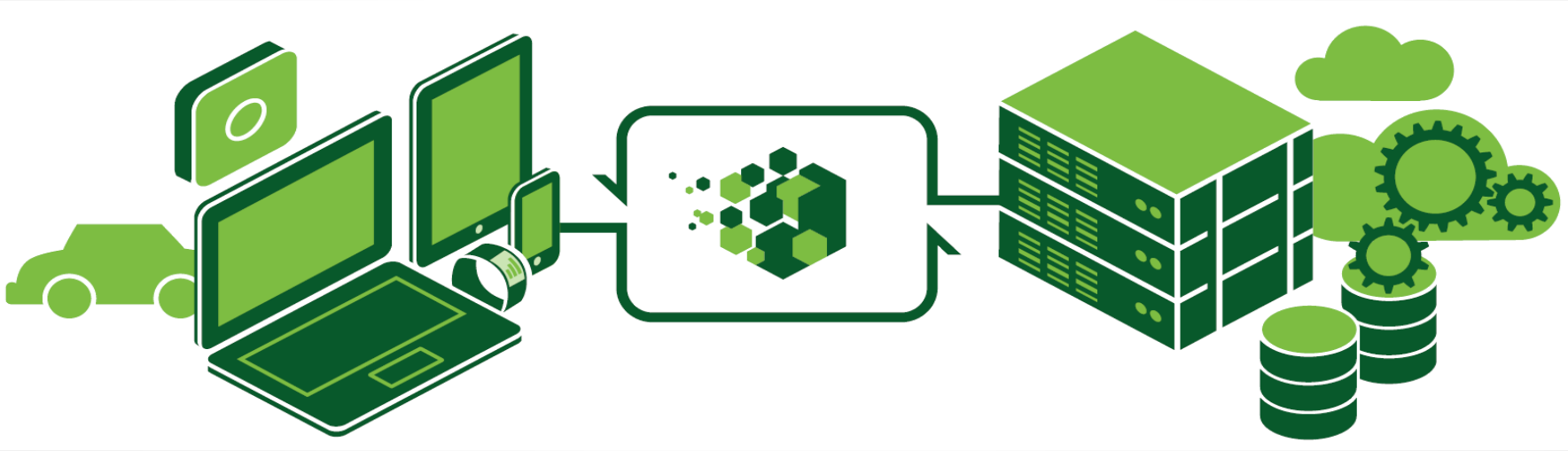
Tutorial: Criação de um Web Service usando Java 8 + Jersey + hibernate 3 e JPA



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

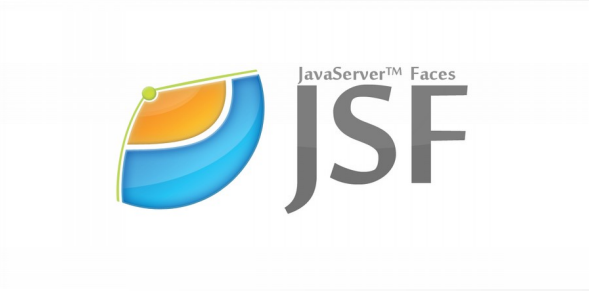
Ementa

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

Ambiente de Desenvolvimento

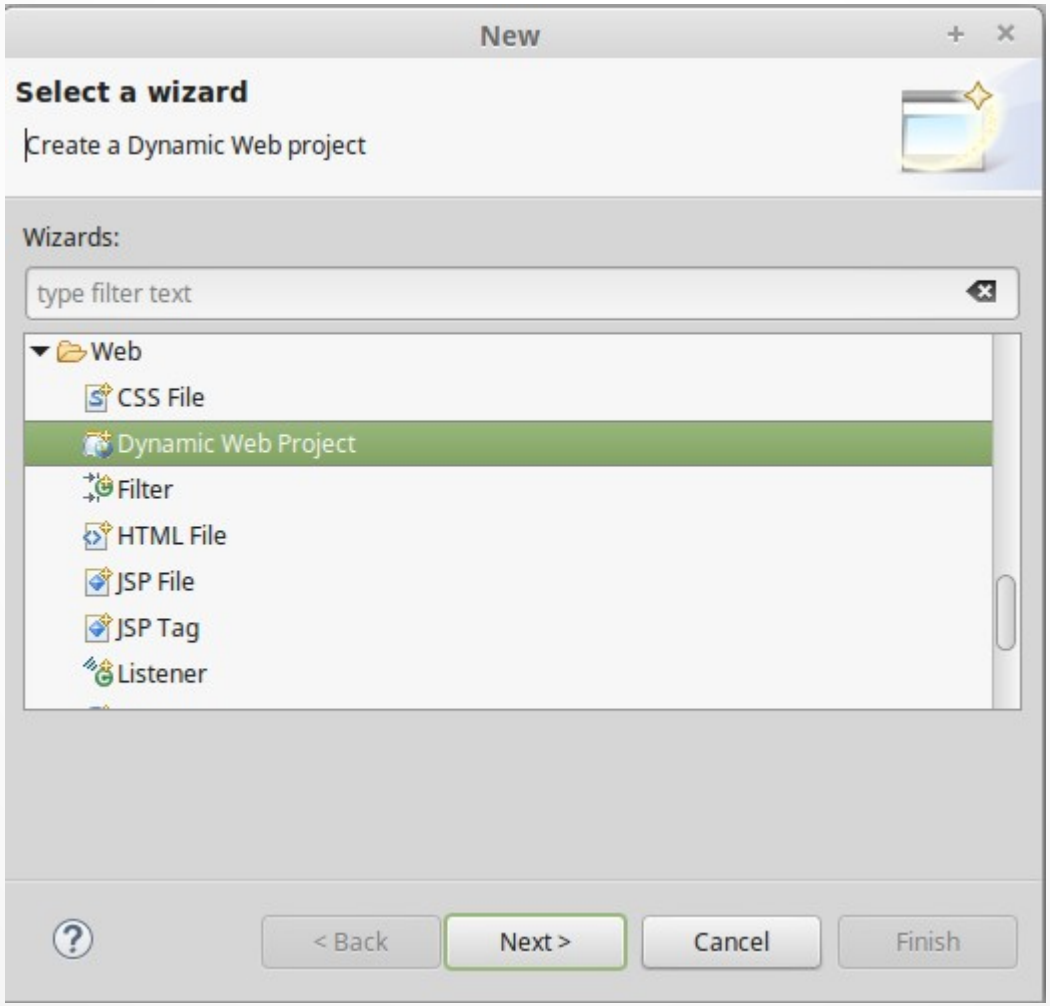
- https://eclipse.org/downloads/ - Eclipse
- http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html?ssSourceSiteId=otnpt - JDK 8
- https://tomcat.apache.org/download-80.cgi - tomcat 8
- https://www.apachefriends.org/pt_br/index.html – xammp

Apresentação das APIs

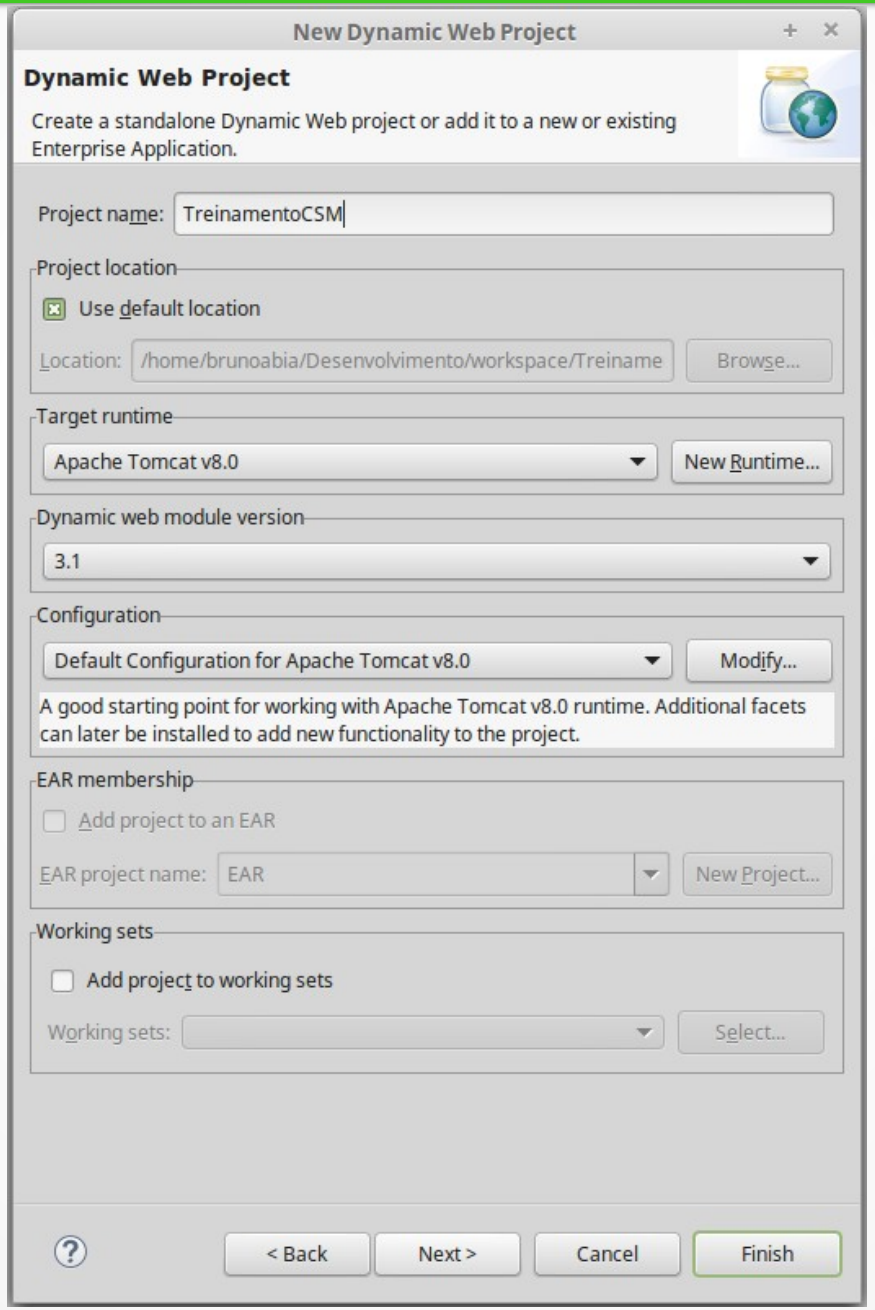


Criação do projeto

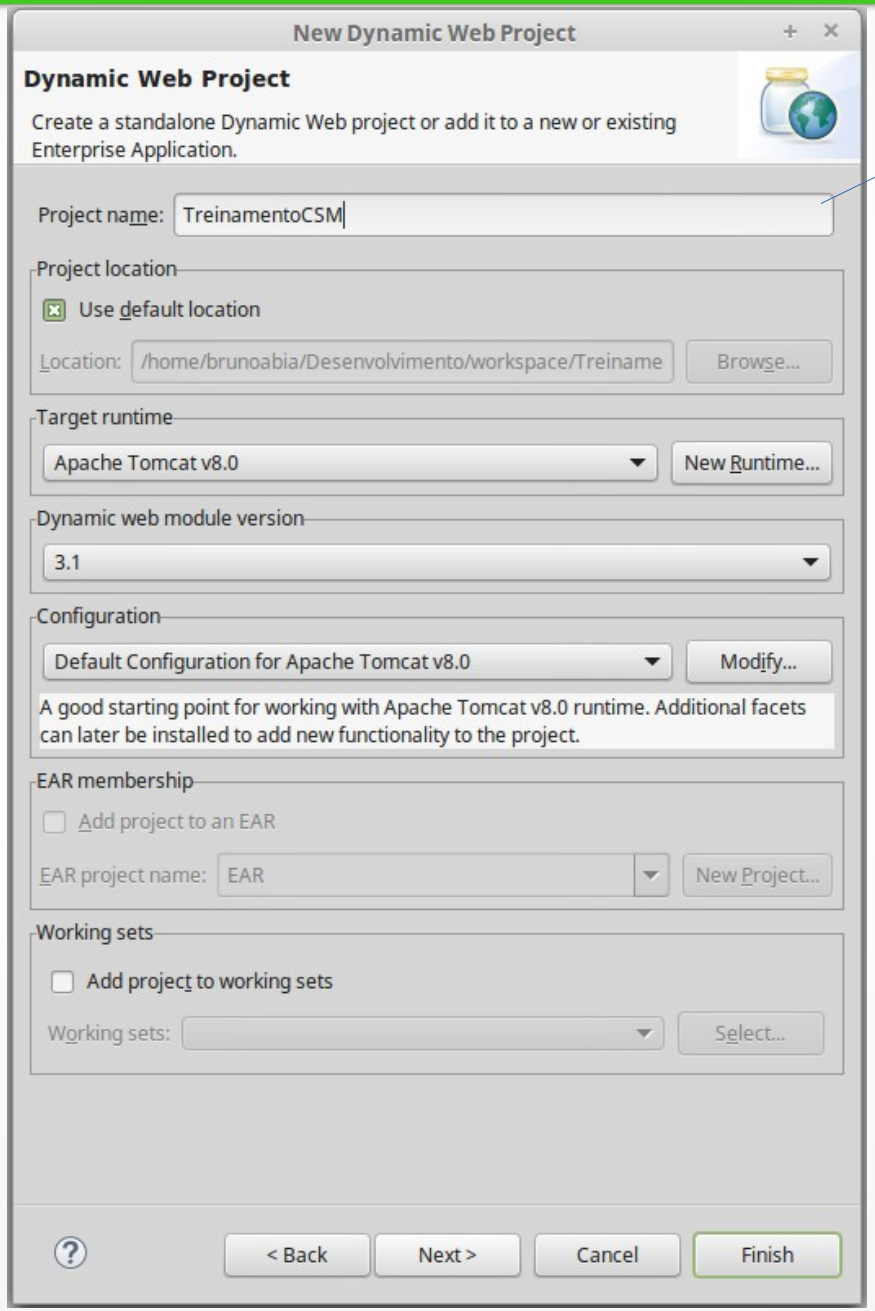
- Create a Dynamic Web project



Criação do Projeto



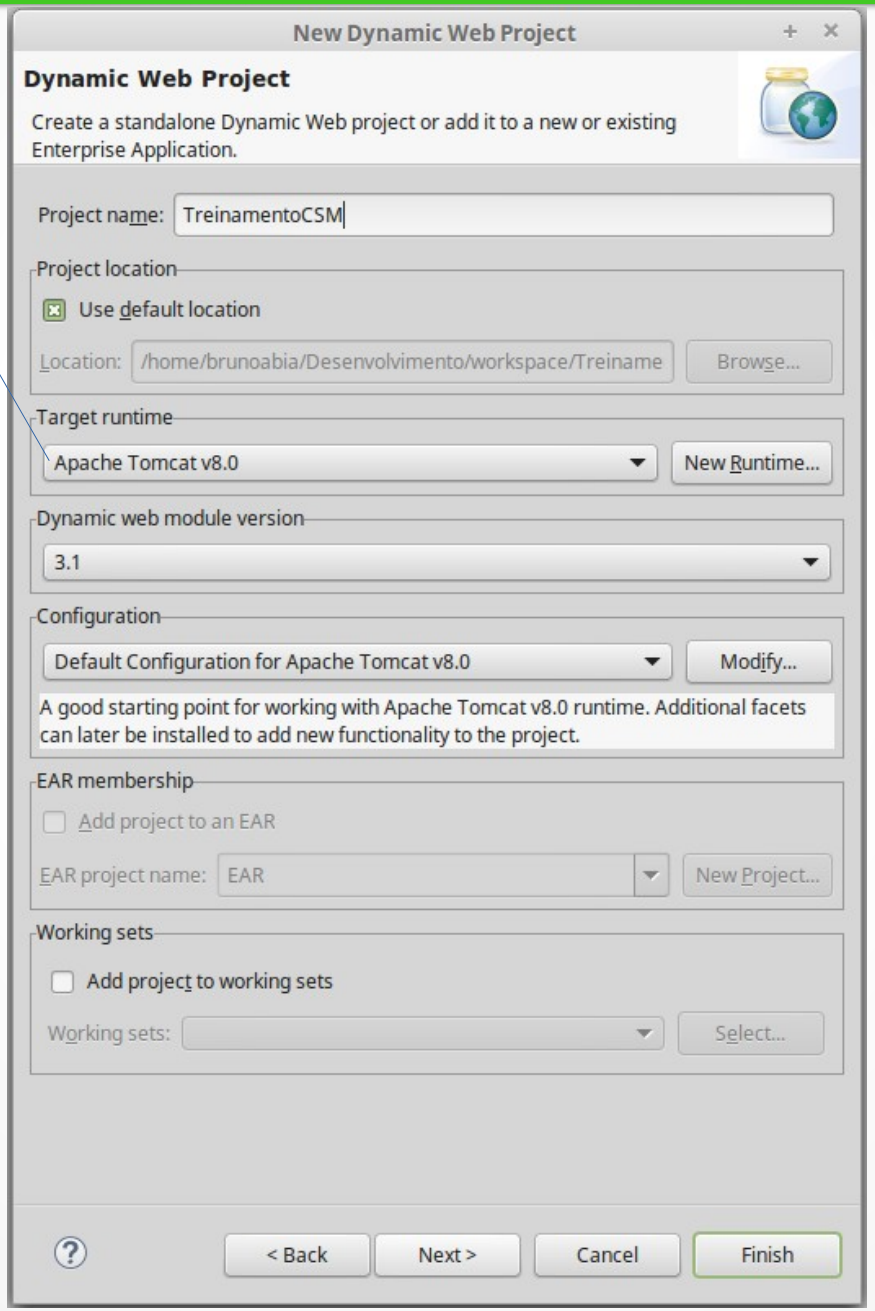
Criação do Projeto



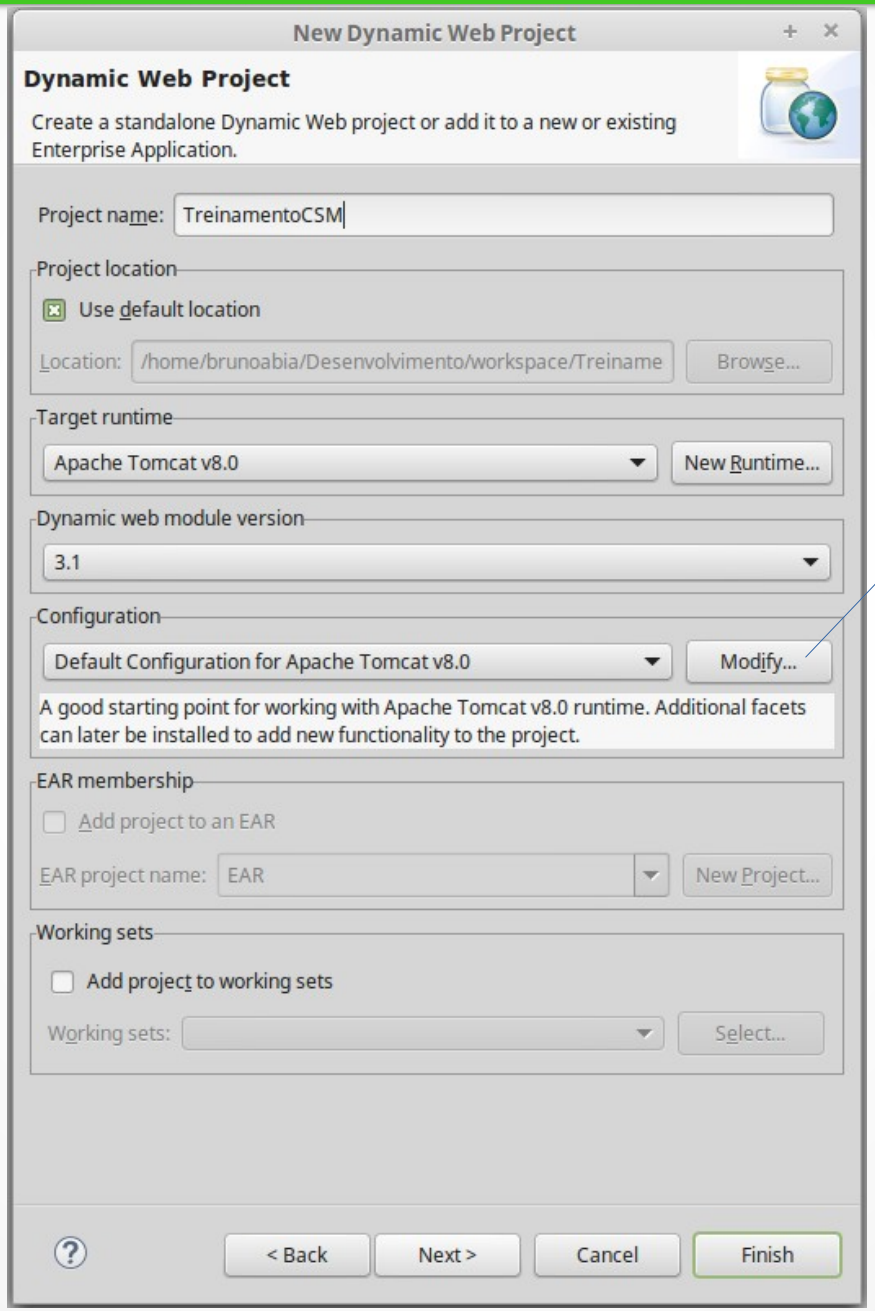
Informe o nome do projeto

Criação do Projeto

O target marque o tomcat

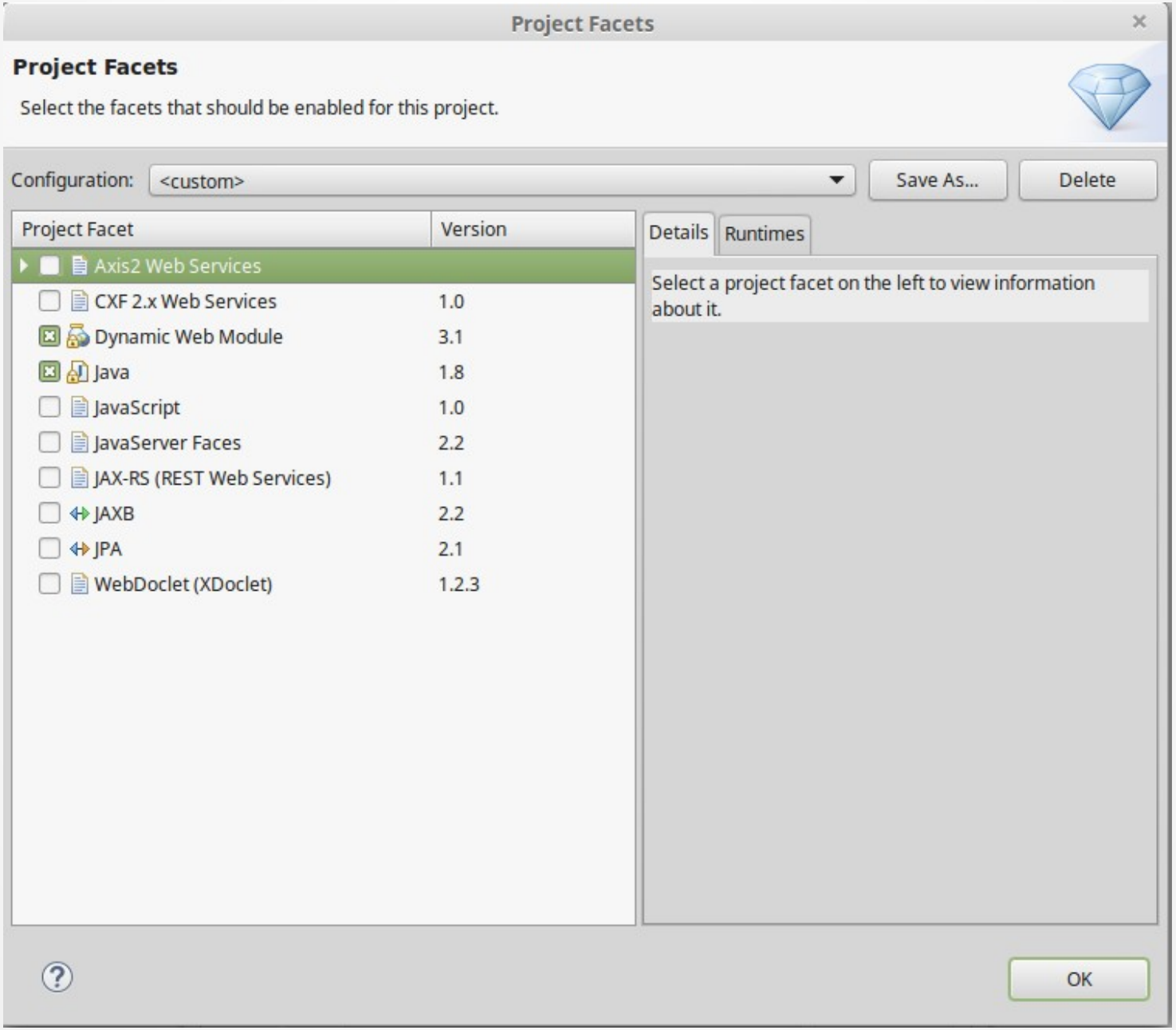


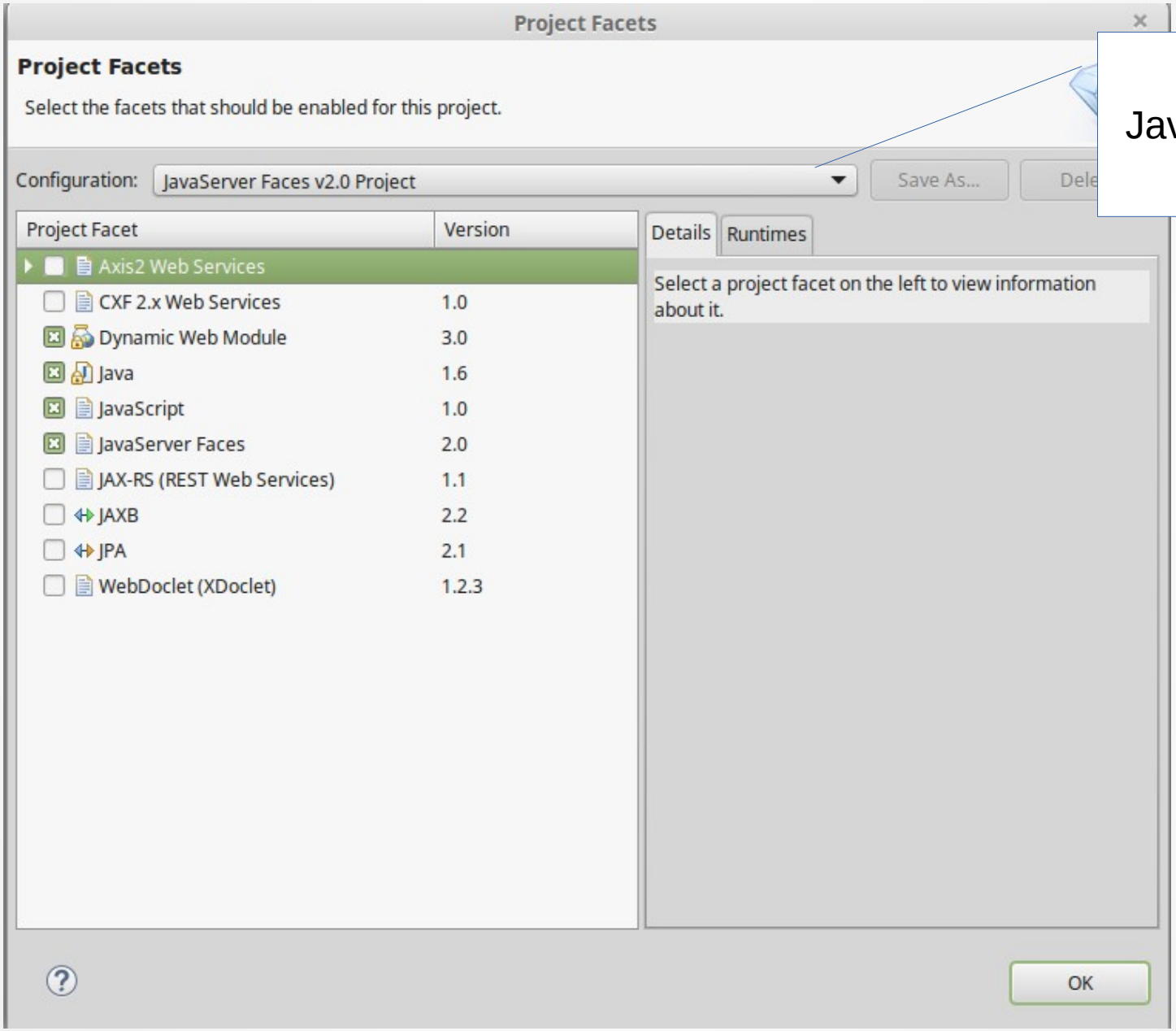
Criação do Projeto



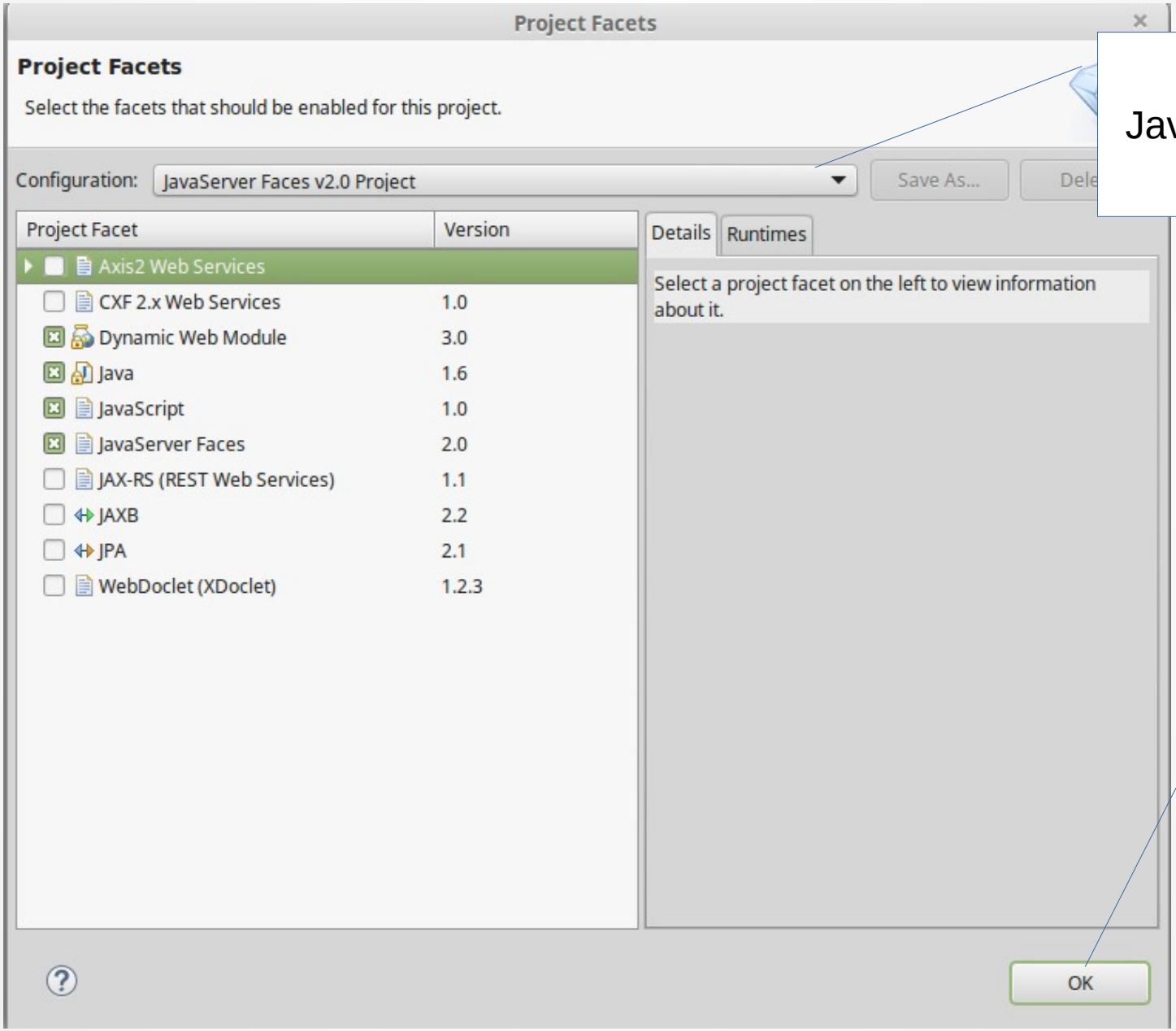
Clique em modify

Criação do Projeto





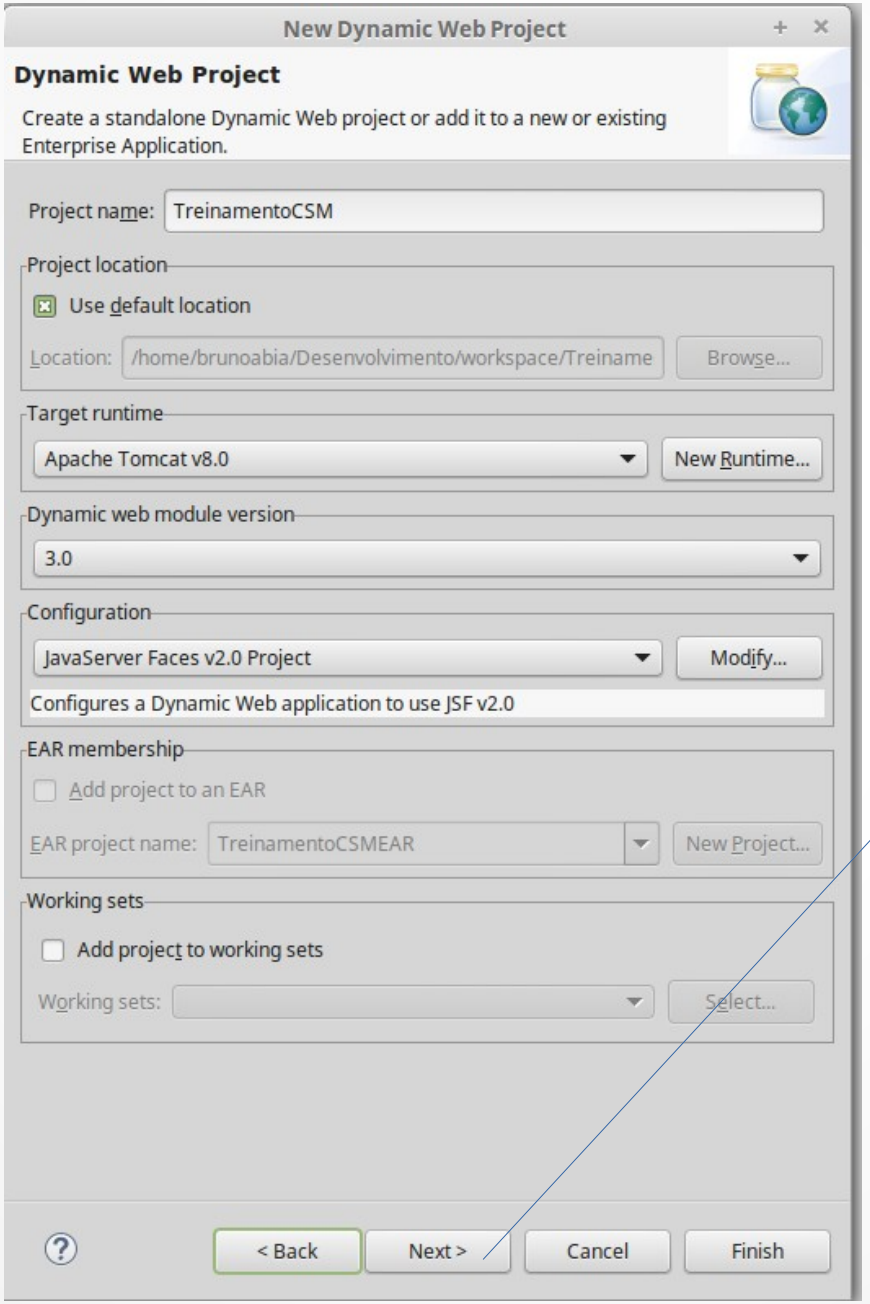
Selecione JavaServer Faces v2.0



Selecione JavaServer Faces v2.0

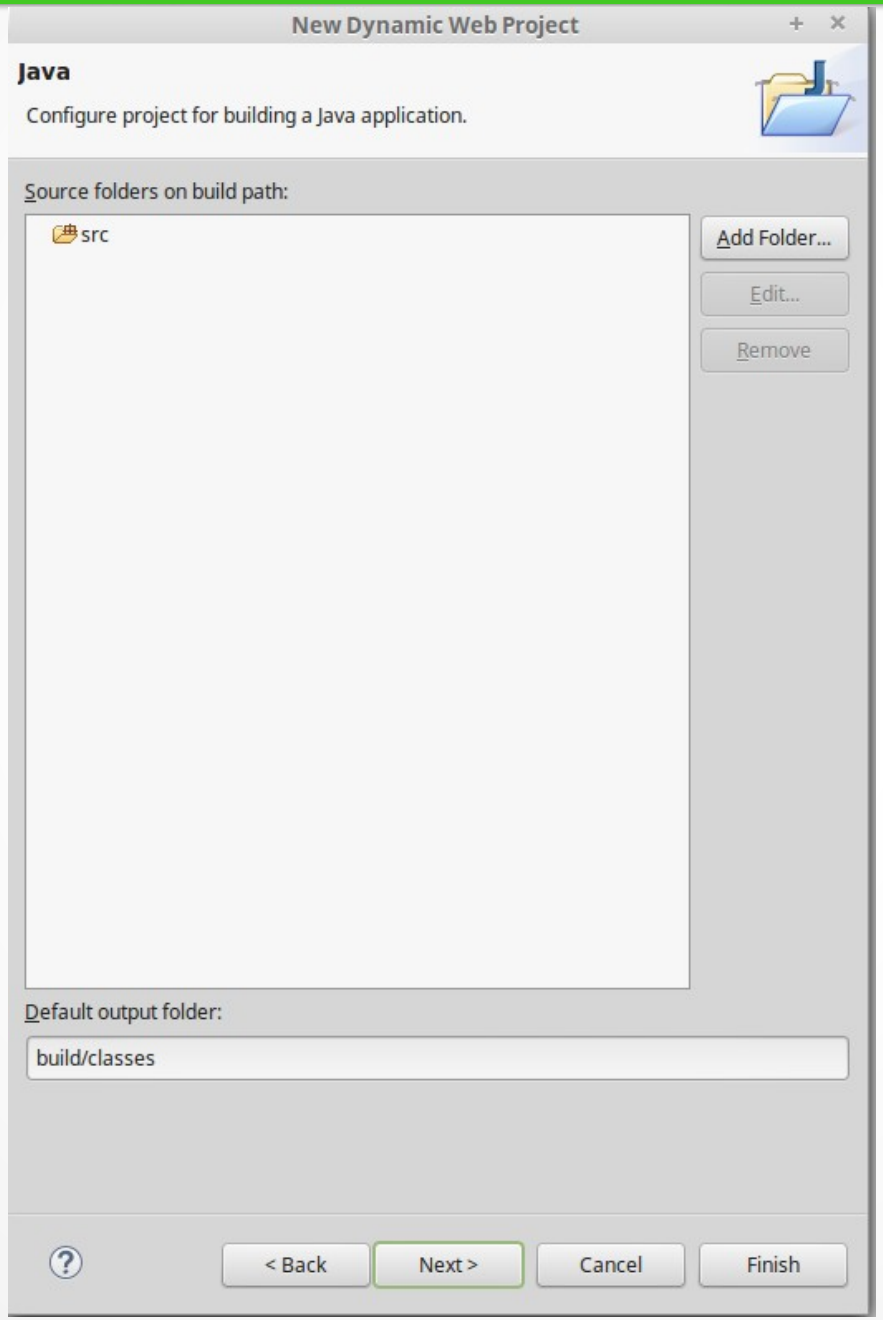
Clique em OK

Criação do Projeto

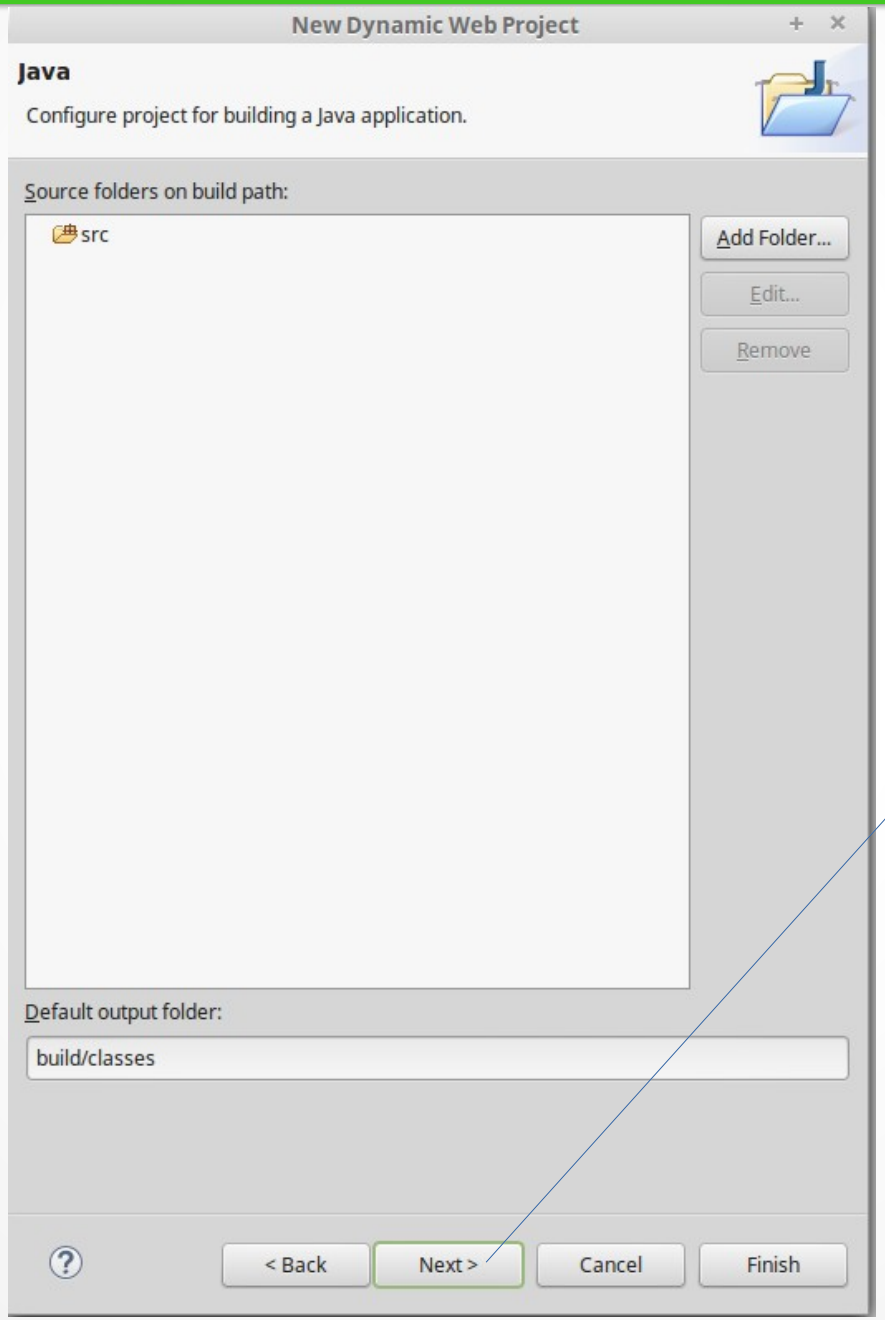


Clique em NEXT

Criação do Projeto

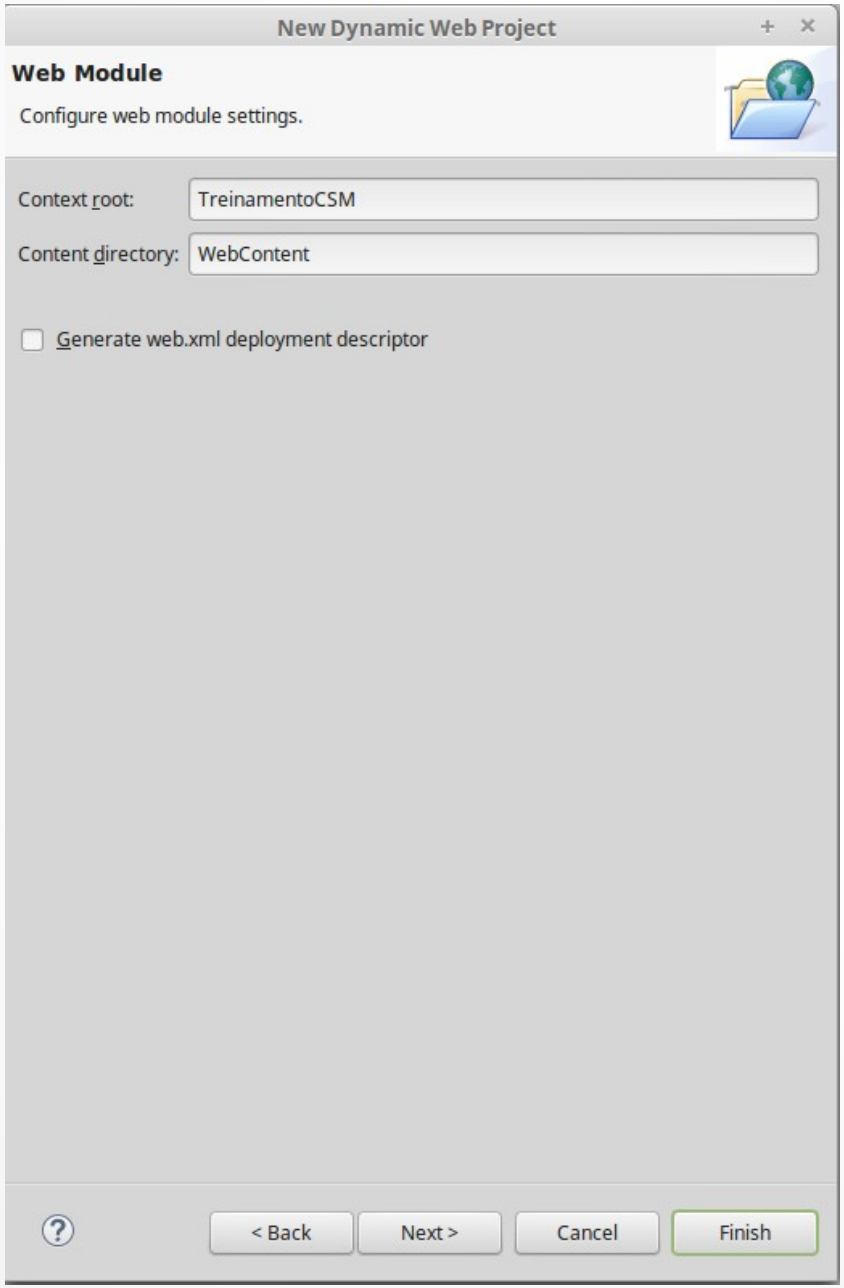


Criação do Projeto



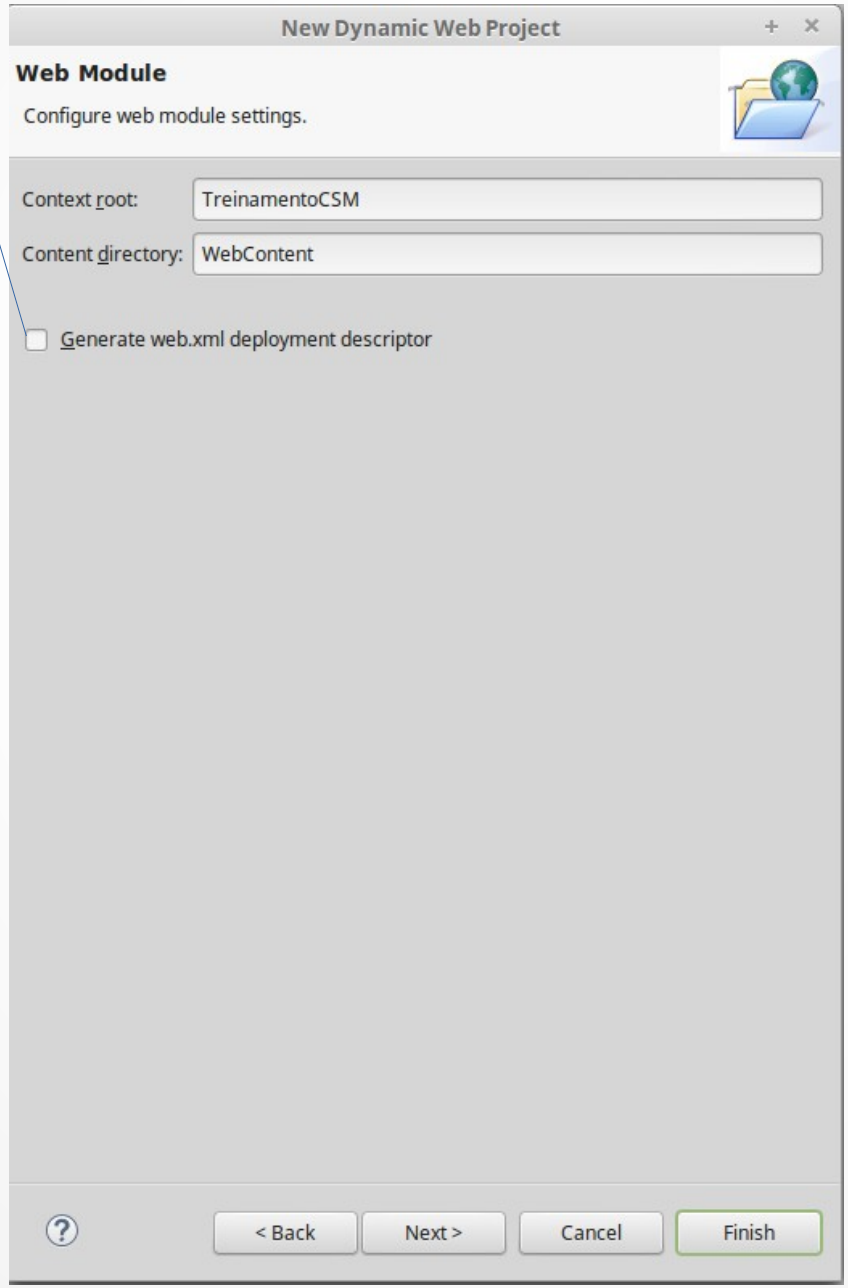
Clique em Next

Criação do Projeto



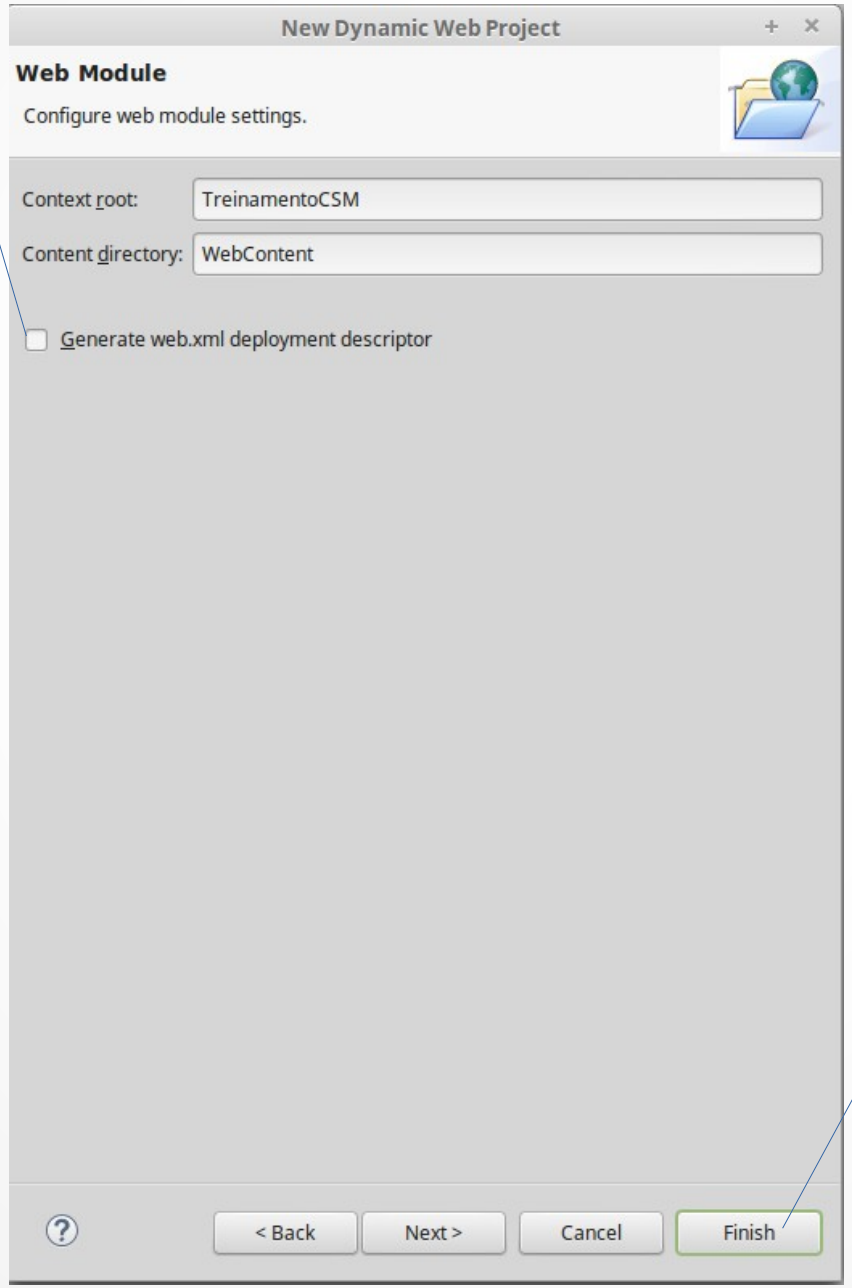
Criação do Projeto

Marque a opção para gerar web.xml



Criação do Projeto

Marque a opção para gerar web.xml

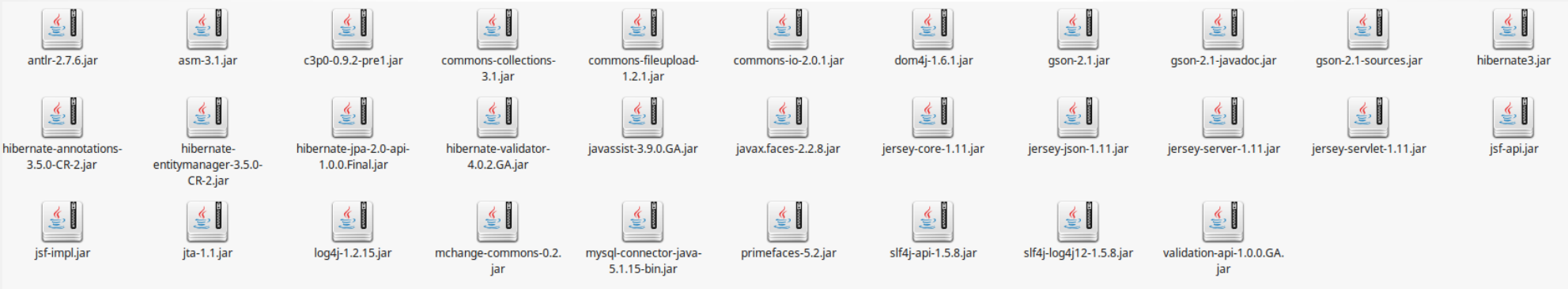


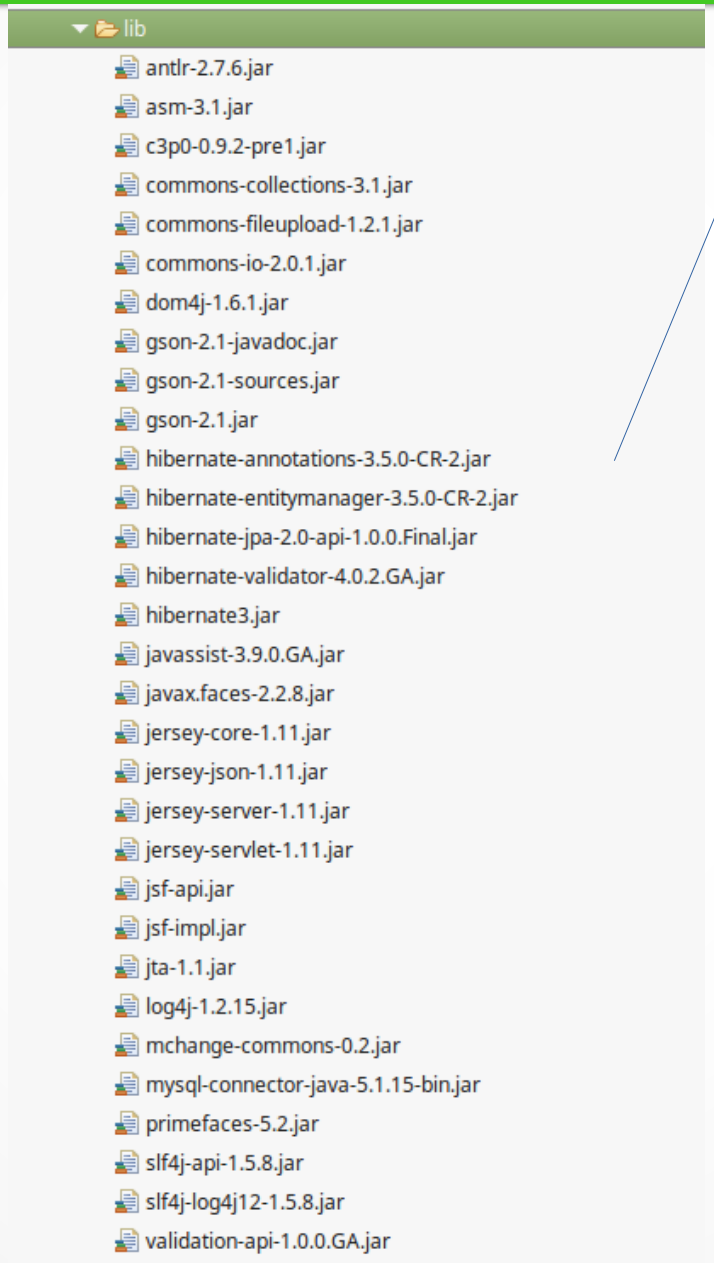
Clique em finish

Projeto Criado !!!!!

Agora vamos começar de verdade...

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





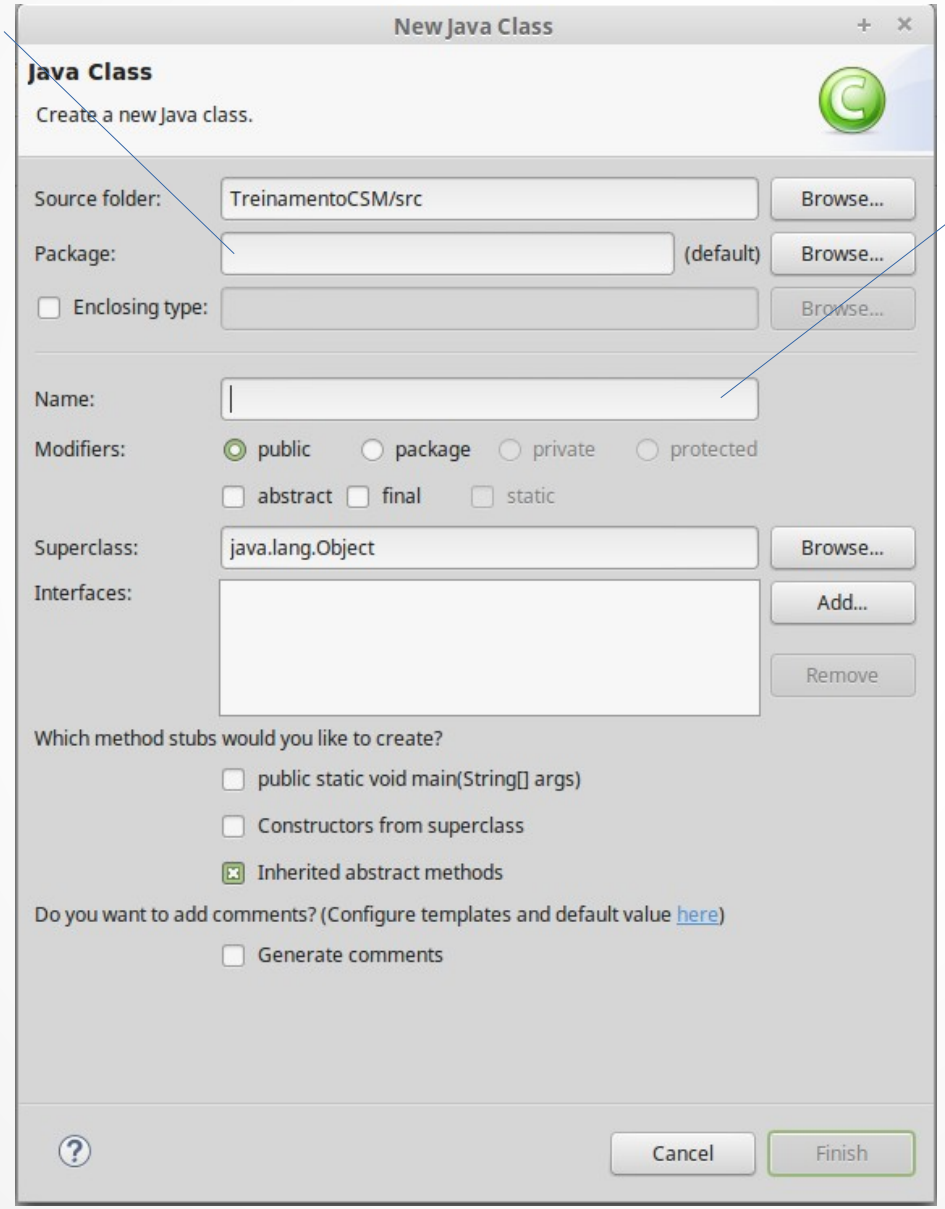
Selecione todos os jars e adicione .build path.

Clicando com o botão Direito e selecionando a opção build path

Vamos para os códigos

Vamos criar
um package para
organizar
nosso código:

br.com.csm.bean

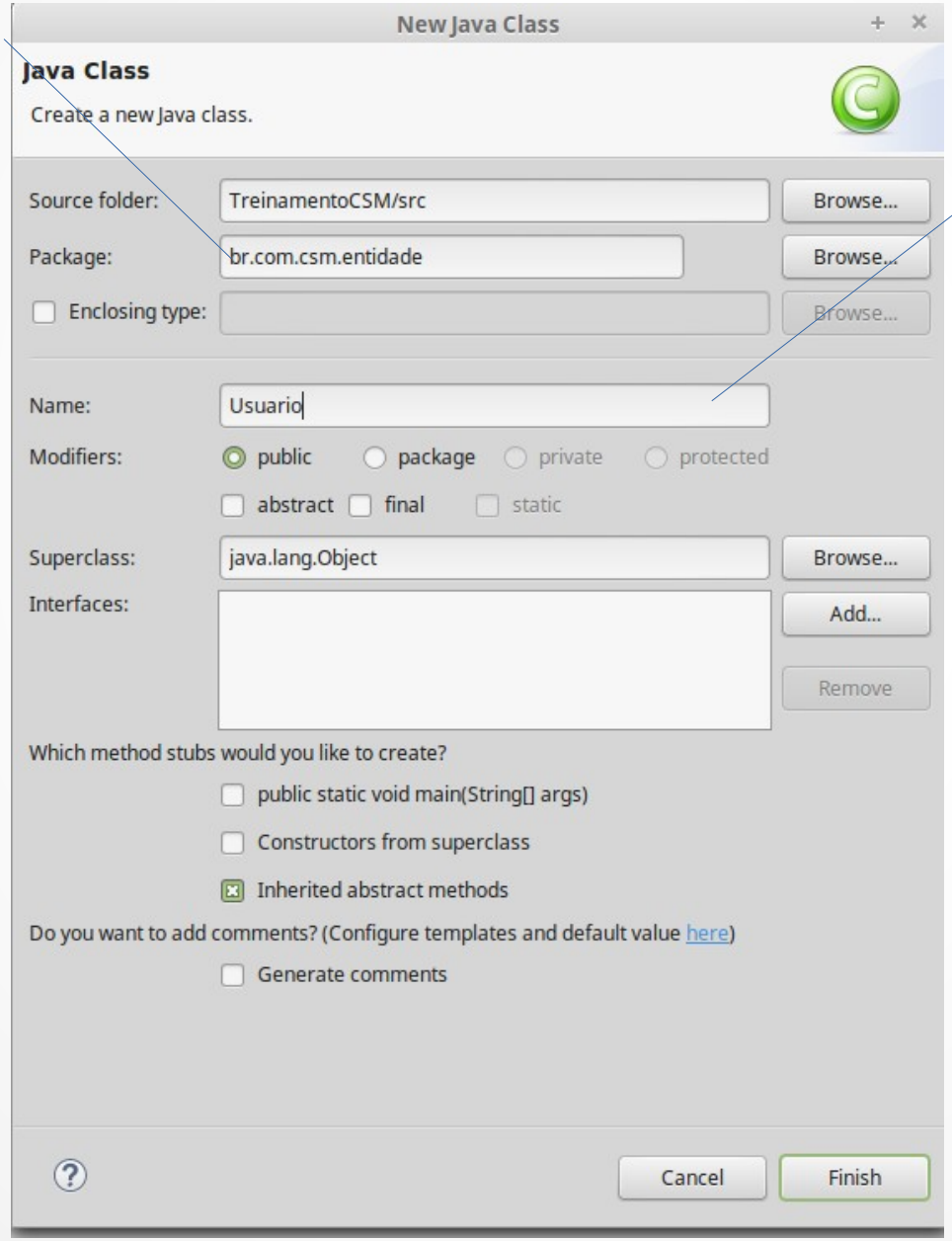


Vamos criar a
classe Usuario

Vamos para os códigos

Vamos criar
um package para
organizar
nosso código:

br.com.csm.bean

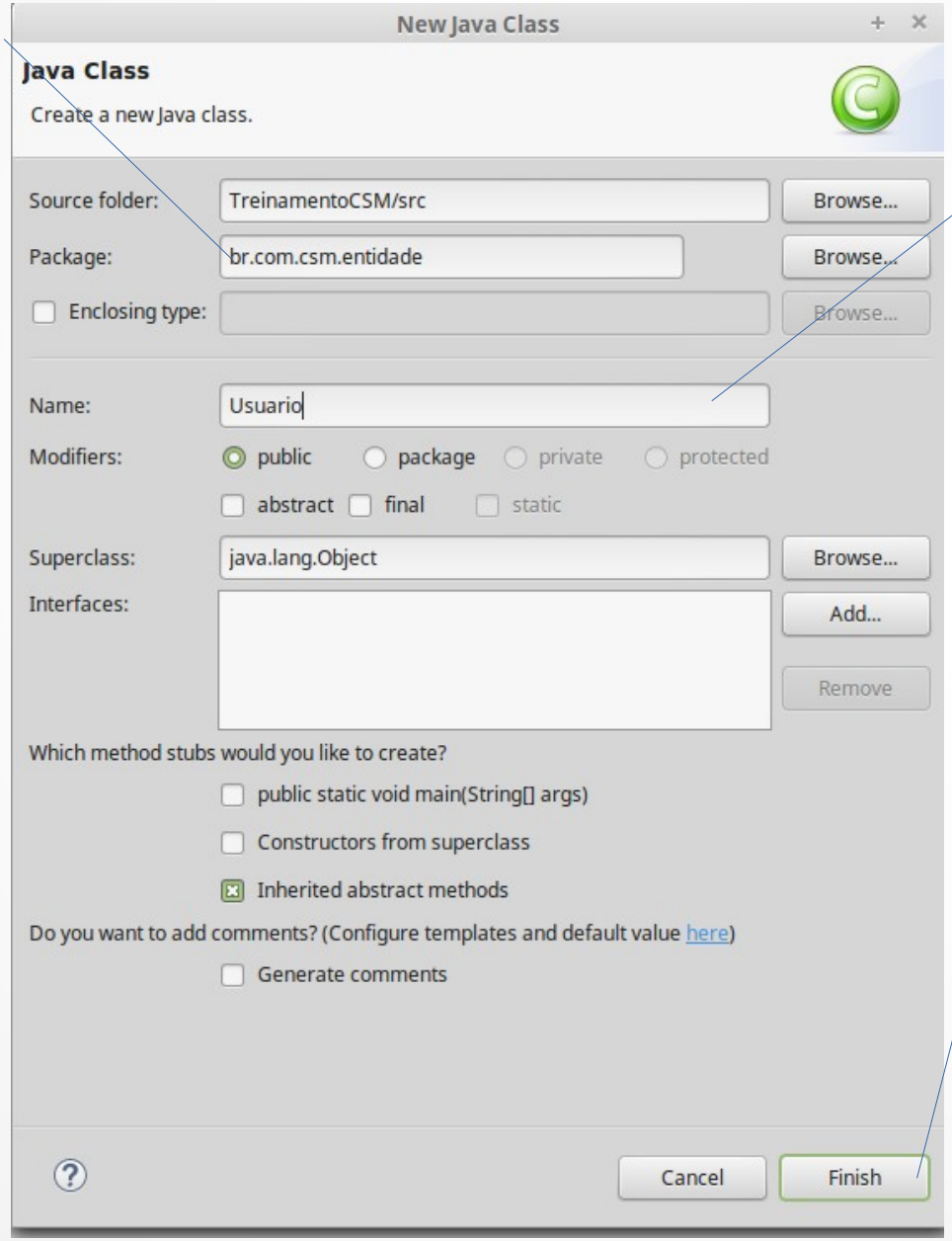


Vamos criar a
classe Usuario

Vamos para os códigos

Vamos criar
um package para
organizar
nosso código:

br.com.csm.bean



Vamos criar a
classe Usuario

Clique
em Finish

Vamos começar a definir os atributos

Usuario.java

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

Criando atributos

Usuario.java

```
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 e um annotation
do Hibernate que simboliza que o mesmo nome da classe sera o da tabela no banco

Usuario.java

```
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 e 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

Usuario.java

```
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 e 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

Usuario.java

```
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 e 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

Usuario.java

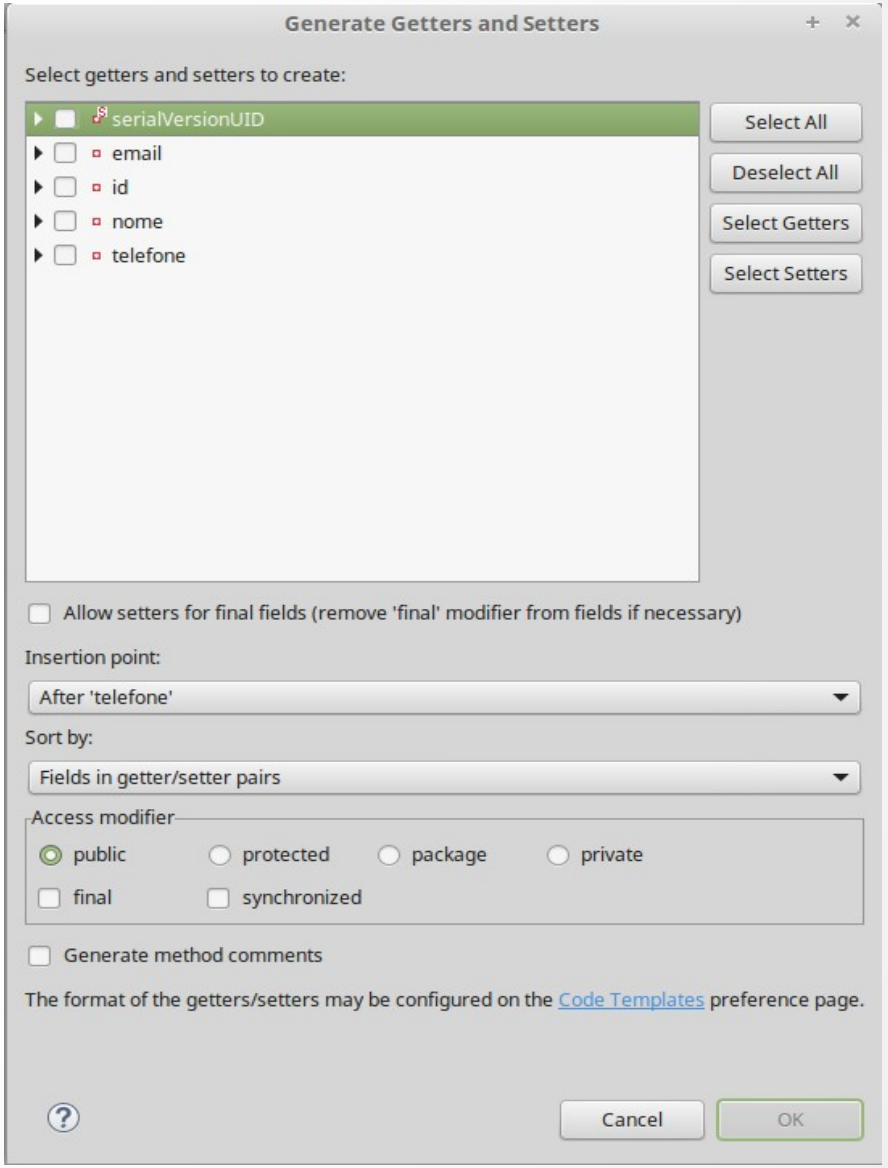
```
1  package br.com.csm.entidade;
2
3  import java.io.Serializable;
4
5  import javax.persistence.Entity;
6  import javax.xml.bind.annotation.XmlRootElement;
7
8  @Entity
9  @XmlRootElement
10 public class Usuario implements Serializable {
11
12     /**
13      *
14      */
15     private static final 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

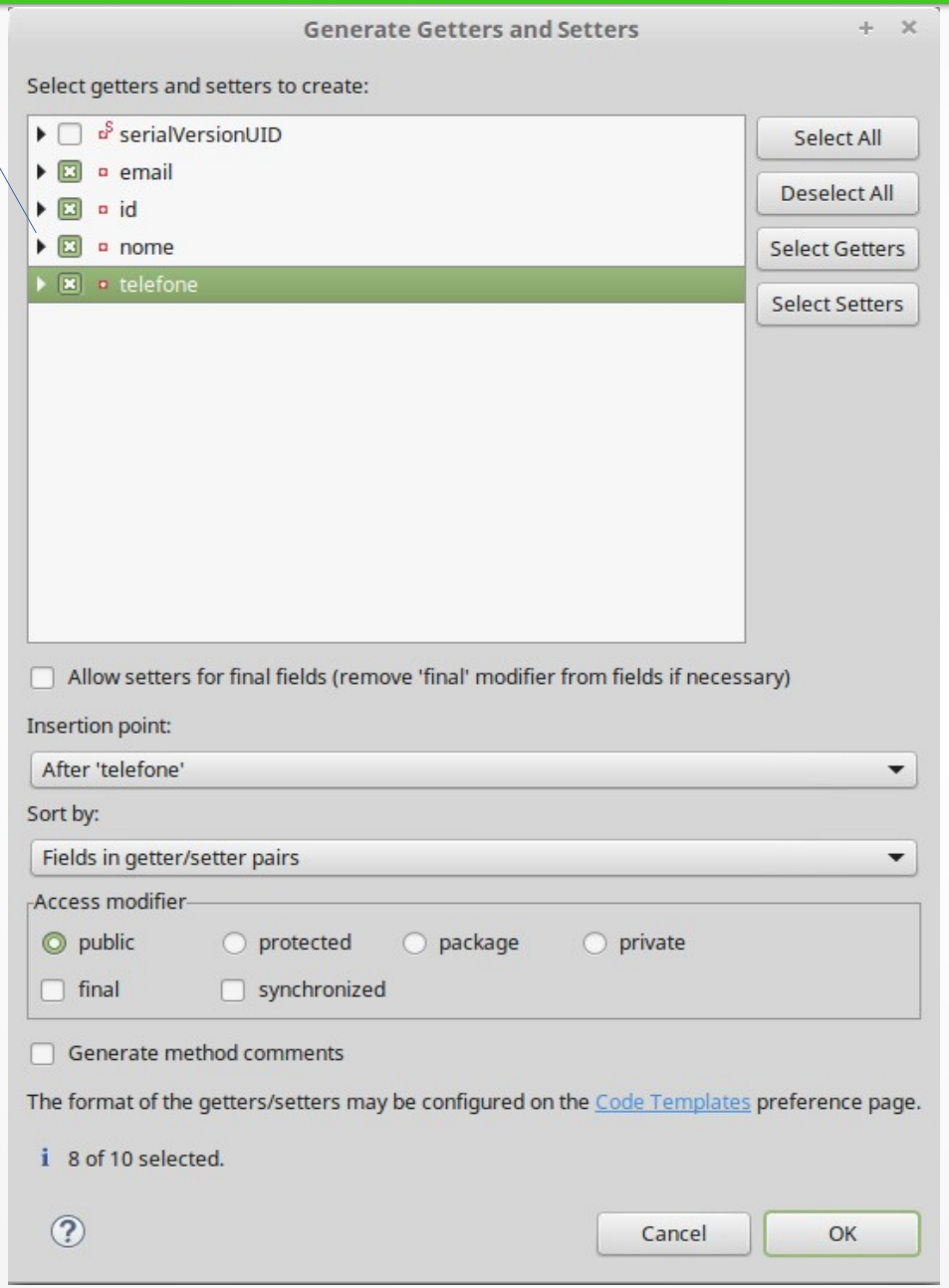
Para nosso exemplo usaremos esses atributos

Vamos criar os gets e sets

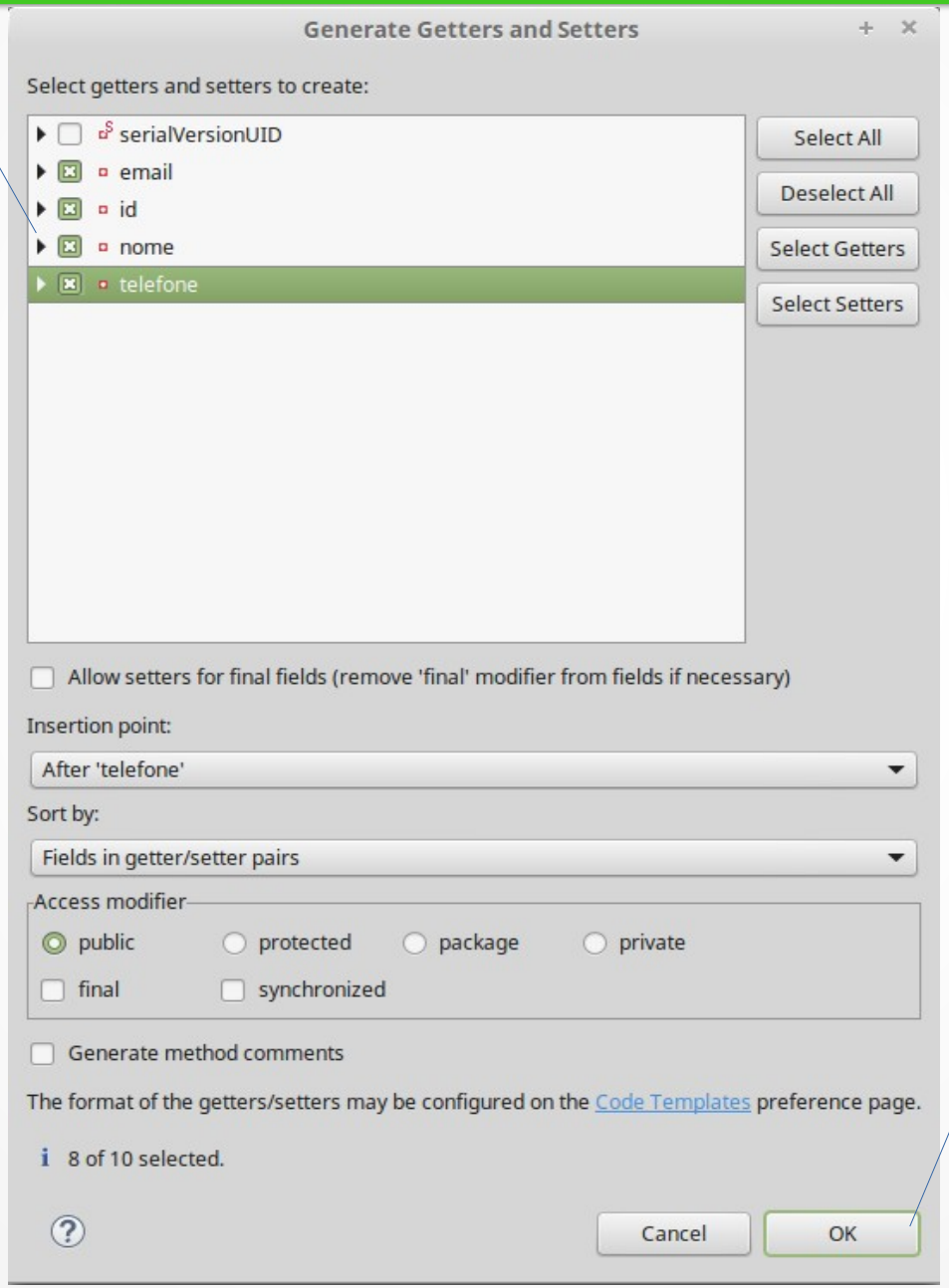
Pressione ctrl+3 e digite ggas



Marque as seguintes opções



Marque as seguintes opções



Clique em OK

```java
@Entity
@XmlRootElement
public class Usuario implements Serializable {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    private long id;
    private String nome;
    private String email;
    private String telefone;

    public long getId() {
        return id;
    }
    public void setId(long id) {
        this.id = id;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getTelefone() {
        return telefone;
    }
    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }
```

```java
@Entity
@XmlRootElement
public class Usuario implements Serializable {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    private long id;
    private String nome;
    private String email;
    private String telefone;

    public long getId() {
        return id;
    }
    public void setId(long id) {
        this.id = id;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getTelefone() {
        return telefone;
    }
    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }
```

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

```java
import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.xml.bind.annotation.XmlRootElement;

@Entity
@XmlRootElement
public class Usuario implements Serializable {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue
    private long id;
    @Column
    private String nome;
    @Column
    private String email;
    @Column
    private String telefone;



    //gets and setters....

```

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

```java
import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.xml.bind.annotation.XmlRootElement;

@Entity
@XmlRootElement
public class Usuario implements Serializable {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue
    private long id;
    @Column
    private String nome;
    @Column
    private String email;
    @Column
    private String telefone;



    //gets and setters....

```

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

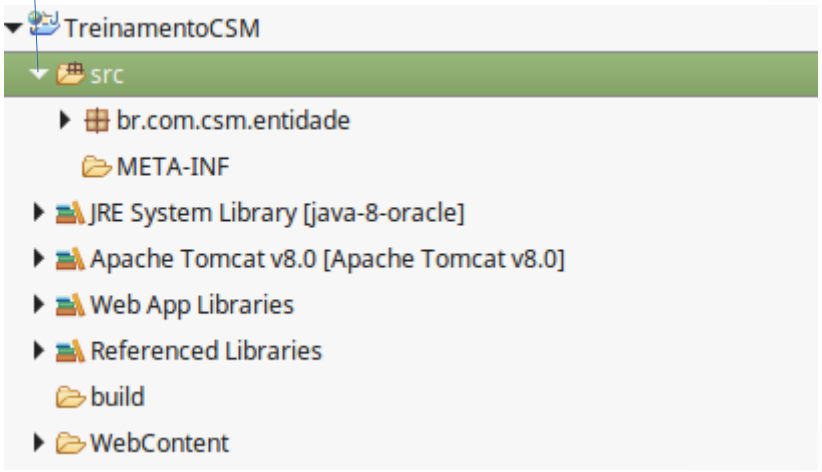
Esse atributo será auto-incremento

```java
import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.xml.bind.annotation.XmlRootElement;

@Entity
@XmlRootElement
public class Usuario implements Serializable {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue
    private long id;
    @Column
    private String nome;
    @Column
    private String email;
    @Column
    private String telefone;



    //gets and setters....

```

@Column e para dizer que Os demais campos São colunas

Antes de prosseguir vamos configurar nossa conexão...

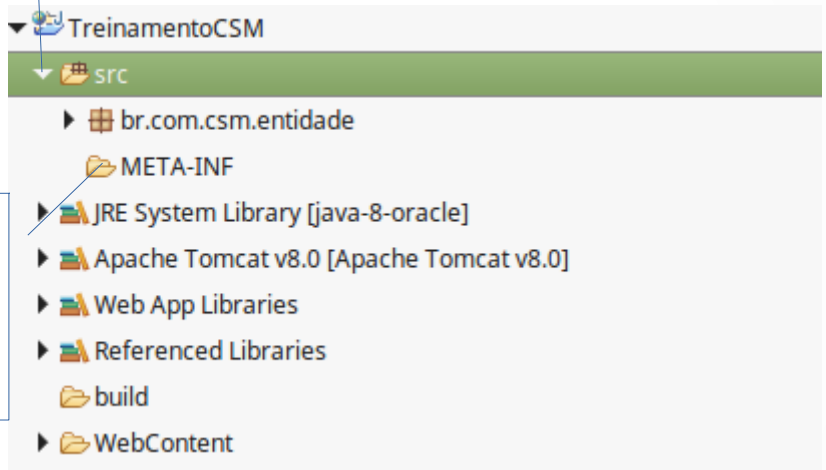
Dentro da pasta src crie uma pasta chamada META-INF



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

Usuario.java | persistence.xml

```xml
<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 name será usado pelo JPA mais tarde

Coloque o nome do seu banco de dados

Usuario.java | persistence.xml

```
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

Usuario.java | persistence.xml | web.xml

```xml
<?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:///j
  <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

Usuario.java | persistence.xml | 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

Usuario.java | persistence.xml | 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

Usuario.java | persistence.xml | web.xml | persistence.xml | web.xml

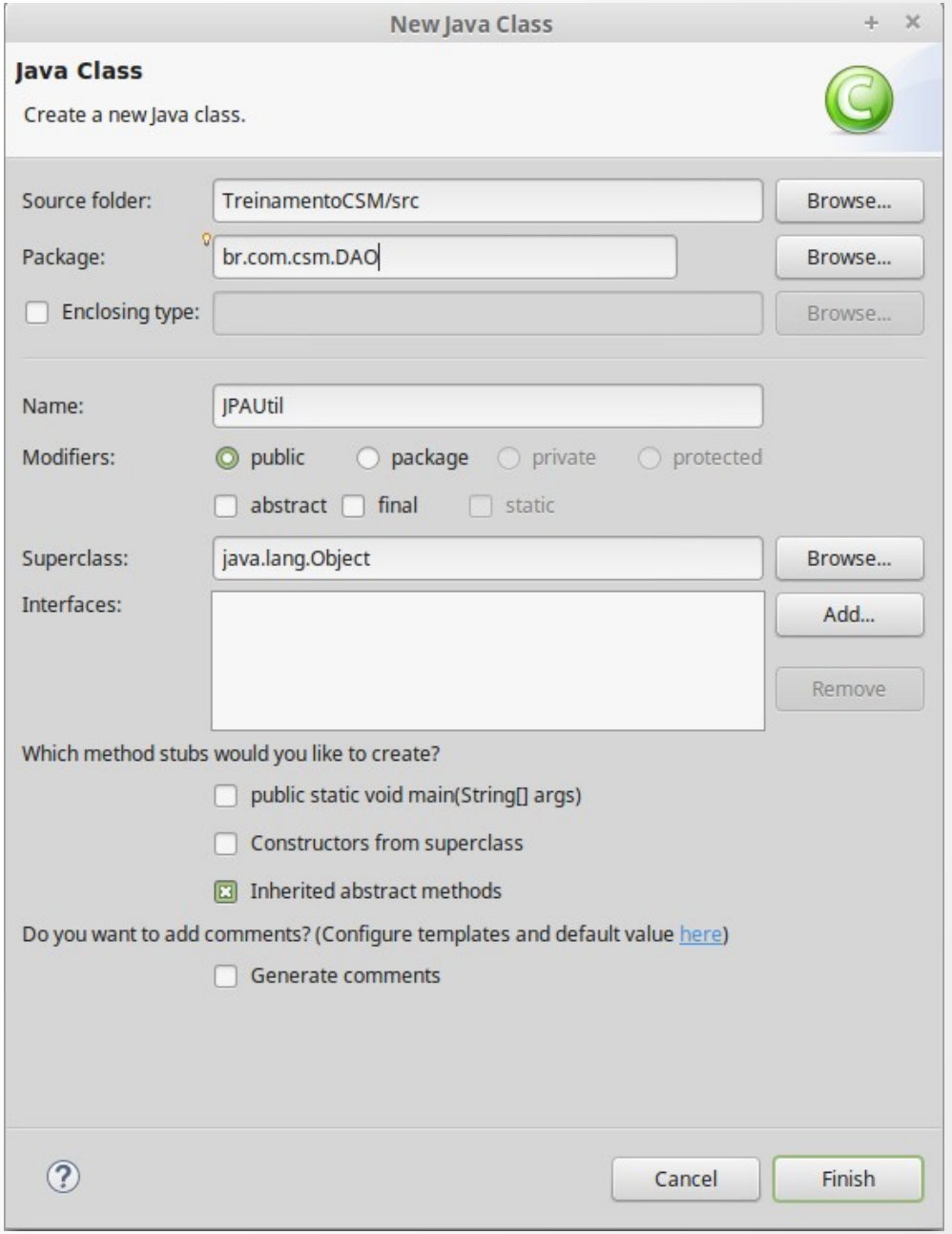
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://
  <display-name>TreinamentoCSM</display-name>
  <welcome-file-list>
    <welcome-file>index.html</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>


  <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>
</web-app>
```

Adicione o código abaixo

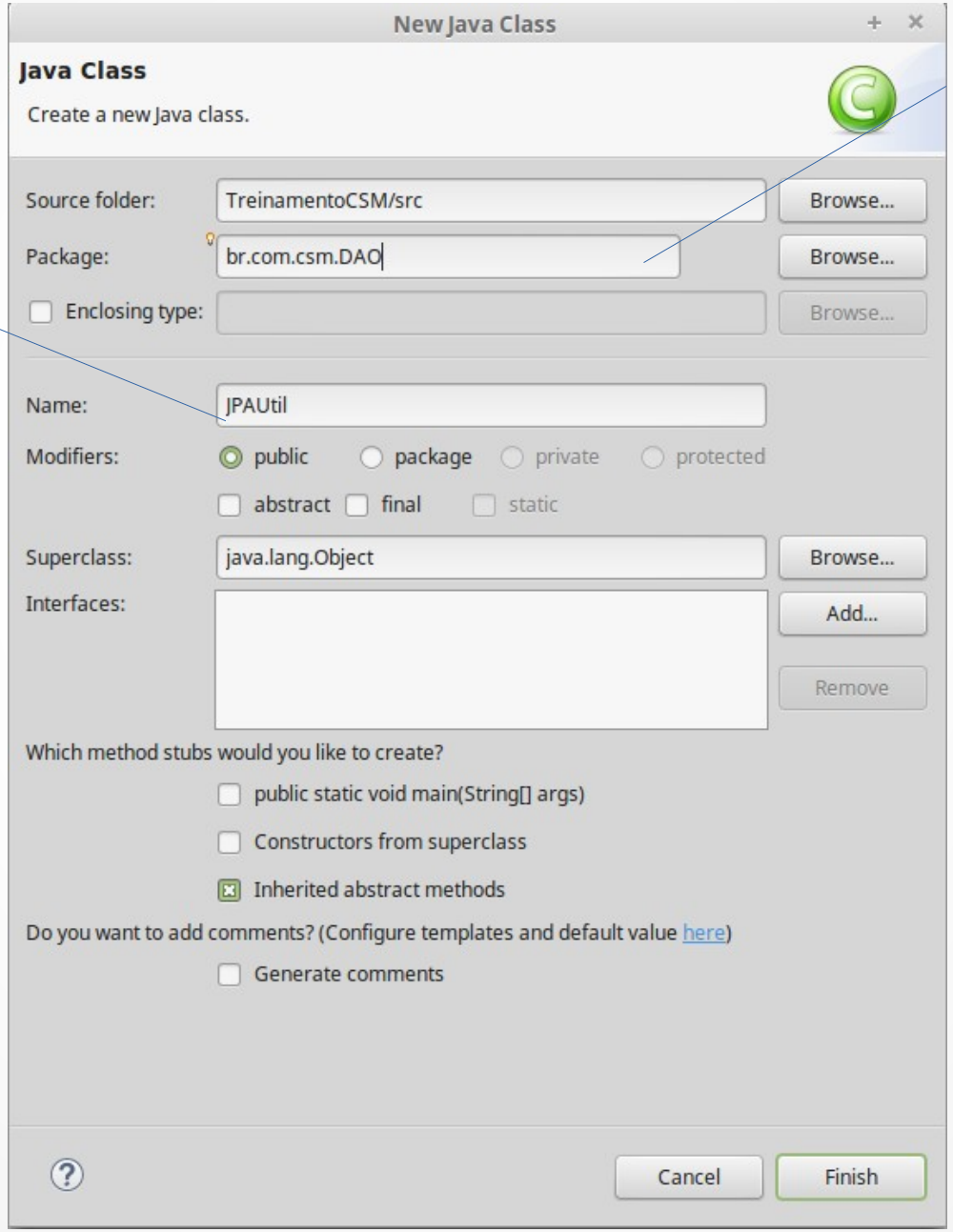
Esse br.com.csm.resources Será explicado mais tarde

Vamos criar nossa JPAUtil

- Crie uma classe chamada JPAUtil

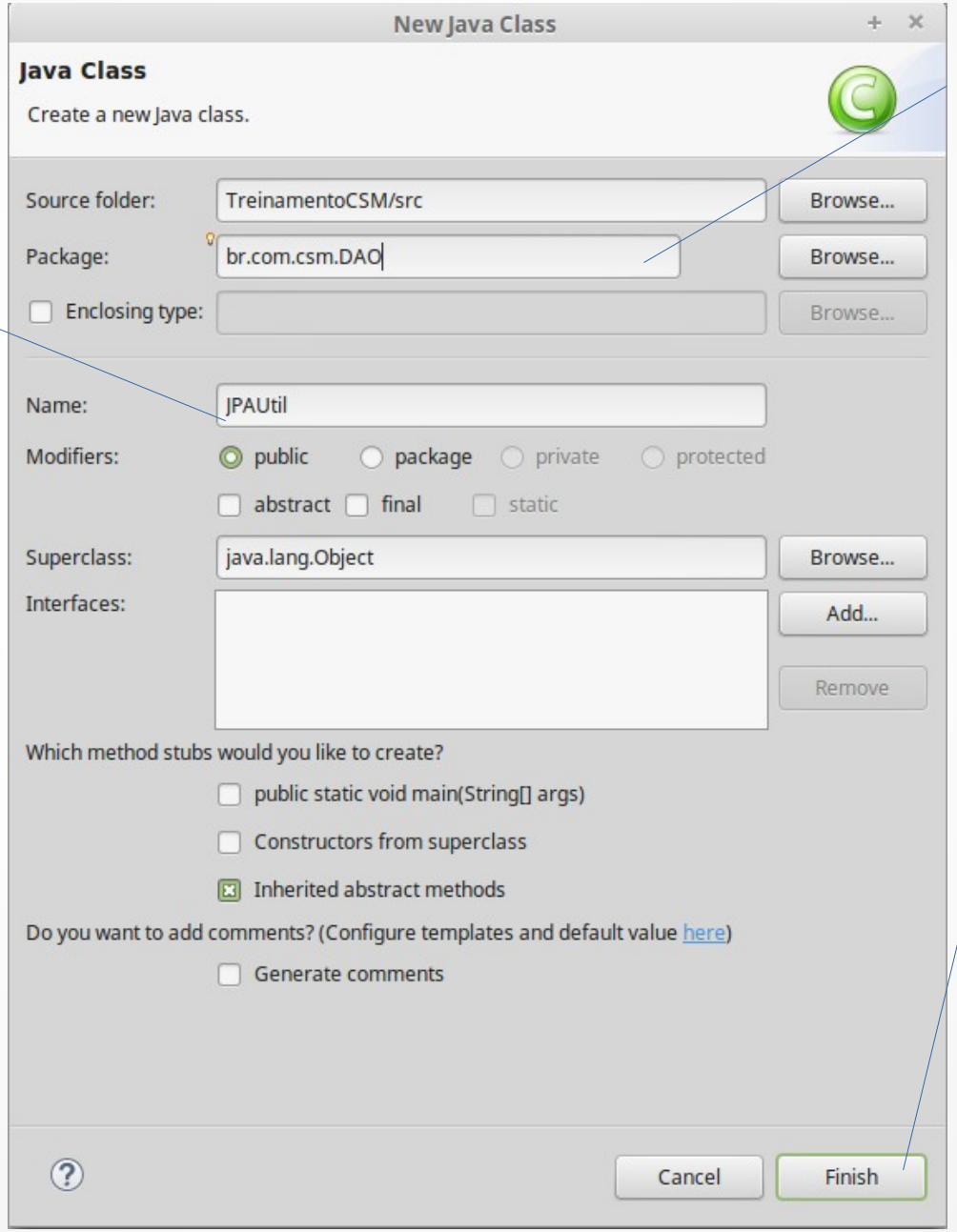


Nome da Classe



Nome do package

Nome da Classe



Nome do package

Clique em Finish

Preencha a classe JPAUtil
Com o seguinte conteúdo

Usuario.java | persistence.xml | web.xml | JPAUtil.java

```
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

Usuario.java | persistence.xml | web.xml | JPAUtil.java

```
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

Usuario.java | persistence.xml

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

Preencha a classe JPAUtil
Com o seguinte conteúdo

Usuario.java | persistence.xml | web.xml | JPAUtil.java

```
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

Usuario.java | persistence.xml

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

Preencha a classe JPAUtil
Com o seguinte conteúdo

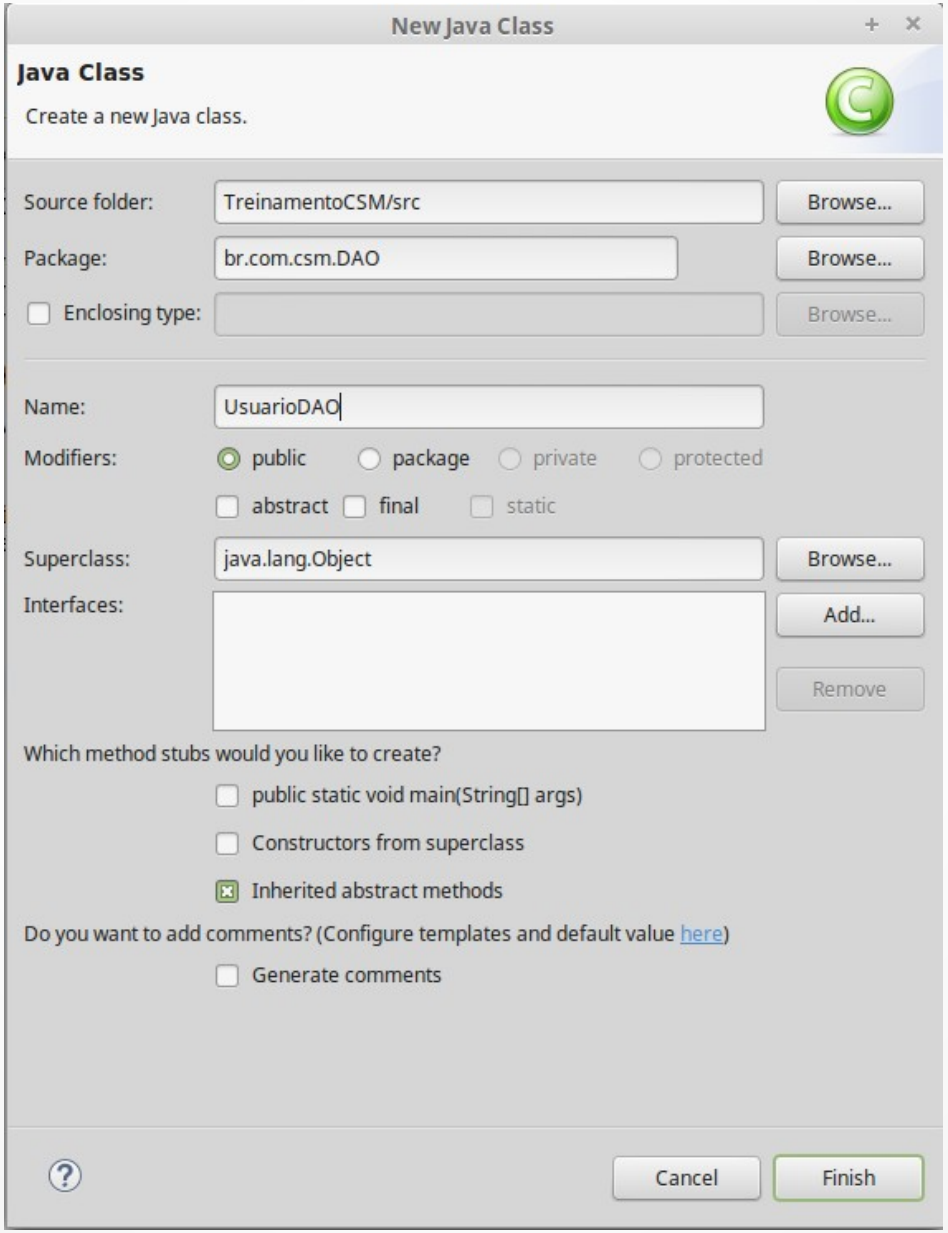
Usuario.java | persistence.xml | web.xml | JPAUtil.java

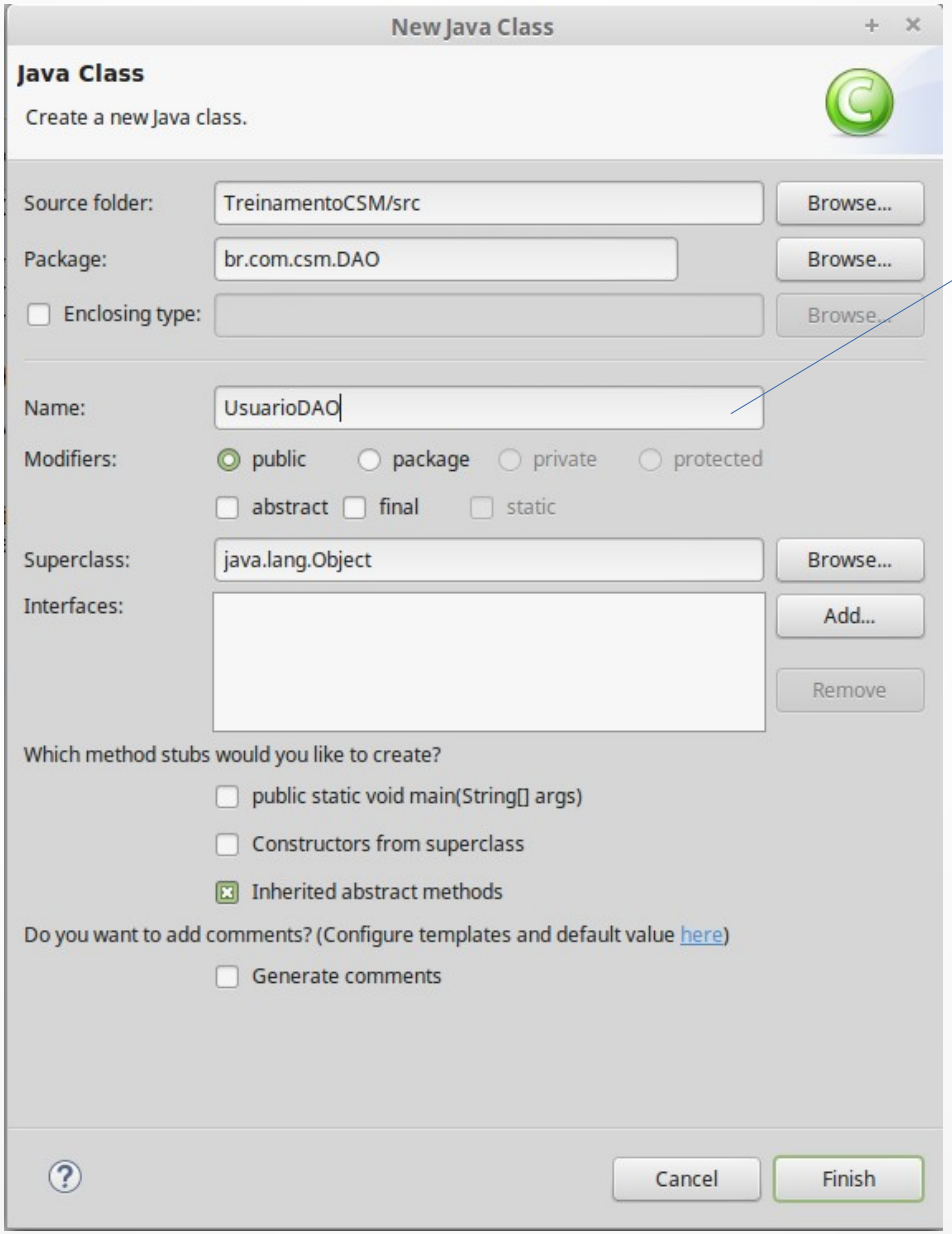
```
1  package br.com.csm.DAO;
2
3  import javax.persistence.EntityManager;
4  import javax.persistence.EntityManagerFactory;
5  import javax.persistence.Persistence;
6
7  public class JPAUtil {
8
9  private static EntityManagerFactory emf = Persistence.createEntityManagerFactory("DB");
10
11
12    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

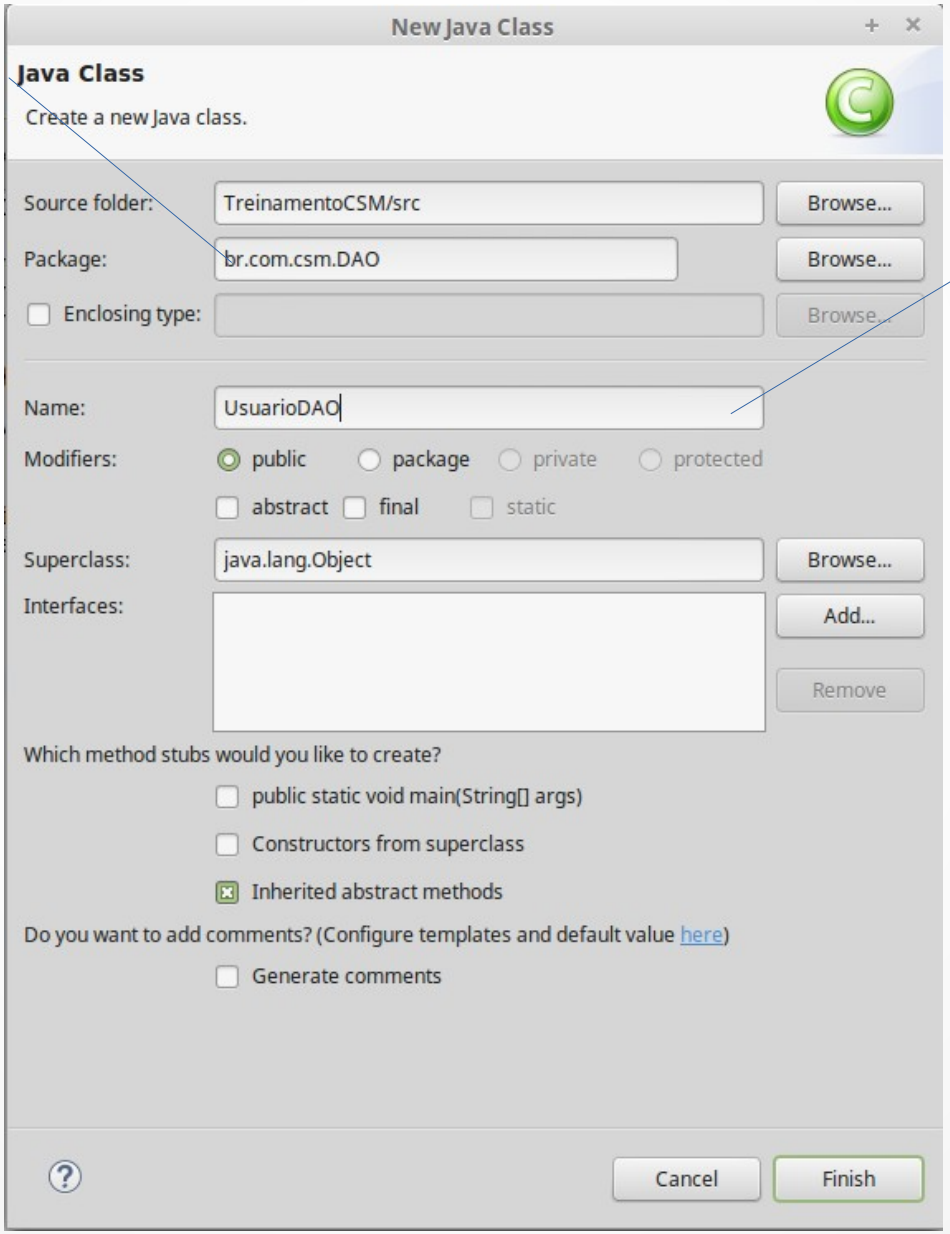
Vamos criar uma classe chamada UsuarioDAO





Nome da classe

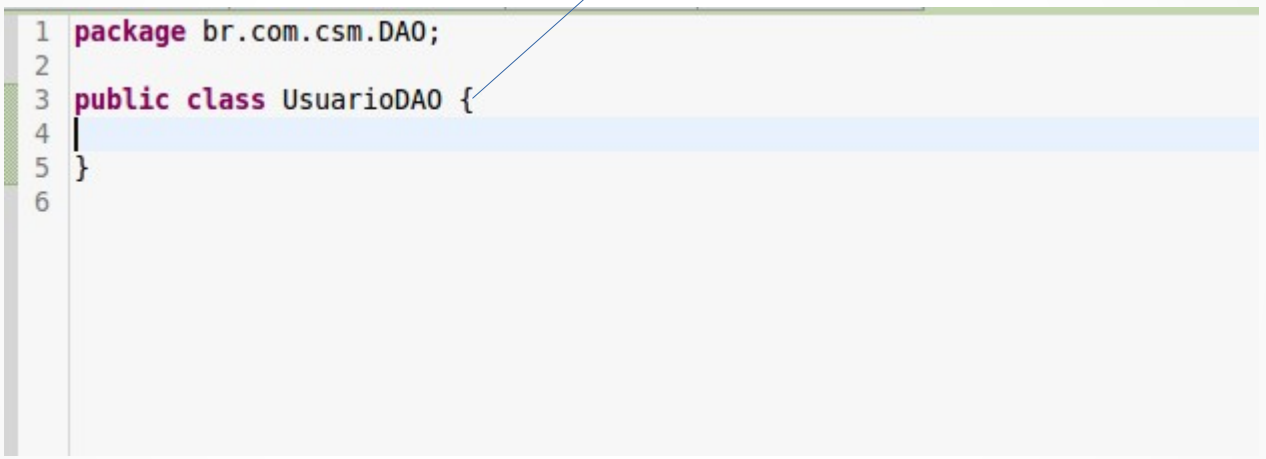
No mesmo package que já tinha sido criado



Nome da classe

```java
package br.com.csm.DAO;

public class UsuarioDAO {

}

```

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
```

```java
package br.com.csm.DAO;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.Query;

import br.com.csm.entidade.Usuario;

public class UsuarioDAO {

    private EntityManager entityManager;

    public UsuarioDAO(EntityManager entityManager){
        this.entityManager= entityManager;
    }

    public void cadastrar(Usuario usuario){
        entityManager.persist(usuario);
    }

    public void alterar(Usuario usuario){
        entityManager.merge(usuario);
    }

    public void excluir(Usuario usuario){
        entityManager.remove(entityManager.merge(usuario));
    }

    public Usuario consultar(Long id){
        return entityManager.getReference(Usuario.class, id);
    }

    public List<Usuario> listar(){

        String sql = "Select usu from Usuario usu order by nome";
        Query query = entityManager.createQuery(sql);

        return query.getResultList();
    }

}
```

```java
package br.com.csm.DAO;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.Query;

import br.com.csm.entidade.Usuario;

public class UsuarioDAO {

    private EntityManager entityManager;

    public UsuarioDAO(EntityManager entityManager){
        this.entityManager= entityManager;
    }

    public void cadastrar(Usuario usuario){
        entityManager.persist(usuario);
    }

    public void alterar(Usuario usuario){
        entityManager.merge(usuario);
    }

    public void excluir(Usuario usuario){
        entityManager.remove(entityManager.merge(usuario));
    }

    public Usuario consultar(Long id){
        return entityManager.getReference(Usuario.class, id);
    }

    public List<Usuario> listar(){

        String sql = "Select usu from Usuario usu order by nome";
        Query query = entityManager.createQuery(sql);

        return query.getResultList();
    }

}
```

Temos os métodos básicos Fornecidos pelo hibernate

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

```java
package br.com.csm.DAO;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.Query;

import br.com.csm.entidade.Usuario;

public class UsuarioDAO {

    private EntityManager entityManager;

    public UsuarioDAO(EntityManager entityManager){
        this.entityManager= entityManager;
    }

    public void cadastrar(Usuario usuario){
        entityManager.persist(usuario);
    }

    public void alterar(Usuario usuario){
        entityManager.merge(usuario);
    }

    public void excluir(Usuario usuario){
        entityManager.remove(entityManager.merge(usuario));
    }

    public Usuario consultar(Long id){
        return entityManager.getReference(Usuario.class, id);
    }

    public List<Usuario> listar(){

        String sql = "Select usu from Usuario usu order by nome";
        Query query = entityManager.createQuery(sql);

        return query.getResultList();
    }

}
```

Temos os métodos básicos Fornecidos pelo hibernate

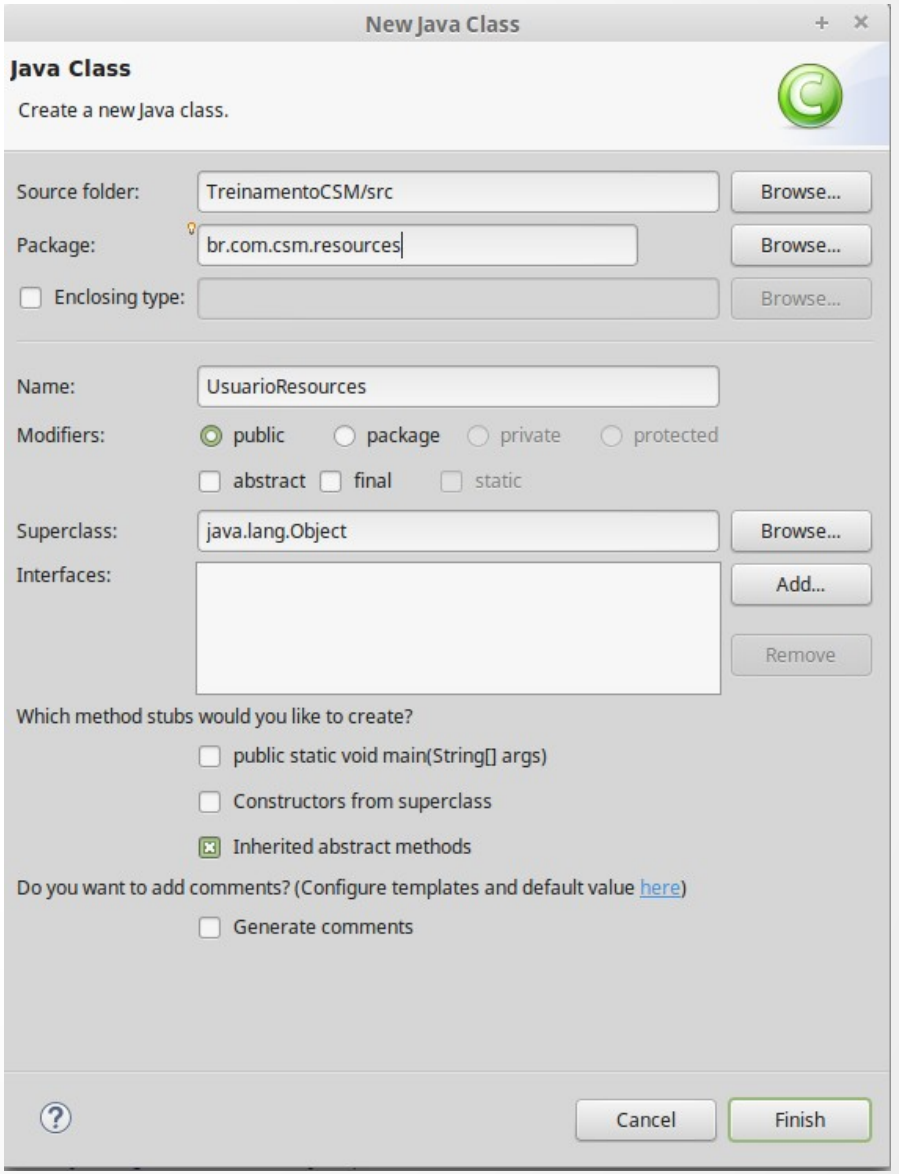
Precisamos de um construtor da classe para passarmos a instância da conexão

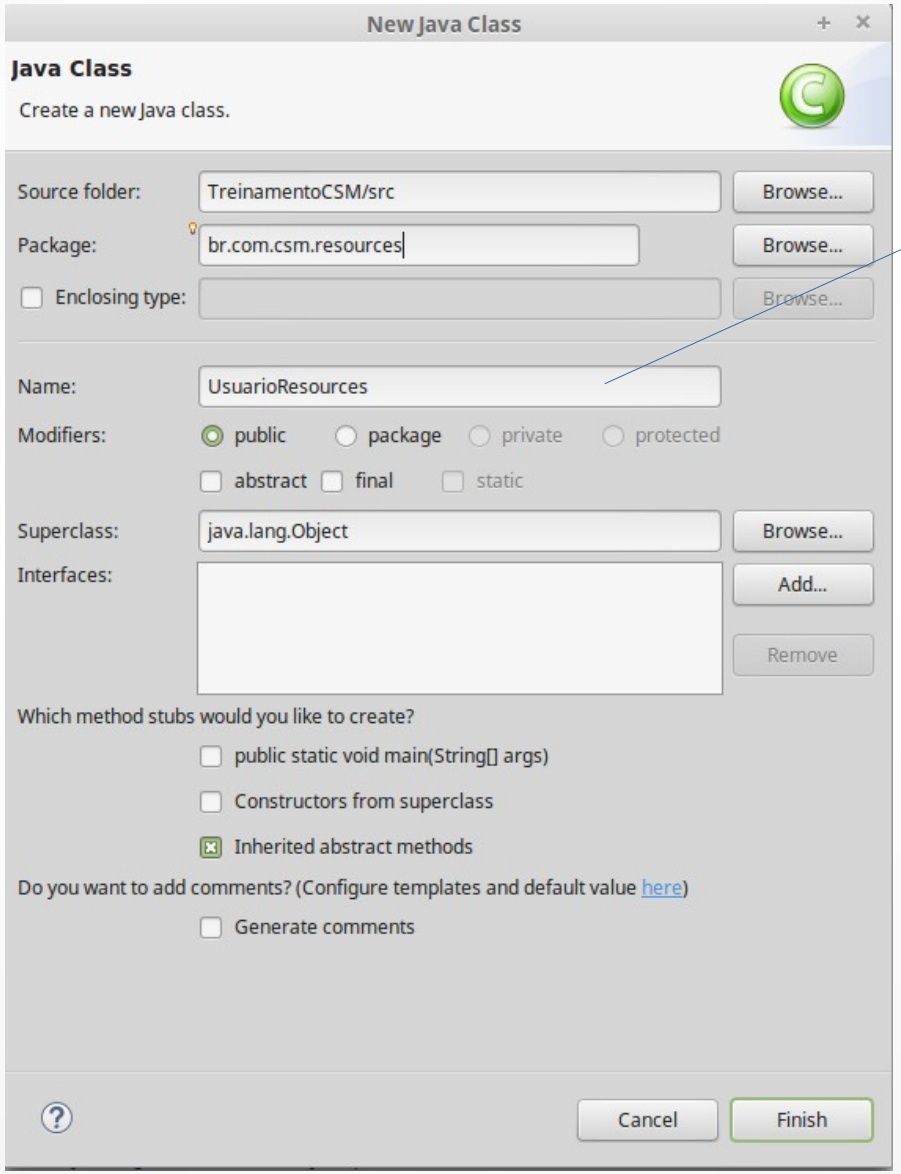
```java
package br.com.csm.DAO;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.Query;

import br.com.csm.entidade.Usuario;

public class UsuarioDAO {

    private EntityManager entityManager;

    public UsuarioDAO(EntityManager entityManager){
        this.entityManager= entityManager;
    }

    public void cadastrar(Usuario usuario){
        entityManager.persist(usuario);
    }

    public void alterar(Usuario usuario){
        entityManager.merge(usuario);
    }

    public void excluir(Usuario usuario){
        entityManager.remove(entityManager.merge(usuario));
    }

    public Usuario consultar(Long id){
        return entityManager.getReference(Usuario.class, id);
    }

    public List<Usuario> listar(){

        String sql = "Select usu from Usuario usu order by nome";
        Query query = entityManager.createQuery(sql);

        return query.getResultList();
    }

}
```

Temos os métodos básicos Fornecidos pelo hibernate

ATENÇÃO NOS IMPORTS

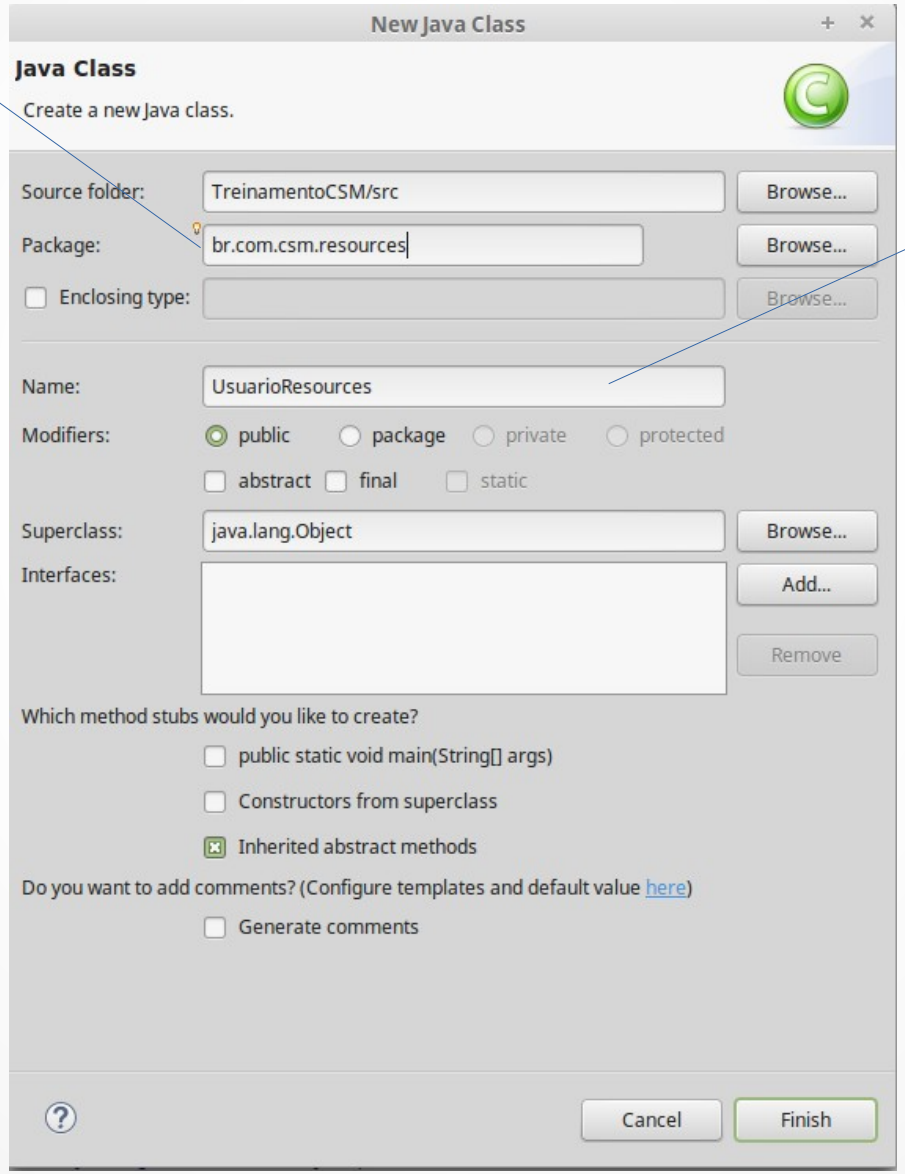
- Vamos criar uma classe chamada UsuarioResources





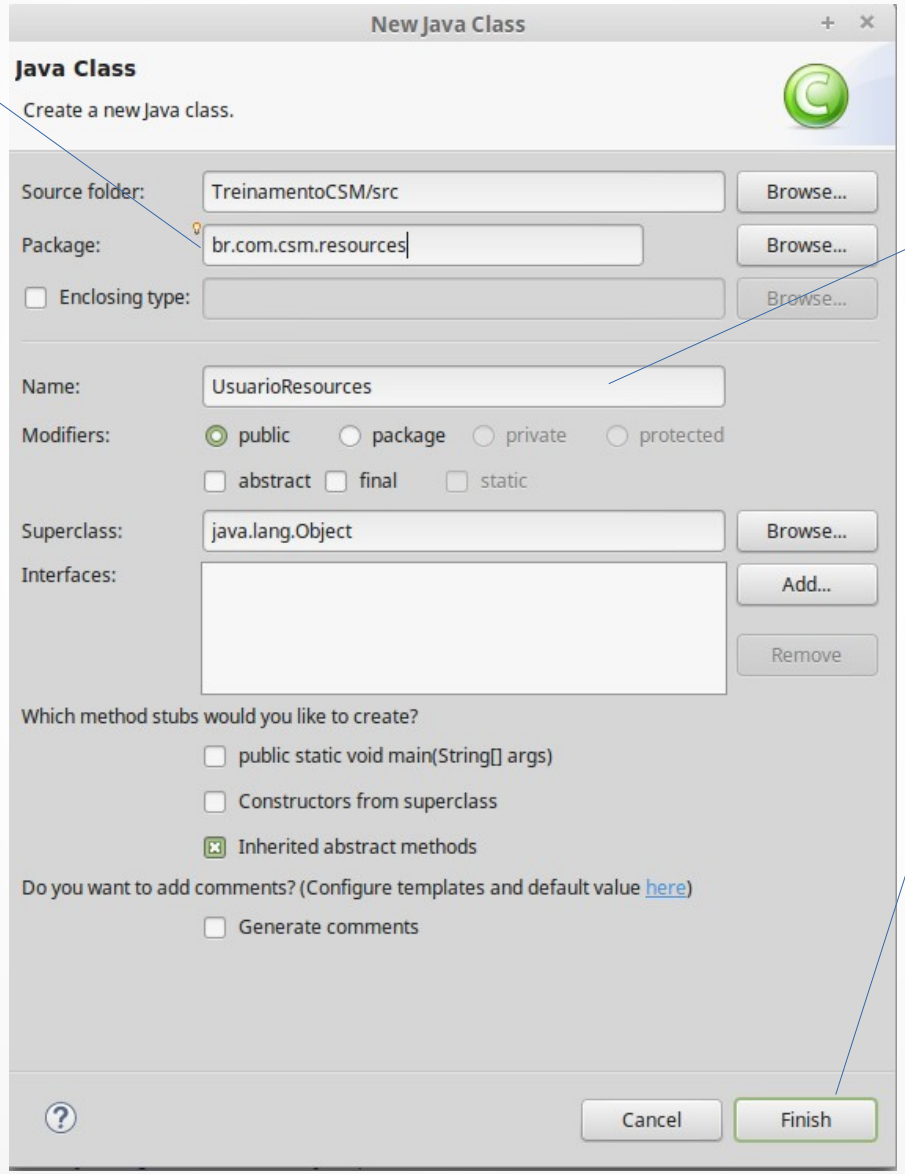
Nome da classe

Nome do package



Nome da classe

Nome do package



Nome da classe

Clique em Finish

```java
package br.com.csm.resources;

public class UsuarioResources {

}

```

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

```java

import java.util.ArrayList;
import java.util.List;

import javax.persistence.EntityManager;
import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import br.com.csm.DAO.JPAUtil;
import br.com.csm.DAO.UsuarioDAO;
import br.com.csm.entidade.Usuario;

@Path("/usuarios")
public class UsuarioResources {


    @GET
    @Path("/listarUsuarios")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public List<Usuario> listUsuarios(){
        List<Usuario> list = new ArrayList<Usuario>();
        EntityManager em = JPAUtil.getEntityManager();
        UsuarioDAO dao = new UsuarioDAO(em);
        list=dao.listar();

        return list;

    }

    @POST
    @Path("/consultarUsuario")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Usuario buscarUsuario(long id){
        Usuario usuario = new Usuario();
        EntityManager em = JPAUtil.getEntityManager();
        UsuarioDAO dao = new UsuarioDAO(em);
        usuario = dao.consultar(id);

        return usuario;
    }


}
```

```java
import java.util.ArrayList;
import java.util.List;

import javax.persistence.EntityManager;
import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import br.com.csm.DAO.JPAUtil;
import br.com.csm.DAO.UsuarioDAO;
import br.com.csm.entidade.Usuario;

@Path("/usuarios")
public class UsuarioResources {


    @GET
    @Path("/listarUsuarios")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public List<Usuario> listUsuarios(){
        List<Usuario> list = new ArrayList<Usuario>();
        EntityManager em = JPAUtil.getEntityManager();
        UsuarioDAO dao = new UsuarioDAO(em);
        list=dao.listar();

        return list;

    }

    @POST
    @Path("/consultarUsuario")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Usuario buscarUsuario(long id){
        Usuario usuario = new Usuario();
        EntityManager em = JPAUtil.getEntityManager();
        UsuarioDAO dao = new UsuarioDAO(em);
        usuario = dao.consultar(id);

        return usuario;
    }


}
```

Essa notação serve para referenciar qual classe resource iremos trabalhos

```java
import java.util.ArrayList;
import java.util.List;

import javax.persistence.EntityManager;
import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import br.com.csm.DAO.JPAUtil;
import br.com.csm.DAO.UsuarioDAO;
import br.com.csm.entidade.Usuario;

@Path("/usuarios")
public class UsuarioResources {


    @GET
    @Path("/listarUsuarios")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public List<Usuario> listUsuarios(){
        List<Usuario> list = new ArrayList<Usuario>();
        EntityManager em = JPAUtil.getEntityManager();
        UsuarioDAO dao = new UsuarioDAO(em);
        list=dao.listar();

        return list;

    }

    @POST
    @Path("/consultarUsuario")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Usuario buscarUsuario(long id){
        Usuario usuario = new Usuario();
        EntityManager em = JPAUtil.getEntityManager();
        UsuarioDAO dao = new UsuarioDAO(em);
        usuario = dao.consultar(id);

        return usuario;
    }


}
```

Essa notação serve para referenciar qual classe resource iremos trabalhos

Usando a notação GET

Essa notação para informamos qual métodos iremos acessar

```java
import java.util.ArrayList;
import java.util.List;

import javax.persistence.EntityManager;
import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import br.com.csm.DAO.JPAUtil;
import br.com.csm.DAO.UsuarioDAO;
import br.com.csm.entidade.Usuario;

@Path("/usuarios")
public class UsuarioResources {


    @GET
    @Path("/listarUsuarios")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public List<Usuario> listUsuarios(){
        List<Usuario> list = new ArrayList<Usuario>();
        EntityManager em = JPAUtil.getEntityManager();
        UsuarioDAO dao = new UsuarioDAO(em);
        list=dao.listar();

        return list;

    }

    @POST
    @Path("/consultarUsuario")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Usuario buscarUsuario(long id){
        Usuario usuario = new Usuario();
        EntityManager em = JPAUtil.getEntityManager();
        UsuarioDAO dao = new UsuarioDAO(em);
        usuario = dao.consultar(id);

        return usuario;
    }


}
```

Essa notação serve para referenciar qual classe resource iremos trabalhos

Usando a notação GET

Essa notação para informamos qual métodos iremos acessar

Essa notação serve para referenciar qual classe resource iremos trabalhos

Esse notação é a mágica do Jersey

Usando a notação GET

```
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> listUsuarios(){
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 para informamos qual métodos iremos acessar

Esse notação é a mágica do Jersey

```java
import java.util.ArrayList;
import java.util.List;

import javax.persistence.EntityManager;
import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import br.com.csm.DAO.JPAUtil;
import br.com.csm.DAO.UsuarioDAO;
import br.com.csm.entidade.Usuario;

@Path("/usuarios")
public class UsuarioResources {


    @GET
    @Path("/listarUsuarios")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public List<Usuario> listUsuarios(){
        List<Usuario> list = new ArrayList<Usuario>();
        EntityManager em = JPAUtil.getEntityManager();
        UsuarioDAO dao = new UsuarioDAO(em);
        list=dao.listar();

        return list;

    }

    @POST
    @Path("/consultarUsuario")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Usuario buscarUsuario(long id){
        Usuario usuario = new Usuario();
        EntityManager em = JPAUtil.getEntityManager();
        UsuarioDAO dao = new UsuarioDAO(em);
        usuario = dao.consultar(id);

        return usuario;
    }


}
```

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
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> listUsuarios(){
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 }
51
```

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

```java
import java.util.ArrayList;
import java.util.List;

import javax.persistence.EntityManager;
import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import br.com.csm.DAO.JPAUtil;
import br.com.csm.DAO.UsuarioDAO;
import br.com.csm.entidade.Usuario;

@Path("/usuarios")
public class UsuarioResources {


    @GET
    @Path("/listarUsuarios")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public List<Usuario> listUsuarios(){
        List<Usuario> list = new ArrayList<Usuario>();
        EntityManager em = JPAUtil.getEntityManager();
        UsuarioDAO dao = new UsuarioDAO(em);
        list=dao.listar();

        return list;

    }

    @POST
    @Path("/consultarUsuario")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Usuario buscarUsuario(long id){
        Usuario usuario = new Usuario();
        EntityManager em = JPAUtil.getEntityManager();
        UsuarioDAO dao = new UsuarioDAO(em);
        usuario = dao.consultar(id);

        return usuario;
    }


}
```

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

Essa notação serve para referenciar qual classe resource iremos trabalhos

Esse notação é a mágica do Jersey

Usando a notação GET

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

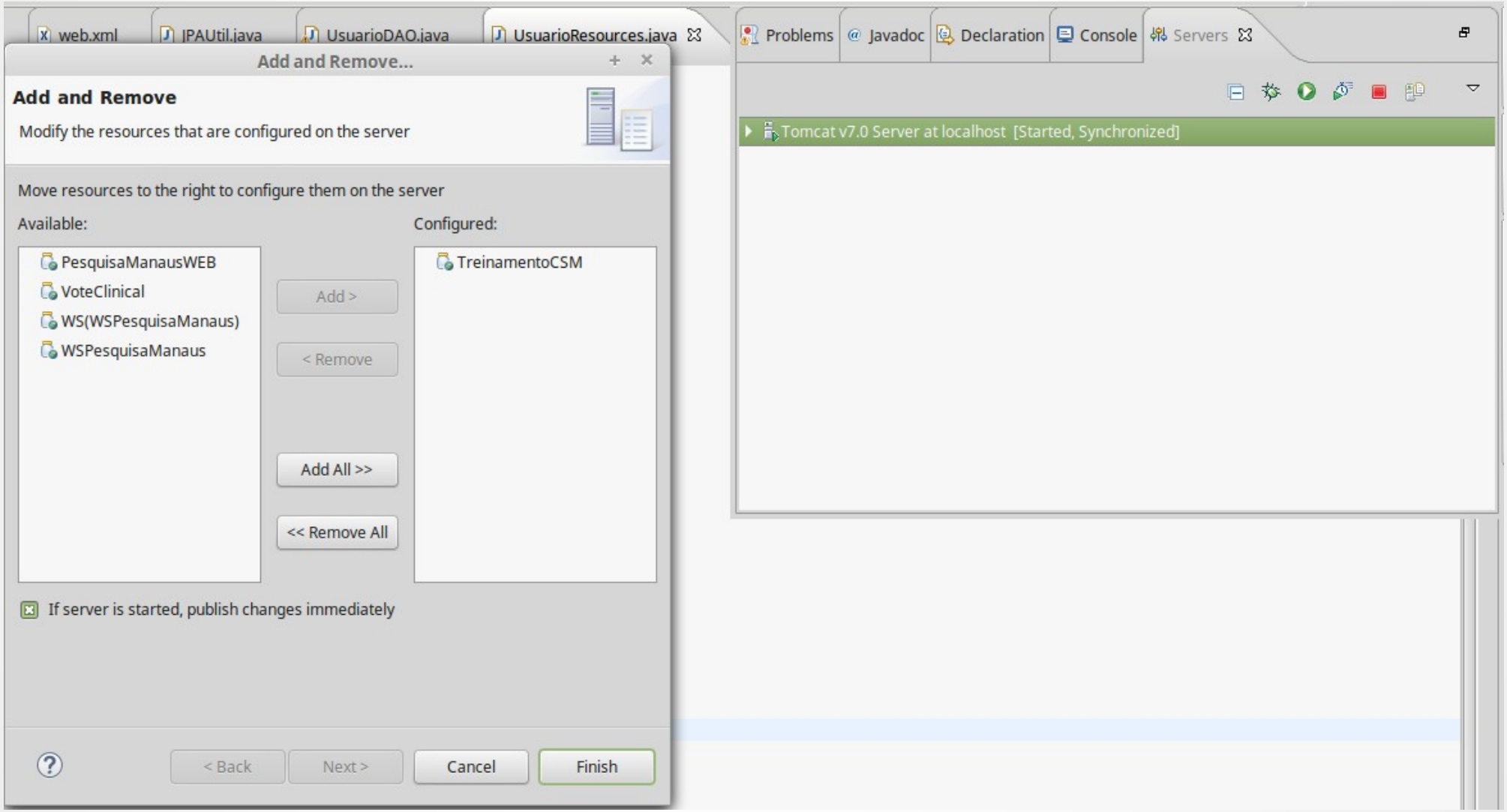
Ele irá transformar o retorno desse método em conteúdo JSON

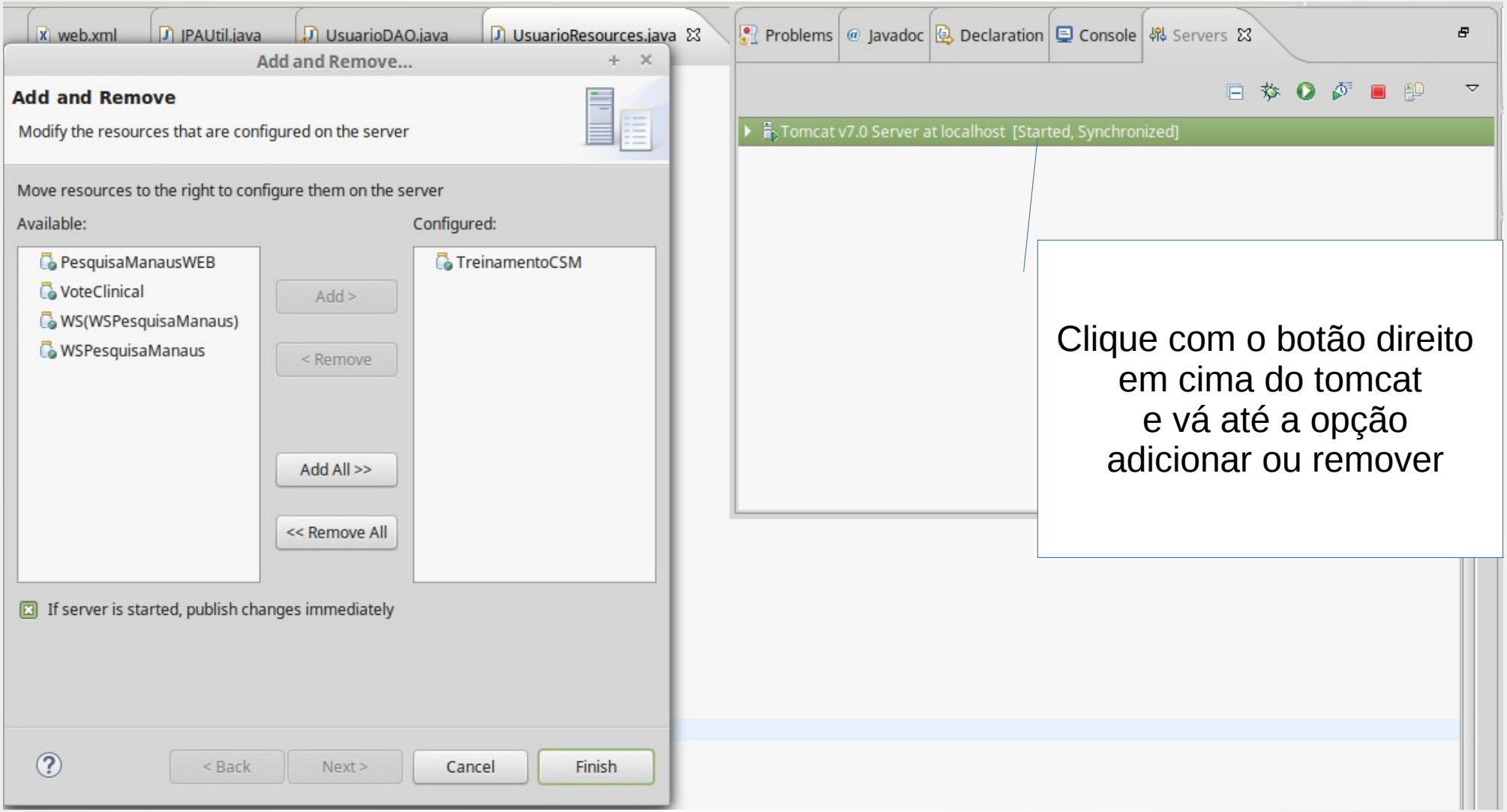
No nosso exemplo só usaremos o GET, mas fica de brinde como fazer usando POST

```
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> listUsuarios(){
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 }
51
```

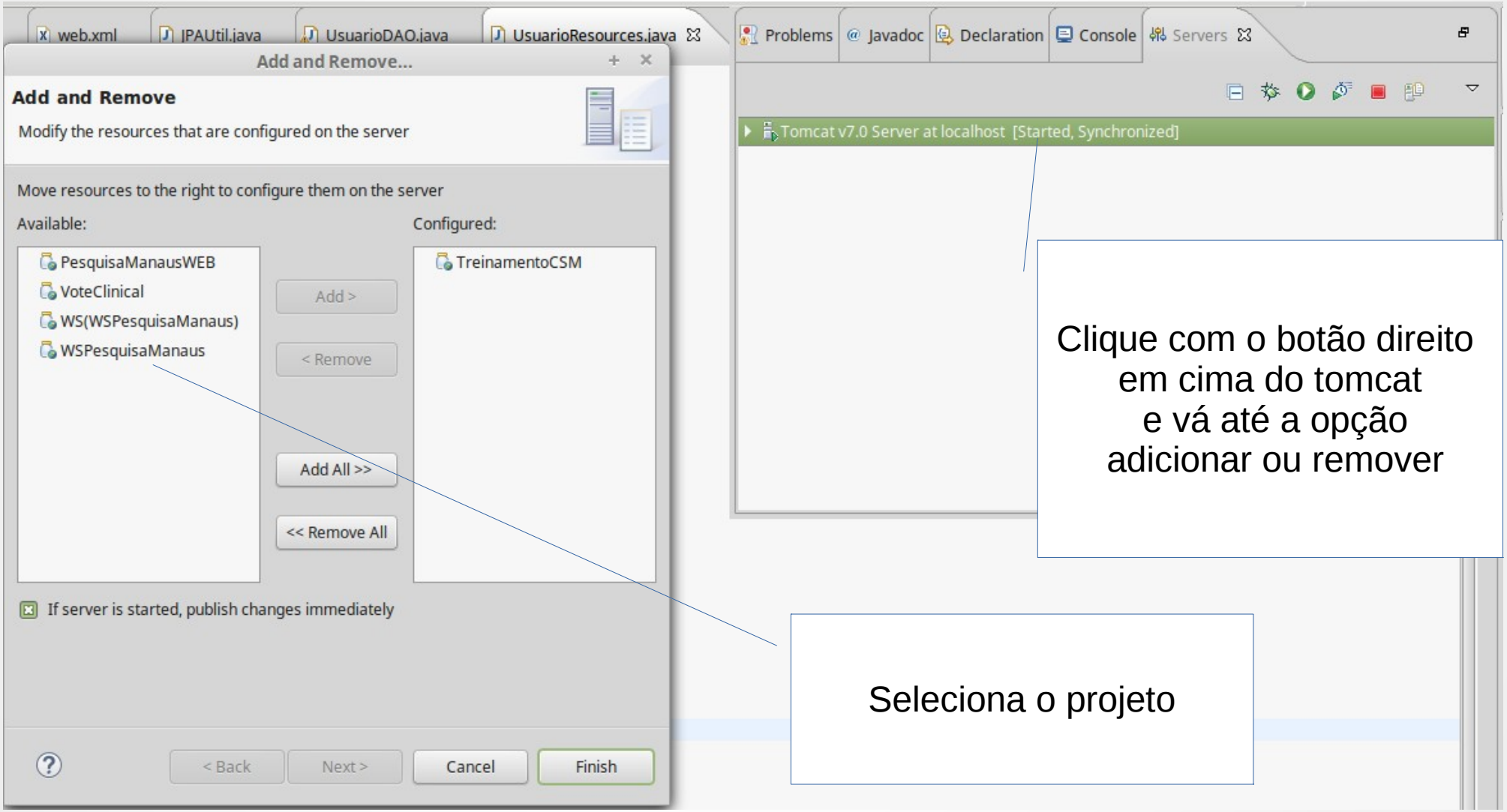
Agora vamos testar...

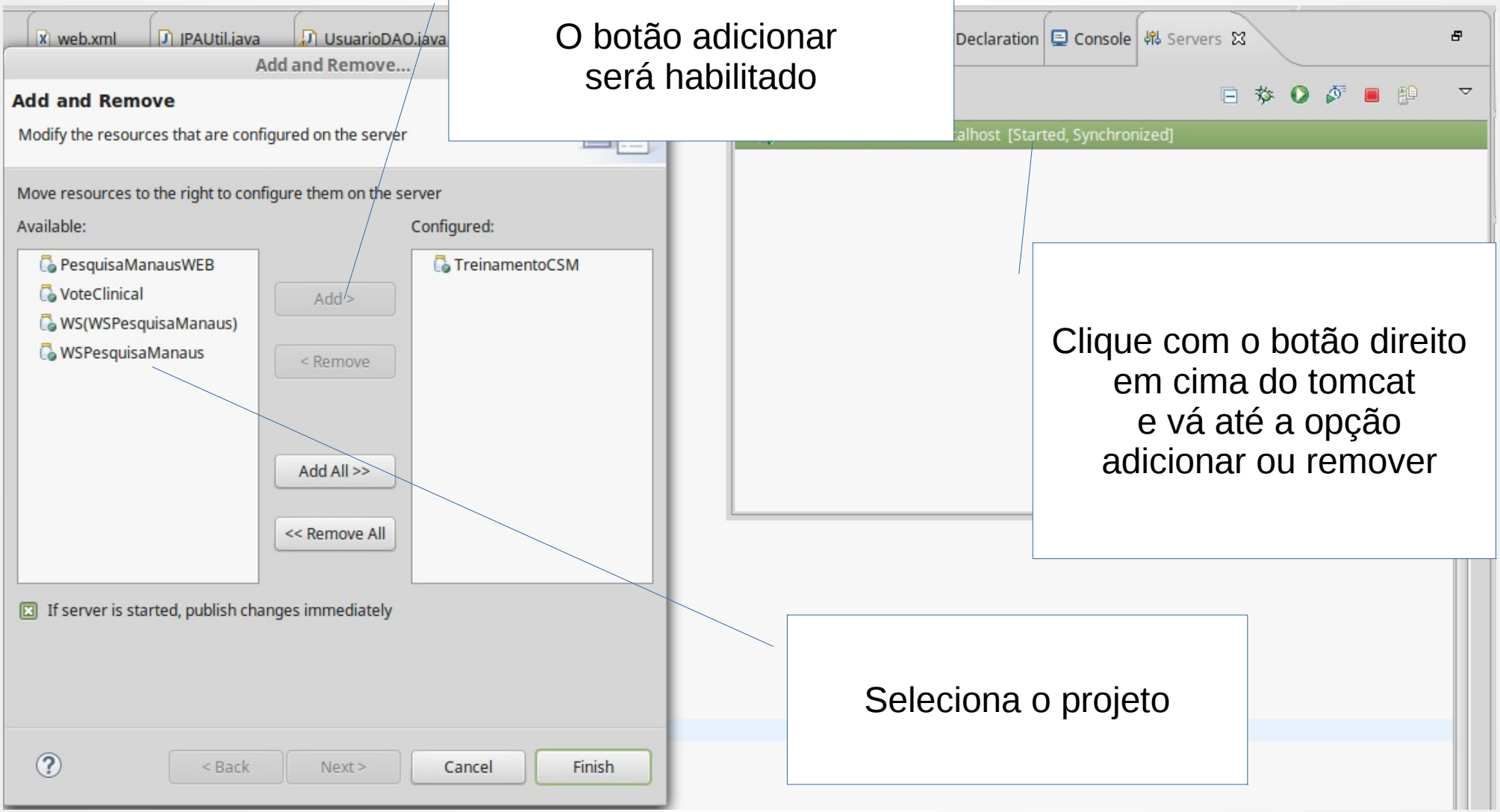
- Primeiro start seu mysql.
- Crie seu Banco de dados
- Após isso...

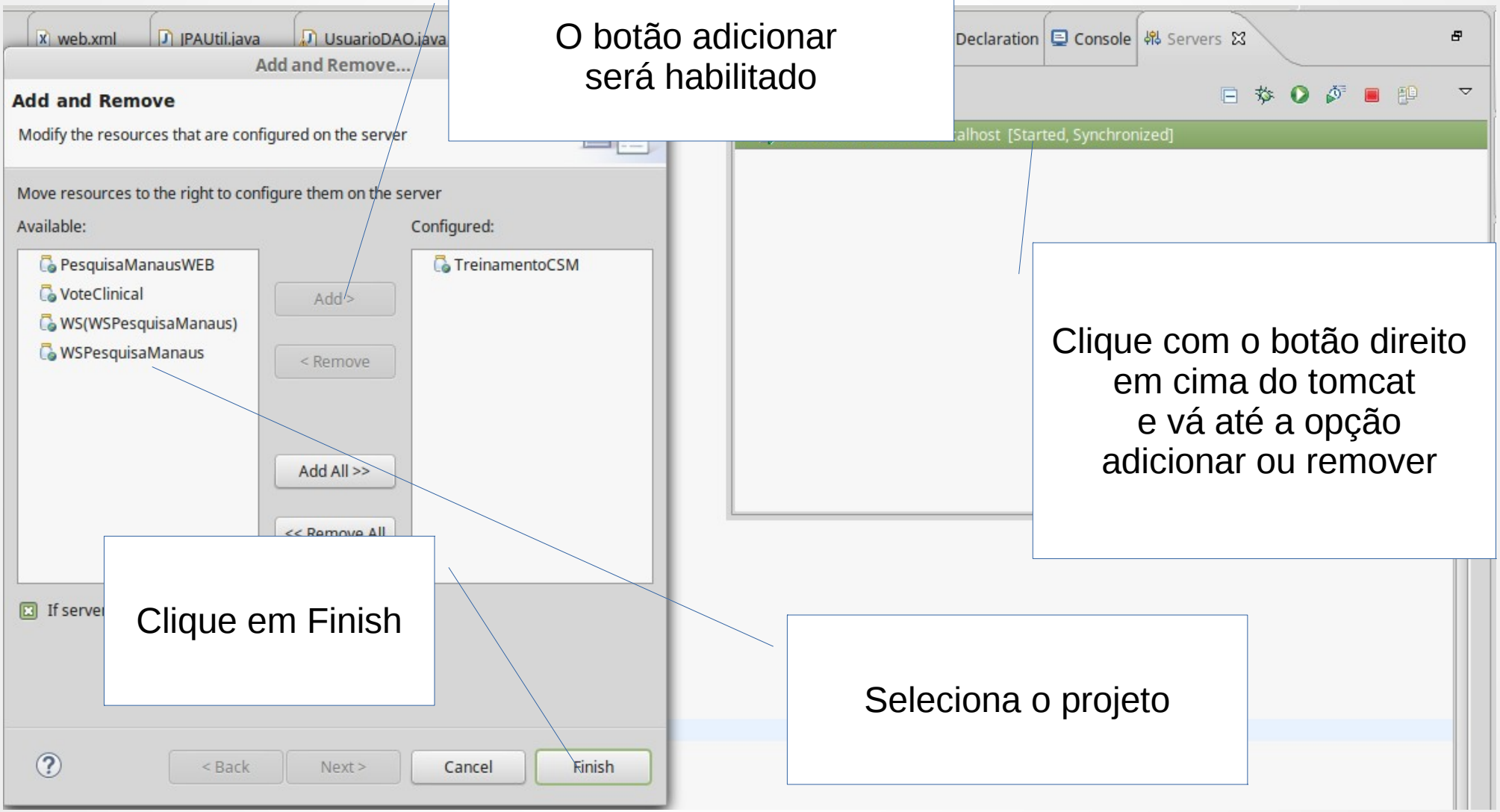

web.xml
IPAUtil.java
UsuarioDAO.java
UsuarioResources.java
Problems
Javadoc
Declaration
Console
Servers
Tomcat v7.0 Server at localhost [Started, Synchronized]
Add and Remove...
Add and Remove
Modify the resources that are configured on the server
Move resources to the right to configure them on the server
Available:
Configured:
PesquisaManausWEB
VoteClinical
WS(WSPesquisaManaus)
WSPesquisaManaus
TreinamentoCSM
Add >
< Remove
Add All >>
<< Remove All
If server is started, publish changes immediately
< Back
Next >
Cancel
Finish

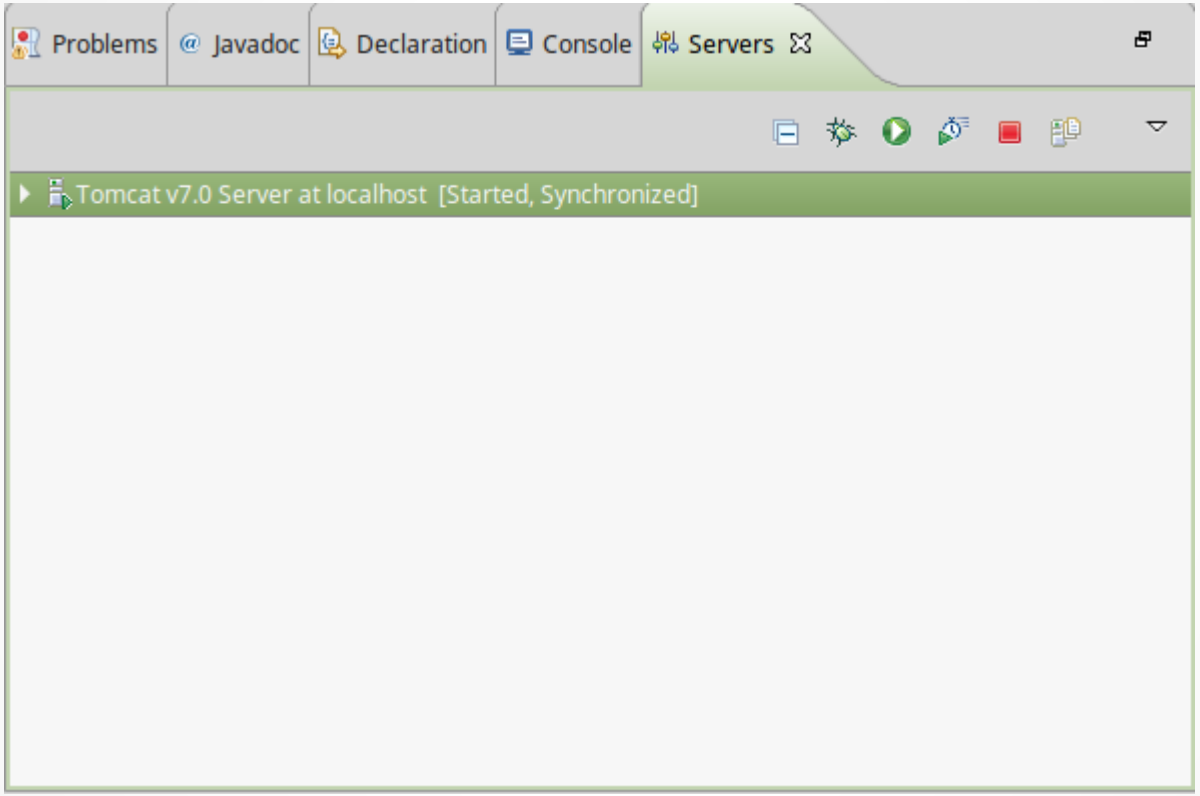


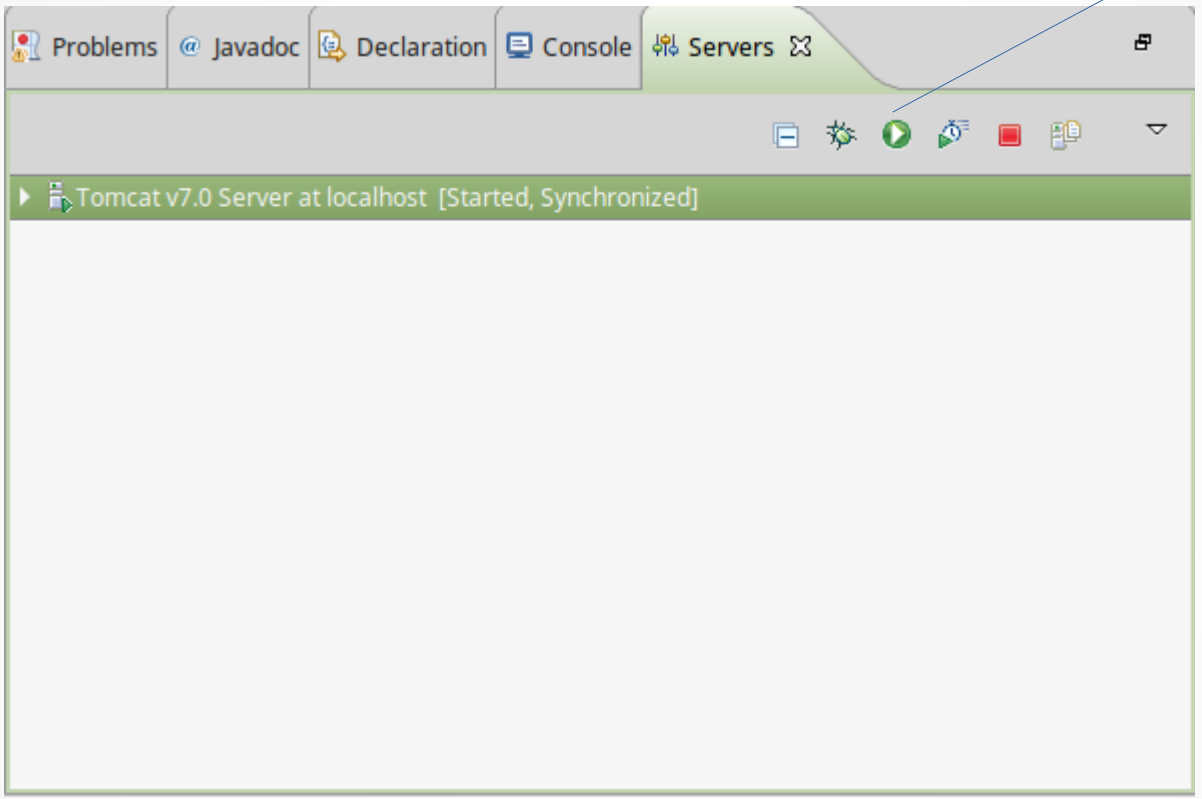
Clique com o botão direito em cima do tomcat e vá até a opção adicionar ou remover


web.xml
IPAUtil.java
UsuarioDAO.java
UsuarioResources.java
Problems
Javadoc
Declaration
Console
Servers
Tomcat v7.0 Server at localhost [Started, Synchronized]
Add and Remove...
Add and Remove
Modify the resources that are configured on the server
Move resources to the right to configure them on the server
Available:
Configured:
PesquisaManausWEB
VoteClinical
WS(WSPesquisaManaus)
WSPesquisaManaus
TreinamentoCSM
Add >
< Remove
Add All >>
<< Remove All
If server is started, publish changes immediately
< Back
Next >
Cancel
Finish
Clique com o botão direito em cima do tomcat e vá até a opção adicionar ou remover
Seleciona o projeto

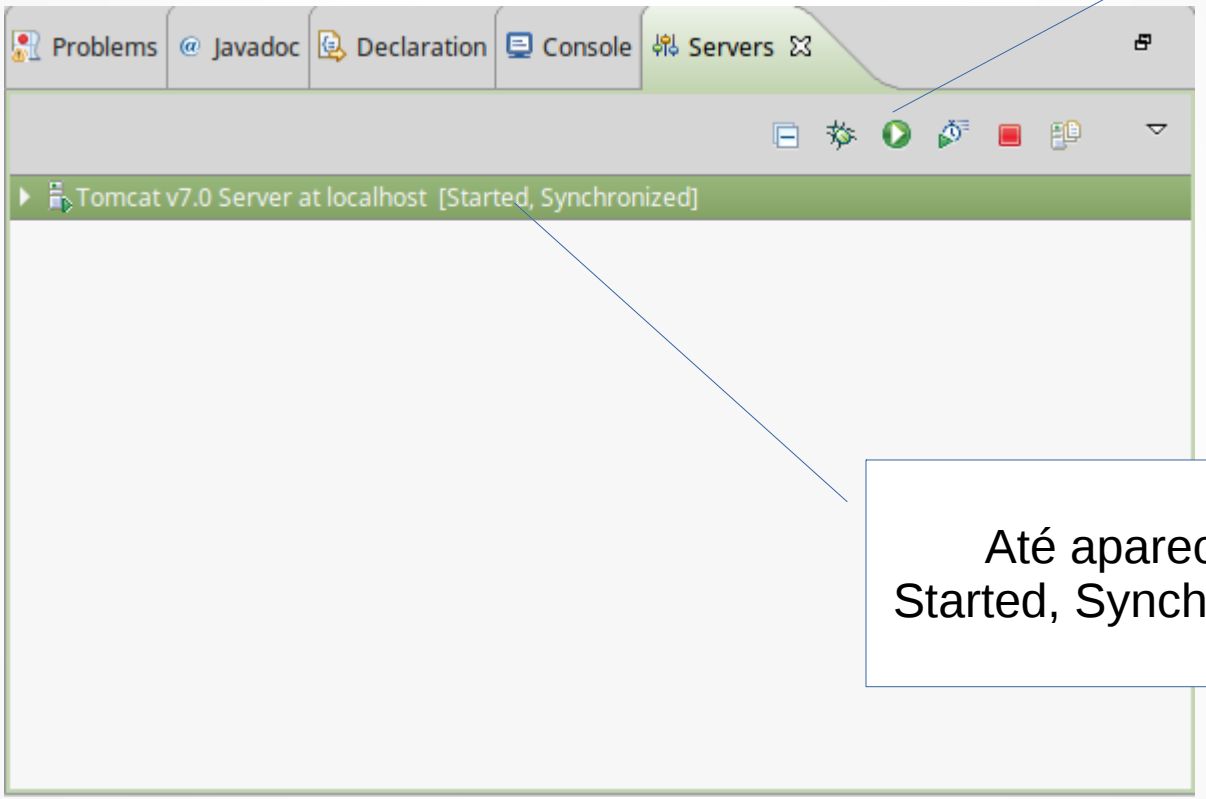

web.xml
IPAUtil.java
UsuarioDAO.java
Add and Remove...
Add and Remove
Modify the resources that are configured on the server
Move resources to the right to configure them on the server
Available:
Configured:
PesquisaManausWEB
VoteClinical
WS(WSPesquisaManaus)
WSPesquisaManaus
TreinamentoCSM
Add >
< Remove
Add All >>
<< Remove All
If server is started, publish changes immediately
< Back
Next >
Cancel
Finish
Declaration
Console
Servers
[Started, Synchronized]
O botão adicionar
será habilitado
Clique com o botão direito
em cima do tomcat
e vá até a opção
adicionar ou remover
Seleciona o projeto


web.xml
IPAUtil.java
UsuarioDAO.java
Add and Remove...
Add and Remove
Modify the resources that are configured on the server
Move resources to the right to configure them on the server
Available:
Configured:
PesquisaManausWEB
VoteClinical
WS(WSPesquisaManaus)
WSPesquisaManaus
TreinamentoCSM
Add >
< Remove
Add All >>
< Back
Next >
Cancel
Finish
Declaration
Console
Servers
[Started, Synchronized]
O botão adicionar será habilitado
Clique com o botão direito em cima do tomcat e vá até a opção adicionar ou remover
Clique em Finish
Seleciona o projeto


Problems
Javadoc
Declaration
Console
Servers
Tomcat v7.0 Server at localhost [Started, Synchronized]



Clique no play

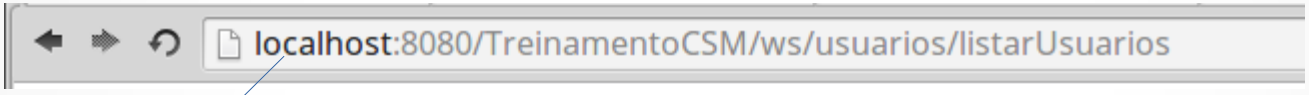


Clique no play

Até aparecer
Started, Synchronized

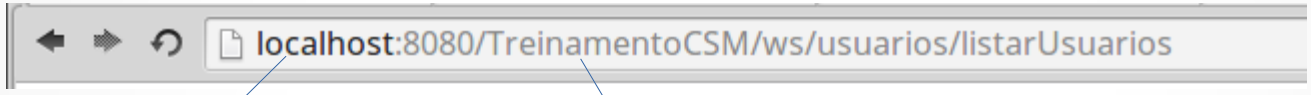
No browser do seu pc

No browser do seu pc



Localhost:8080, pois é porta usada pelo tomcat

No browser do seu pc

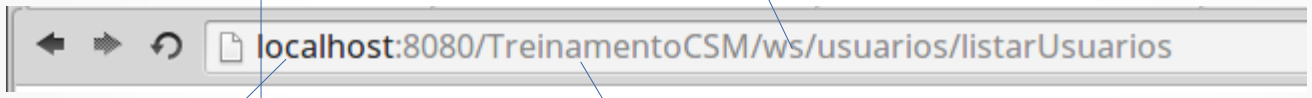


Localhost:8080, pois é porta usada pelo tomcat

Nome do Projeto

No browser do seu pc

Chamda da
Servlet que foi
informado no web.xml



Localhost:8080,
pois é
porta usada
pelo tomcat

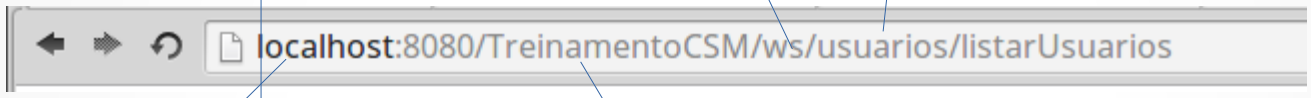
Nome do Projeto

```
<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

Chamda da Servlet que foi informado no web.xml

Path da classe UsuarioResource



Localhost:8080, pois é porta usada pelo tomcat

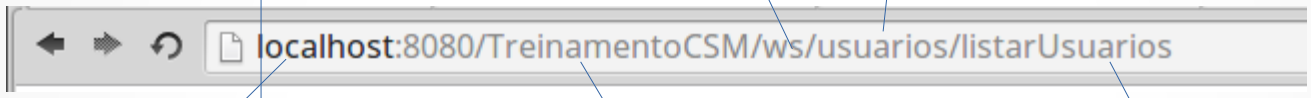
Nome do Projeto

```
<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

Chamda da Servlet que foi informado no web.xml

Path da classe UsuarioResource



Localhost:8080, pois é porta usada pelo tomcat

Nome do Projeto

Método da classe UsuarioResources

```
<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 Ãbia 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