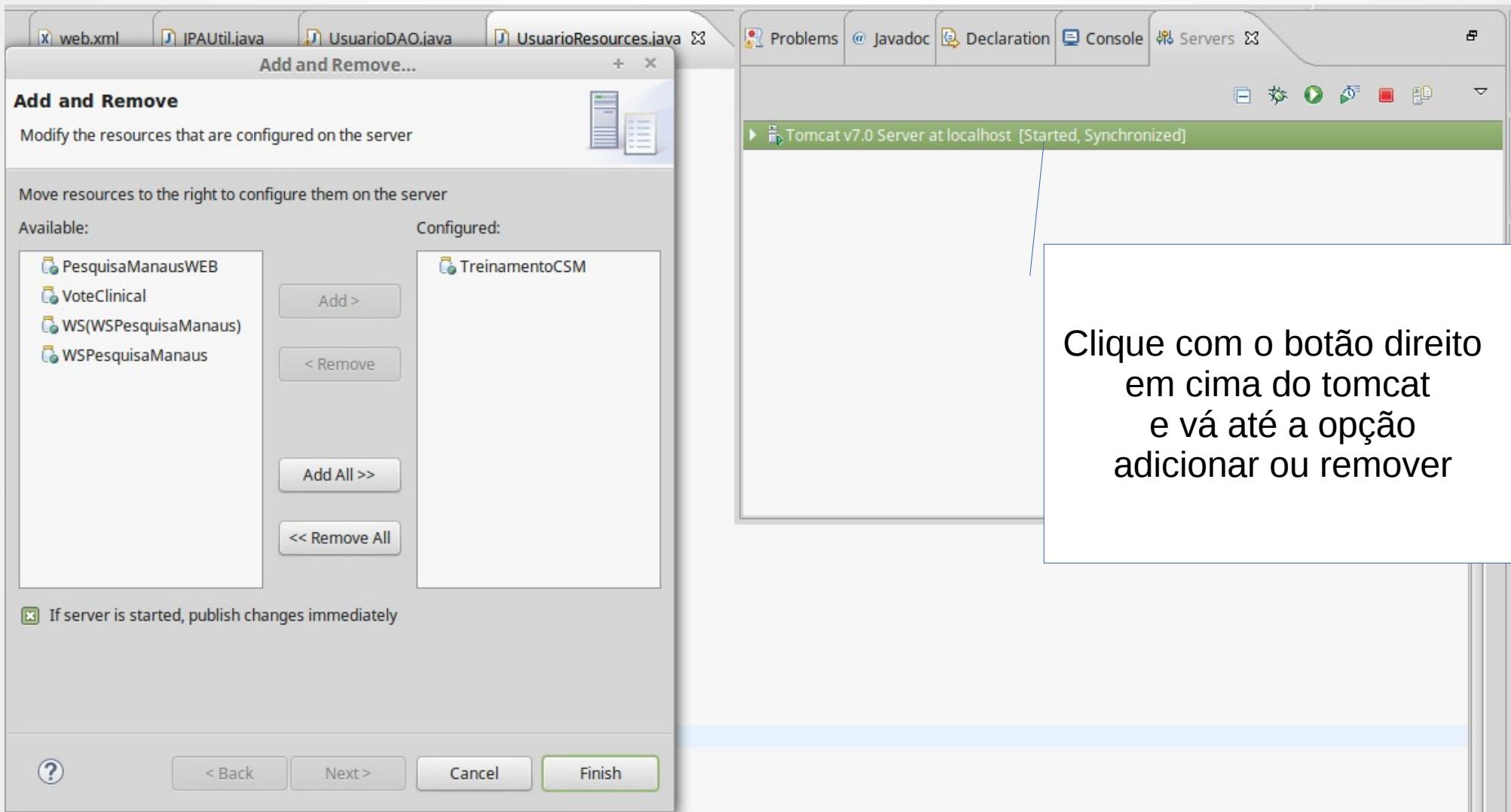
Ele irá transformar o retorno desse método em conteúdo JSON

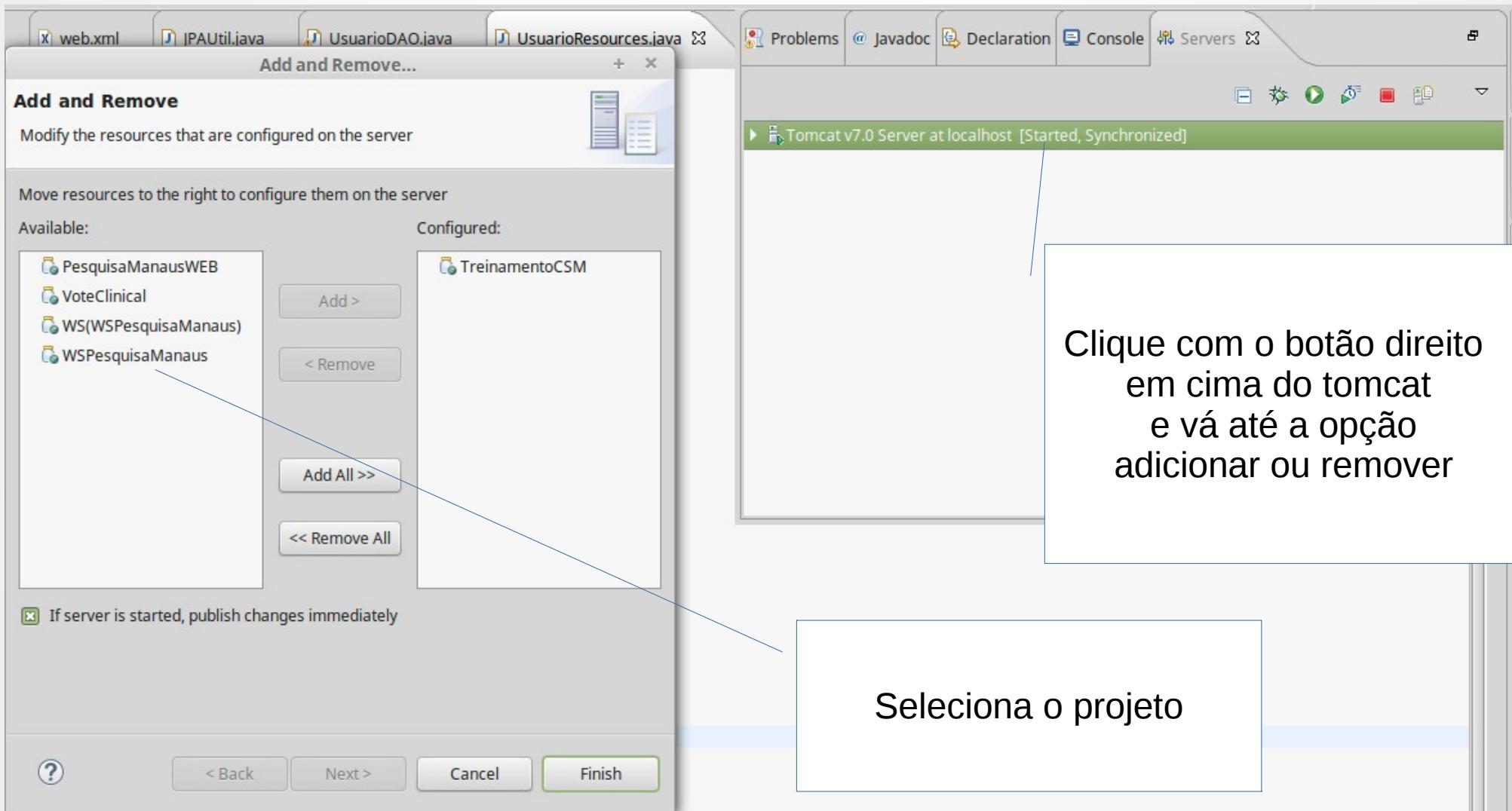
No nosso exemplo só usaremos o GET, mas fica de brinde como fazer usando POST

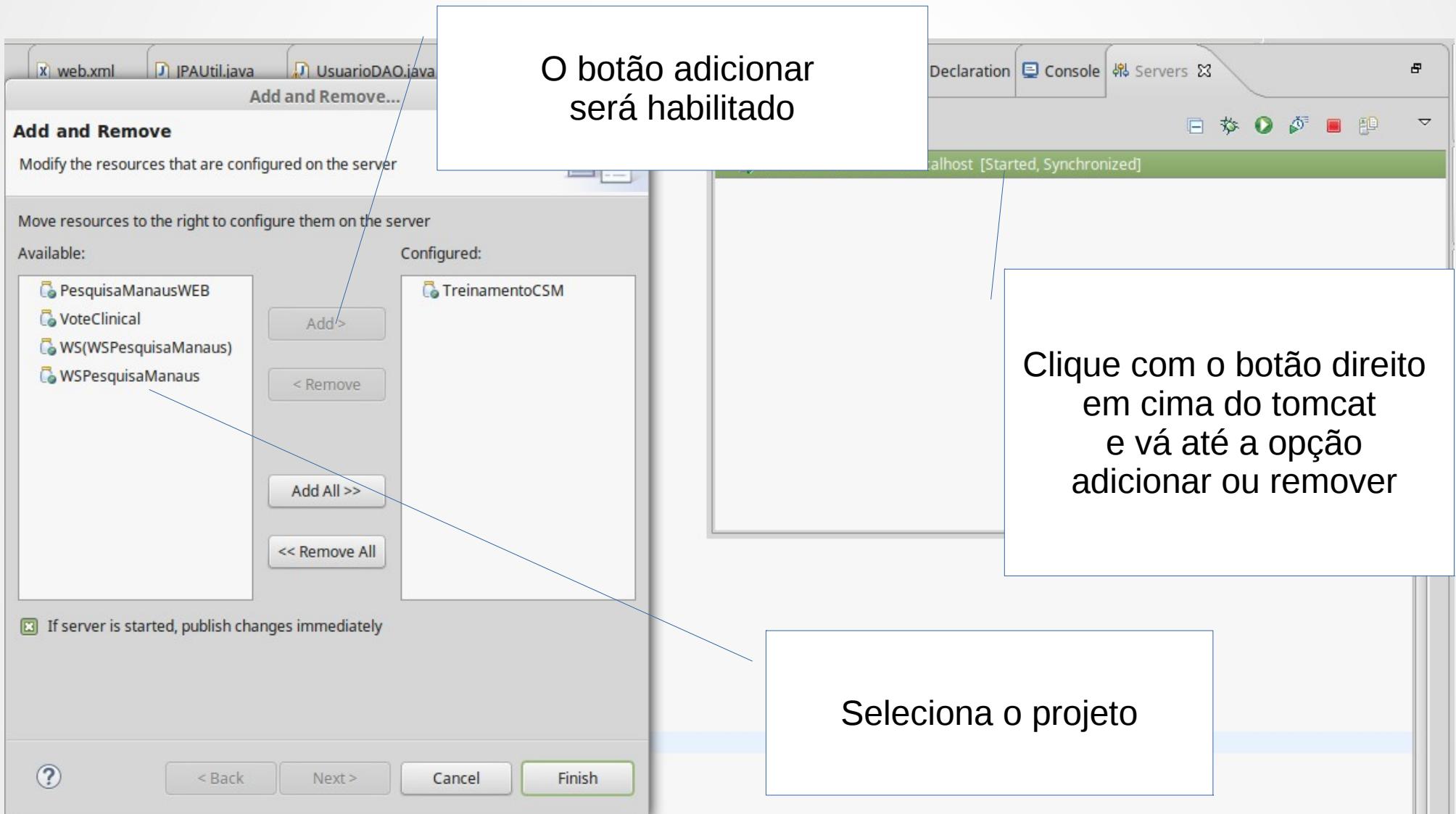
# Agora vamos testar...

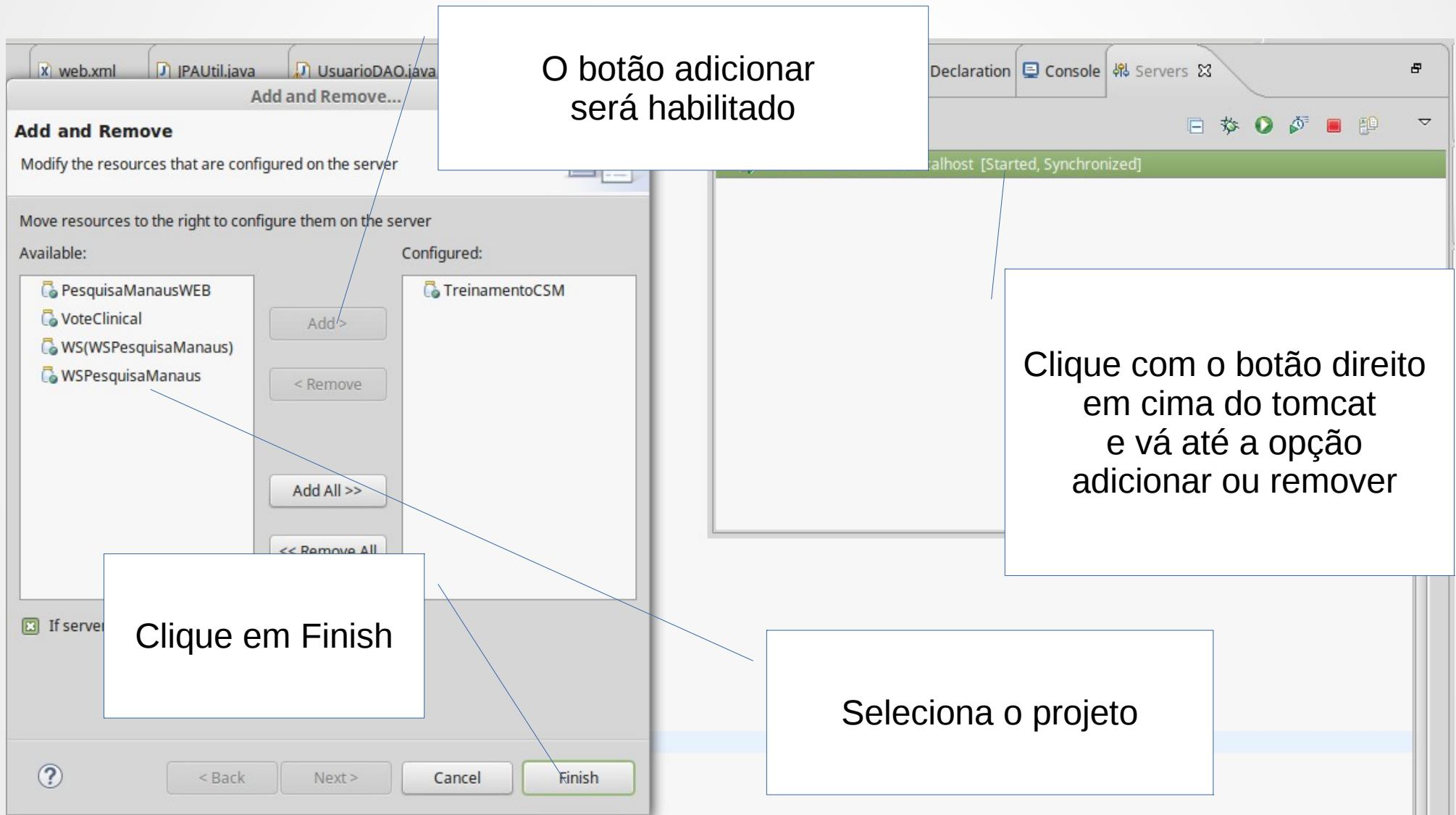
- Primeiro start seu mysql.
- Crie seu Banco de dados
- Após isso...

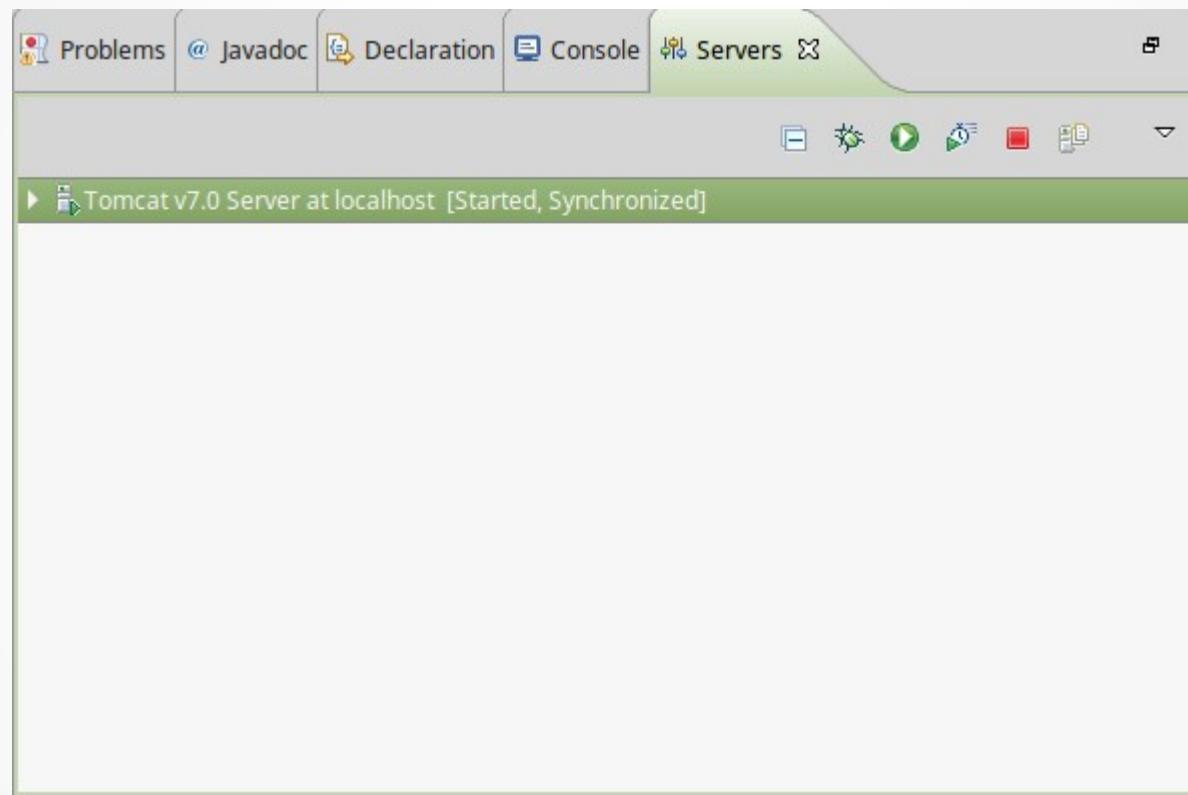


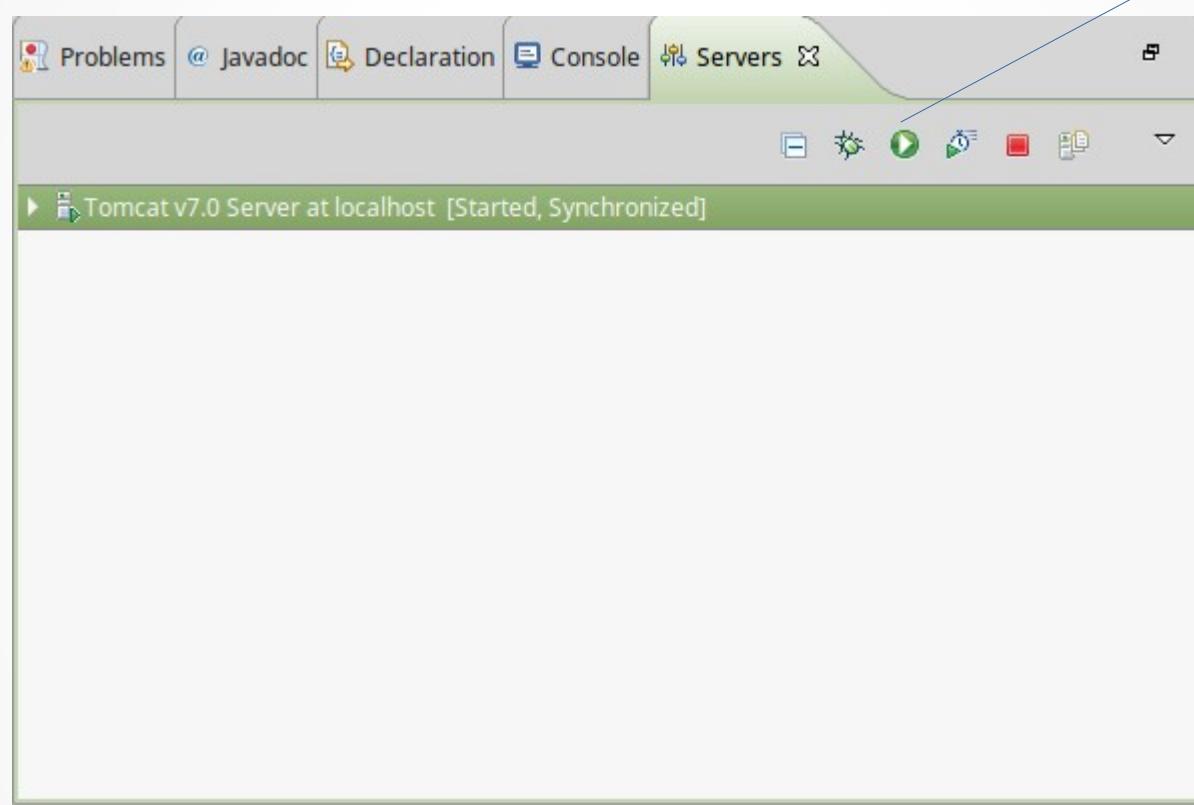




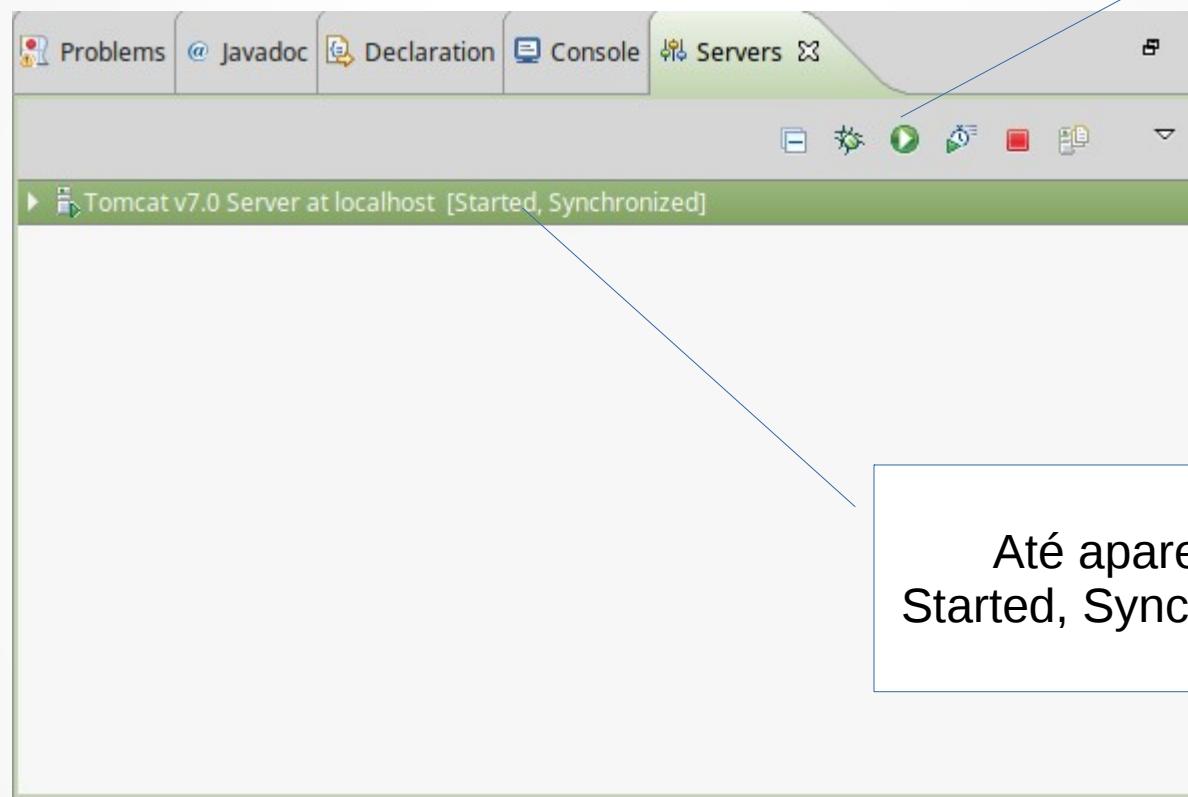








# Clique no play



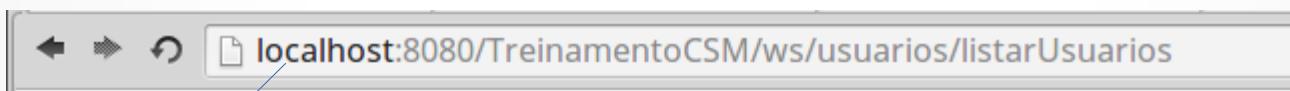
Até aparecer  
Started, Synchronized

Clique no play

# No browser do seu pc



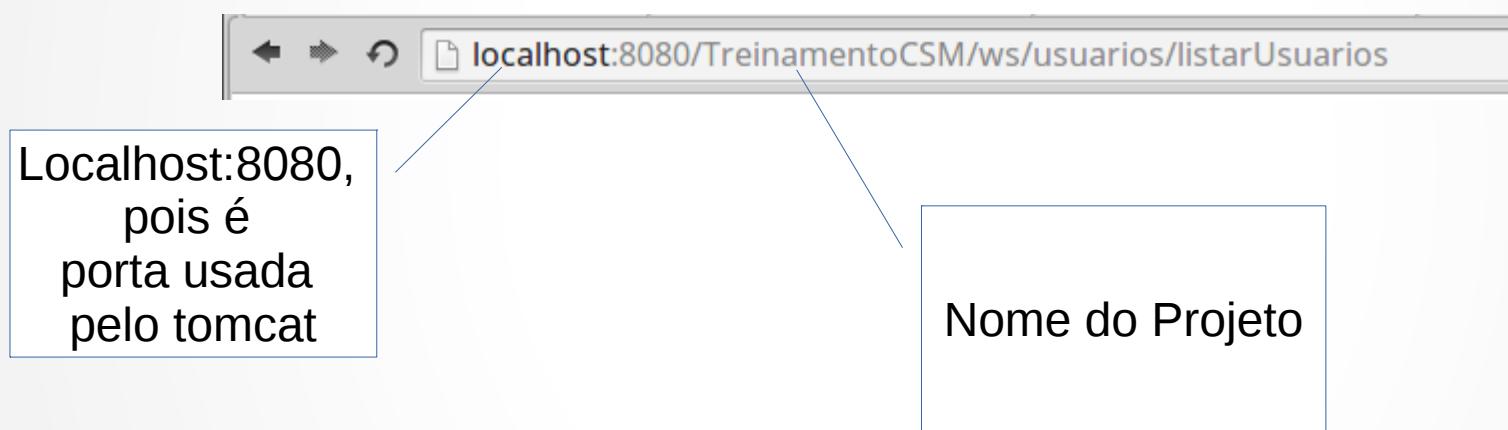
# No browser do seu pc



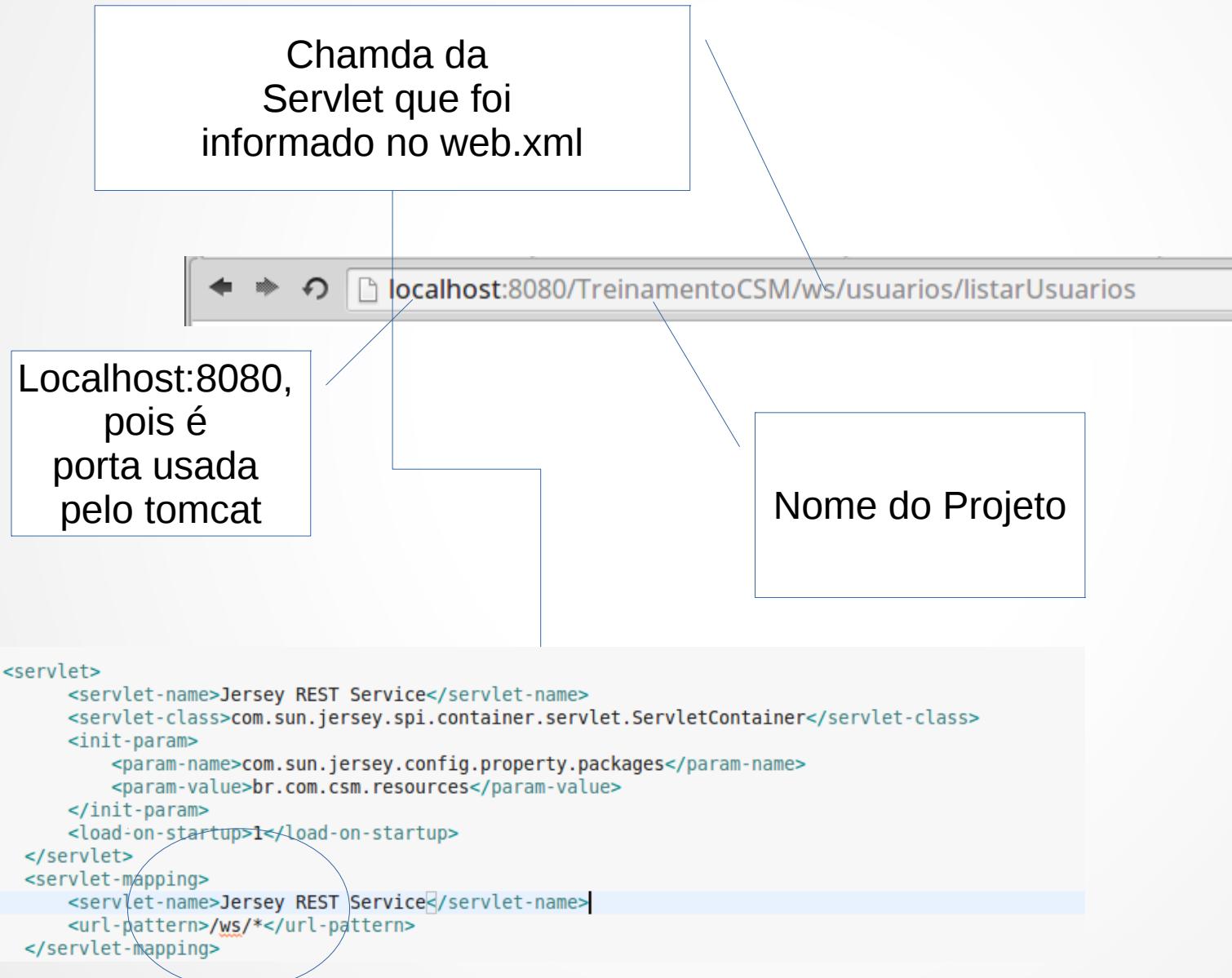
Localhost:8080,  
pois é  
porta usada  
pelo tomcat



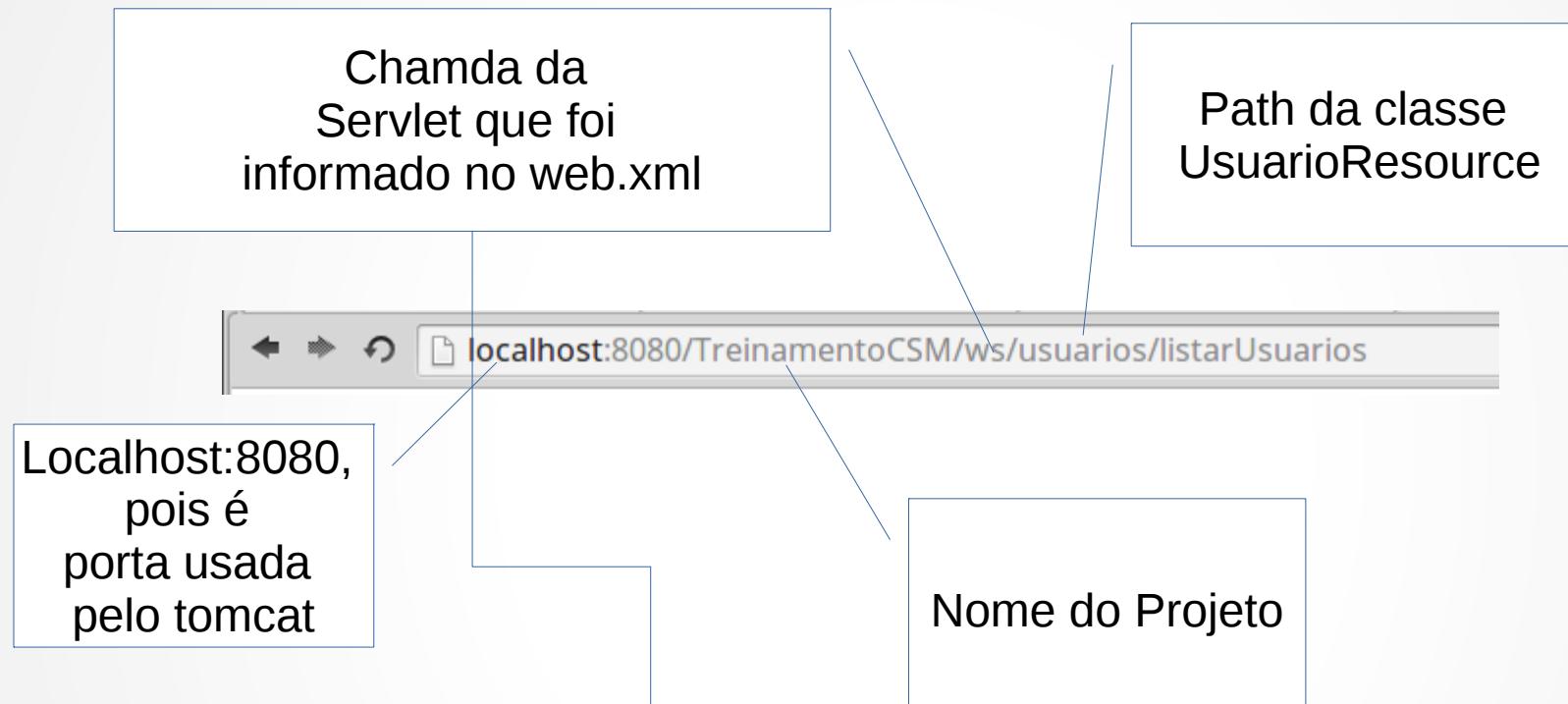
# No browser do seu pc



# No browser do seu pc

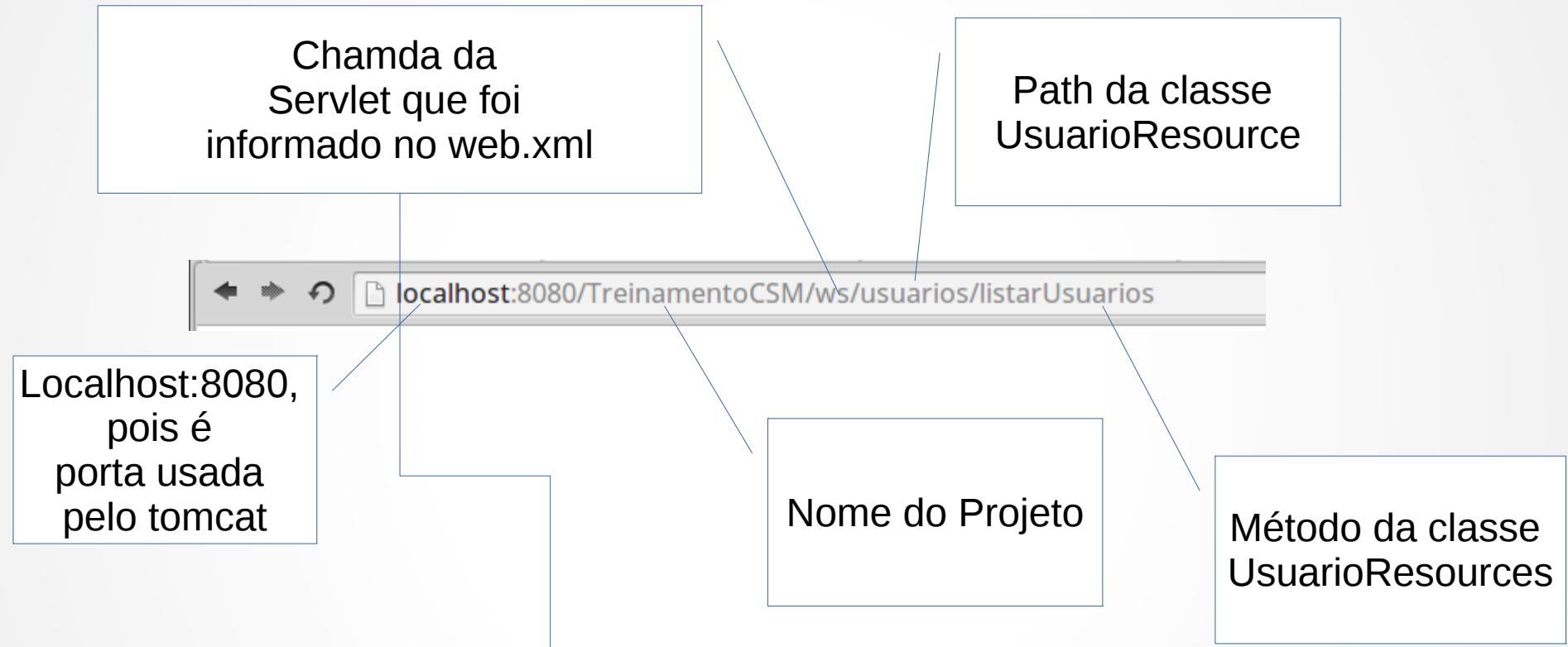


# No browser do seu pc



```
<servlet>
    <servlet-name>Jersey REST Service</servlet-name>
    <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
    <init-param>
        <param-name>com.sun.jersey.config.property.packages</param-name>
        <param-value>br.com.csm.resources</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>Jersey REST Service</servlet-name>
    <url-pattern>/ws/*</url-pattern>
</servlet-mapping>
```

# No browser do seu pc



```
<servlet>
    <servlet-name>Jersey REST Service</servlet-name>
    <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
    <init-param>
        <param-name>com.sun.jersey.config.property.packages</param-name>
        <param-value>br.com.csm.resources</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>Jersey REST Service</servlet-name>
    <url-pattern>/ws/*</url-pattern>
</servlet-mapping>
```

# Funcionou!!!!

```
{"usuario":[{"email":"alice.adativa@gmail.com","id":"2","nome":"Alice Adativa","telefone":"981971753"}, {"email":"bruno.abia@gmail.com","id":"1","nome":"Bruno Ábia Souza","telefone":"981591643"}]}
```



# O que pode ser melhorado?

- Criar uma interface com usuário para possa ser adicionado pela WEB dados no banco.
- Uso de classes Genéricas
- E etc...

# Obrigado!

- <https://github.com/brunoabia/wscsm> - todo material usado nessa apresentação
- E-mail para contato: [bruno.abia@gmail.com](mailto:bruno.abia@gmail.com)