

MANUAL DE INTEGRACION DE UN PROYECTO EN MAVEN Y CAS JASIG

REQUISITOS:

CAS JASIG 3.5.2 descargado de la siguiente pagina

http://www.jasig.org/cas_server_3_5_2_release

NETBEANS 8.0

<https://netbeans.org/downloads/>

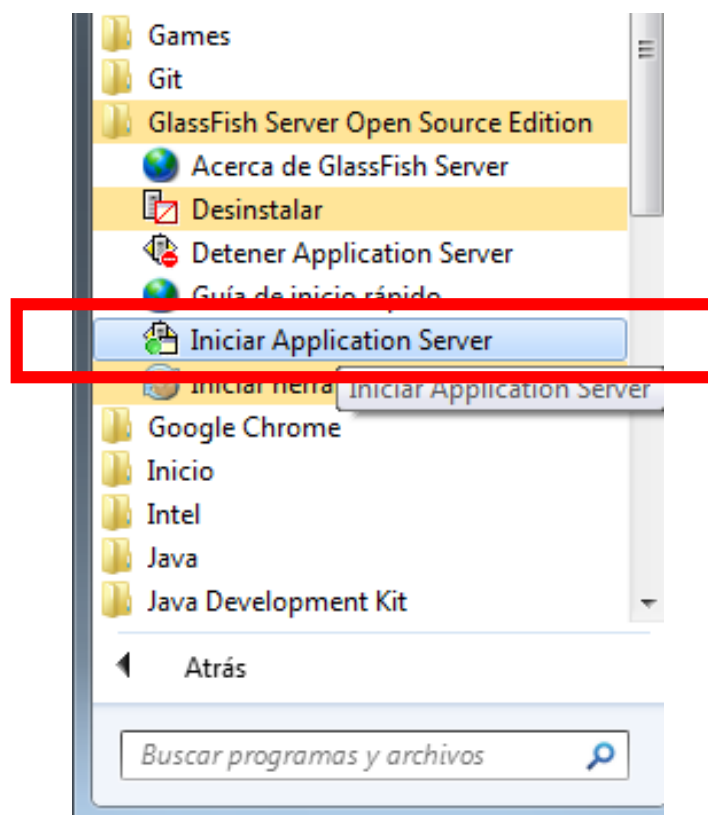
GLASSFISH 4.0

<https://glassfish.java.net/download.html>

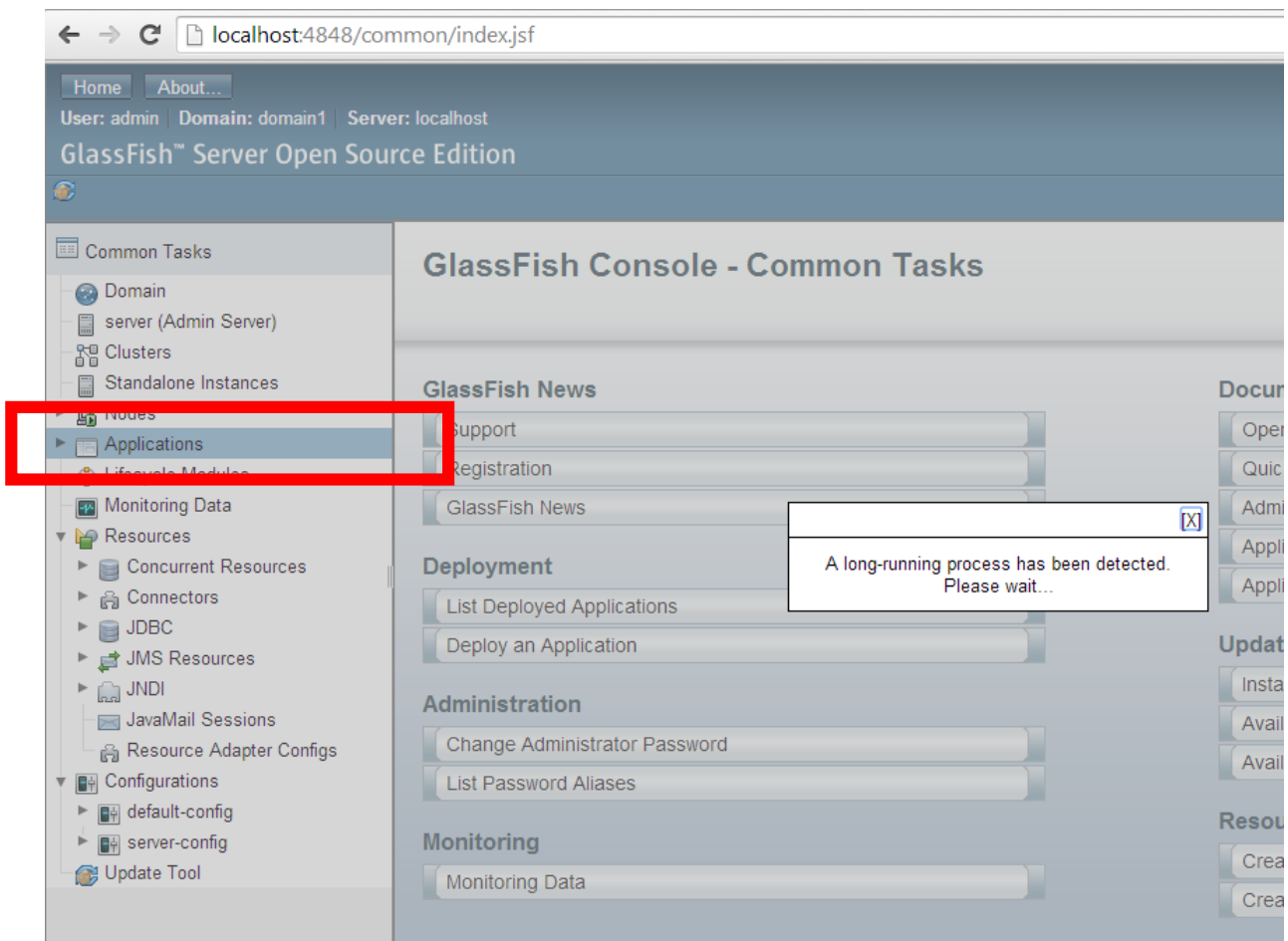
POSTGRESQL

<http://www.postgresql.org/download/>

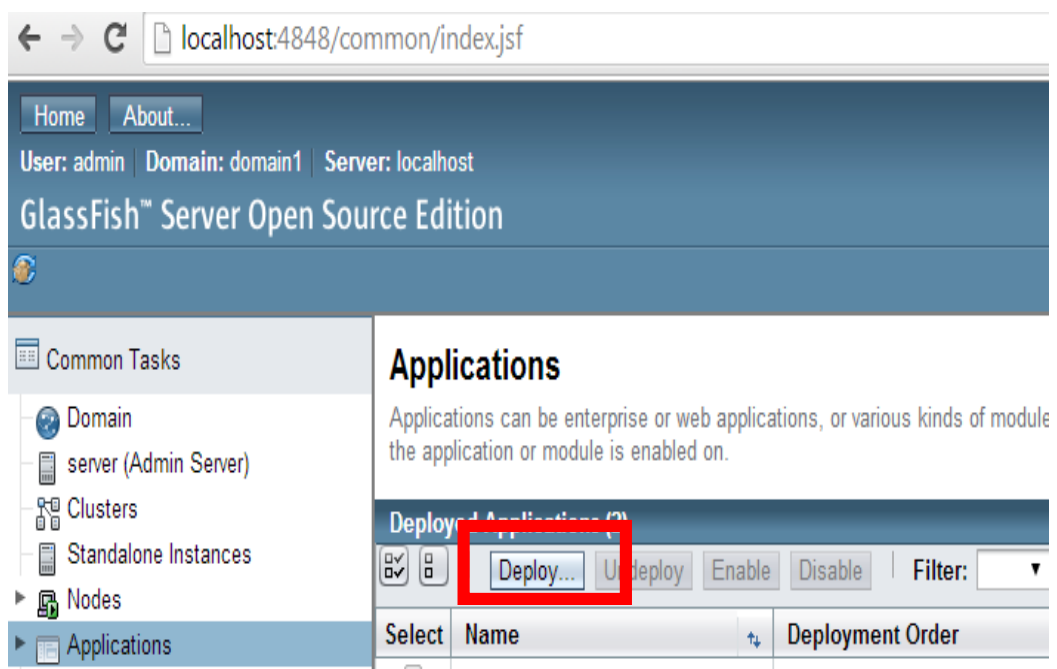
1. Desplegar Cas Jasig en nuestro Servidor
 - Primero se irá a desplegar el .war del proyecto *Cas Jasig* en el *glassfish* para ello se inicia el servidor.



- Se puede ir a <http://localhost:8080/> o directamente a <http://localhost:4848/> -> se escoge la opción Applications



- Se escoge *deploy*



- Se seleccionará el archivo en este caso el .war del Cas Jasig

← → ↻ localhost:4848/common/index.jsf

Home About...

User: admin | Domain: domain1 | Server: localhost

GlassFish™ Server Open Source Edition

Common Tasks

- Domain
- server (Admin Server)
- Clusters
- Standalone Instances
- Nodes
- Applications
- Lifecycle Modules
- Monitoring Data
- Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - IMS Resources

Deploy Applications or Modules

Specify the location of the application or module to deploy. An application can be in a packaged file or specified as a directory.

Location: ☒ Packaged File to Be Uploaded to the Server

Ningún archivo seleccionado

☐ Local Packaged File or Directory That Is Accessible from GlassFish Server

Type: *

- Buscas el .war de la aplicación de Cas Jasig generalmente se encuentra en la siguiente ruta :

...cas-server-3.5.2-release\cas-server-3.5.2\modules\cas-server-webapp-3.5.2.war [puede cambiar el nombre del .war si lo desea]

- Luego da clic en OK

Deploy Applications or Modules

Specify the location of the application or module to deploy. An application can be in a packaged file or specified as a directory.

Location: ☒ Packaged File to Be Uploaded to the Server

cas-server-w...p-3.5.2.war

☐ Local Packaged File or Directory That Is Accessible from GlassFish Server

Type: *

* Indicates required field

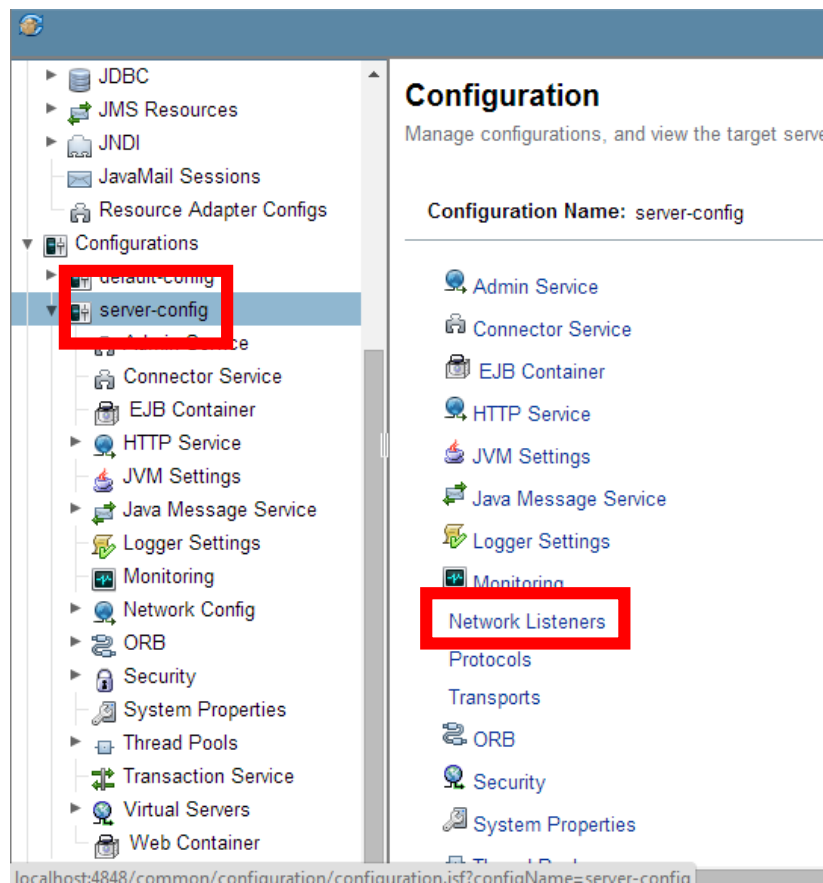
- Como resultado se tendrá la siguiente imagen

Applications

Applications can be enterprise or web applications, or various kinds of modules. Restart an application or module by clicking on the reload link, this action will apply only to the targets that the application or module is enabled on.

Deployed Applications (2)						
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Deploy...	Undeploy	Enable	Disable	Filter: <input type="text"/>
Select	Name	Deployment Order	Enabled	Engines	Action	
<input type="checkbox"/>	MavenPet-web	100	✓	web	Launch Redeploy Reload	
<input type="checkbox"/>	cas-server-webapp-3.5.2	100	✓	web	Launch Redeploy Reload	

- Para el uso del Cas se necesita una conexión segura se proseguirá a cambiar el puerto
 - Se elegirá *Server-Config*; Se escoge *Network Listeners*



- Luego *http-listener-2*



- Se cambia el *puerto a 8443* y se procede a guarda

General	SSL	HTTP	File Cache
---------	-----	------	------------

Edit Network Listener

Modify an existing network listener.

[Load Defaults](#)

Configuration Name: server-config

Name: http-listener-2

Protocol: http-listener-2

Status: ☒ Enabled

Security: ☒ Enabled

JK Listener: ☐ Enabled

If selected, listener is an Apache mod_jk listener

Port: *

- En la pestaña *SSL* se habilita la *casilla SSL3*

General	SSL	HTTP	File Cache
---------	-----	------	------------

SSL

Modify SSL settings.

Configuration Name: server-config

SSL: ☒ Enabled

TLS: ☒ Enabled

Client Authentication: ☐ Enabled
Requires the client to authenticate itself to the server.

Certificate NickName: *
Takes a single value, identifies the server's keypair and certificate.

Key Store:
Name of the keystore file (for example, keystore.jks)

Trust Algorithm:
Name of the trust management algorithm (for example, PKIX) to use

Max Certificate Length:
Maximum number of non-self-issued intermediate certificates that c

- Luego de seguir los pasos anteriores, se dirige de nuevo a *Applications* y se da clic en launch; saldrán dos links, se elegirán el segundo link que tiene ya conexión segura que al cambiar el puerto glassfish genera un certificado por defecto con nuestro usuario de administrador



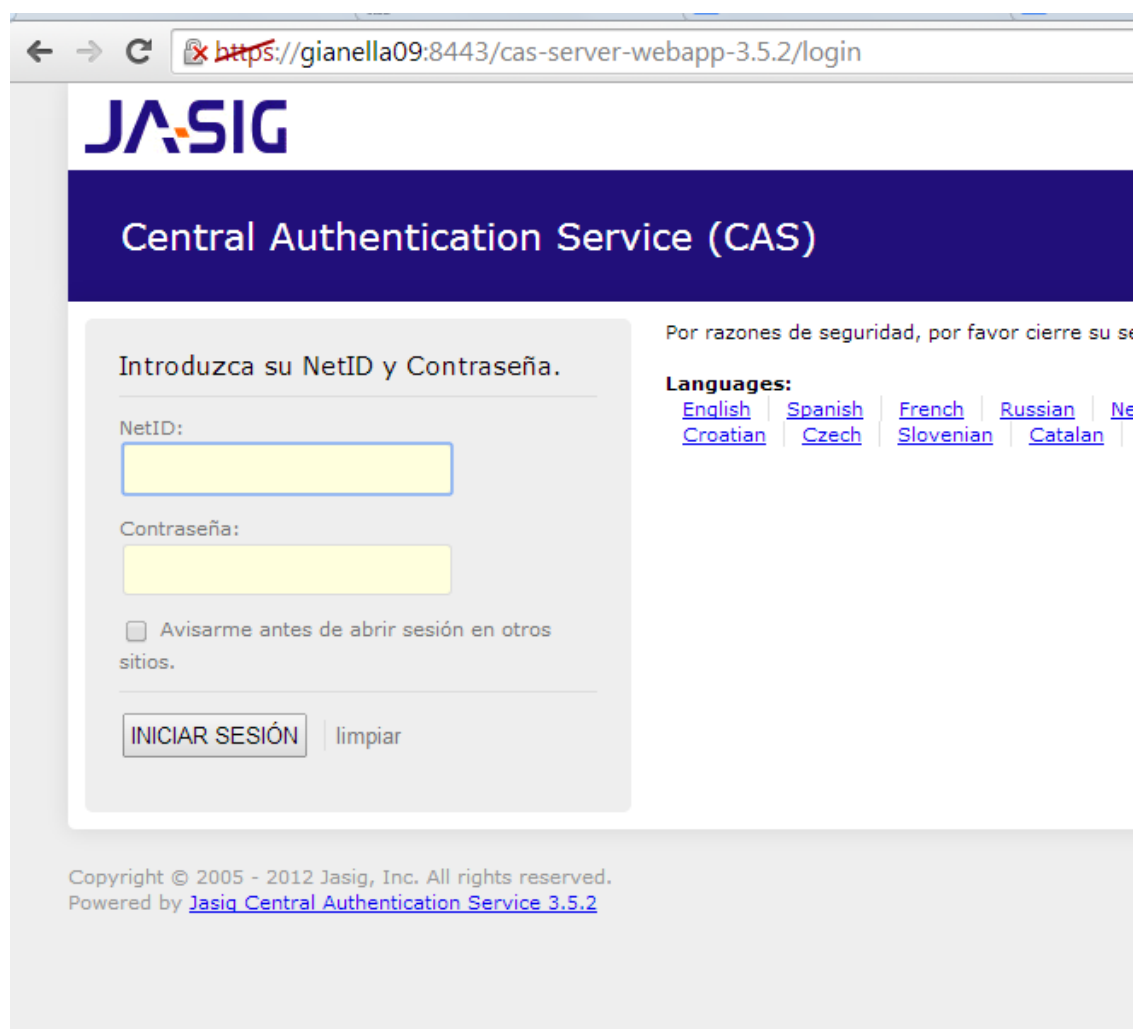
Web Application Links

If the server or listener is not running, the link may not work. In this event, check the status of the se

Application Name: cas-server-webapp-3.5.2

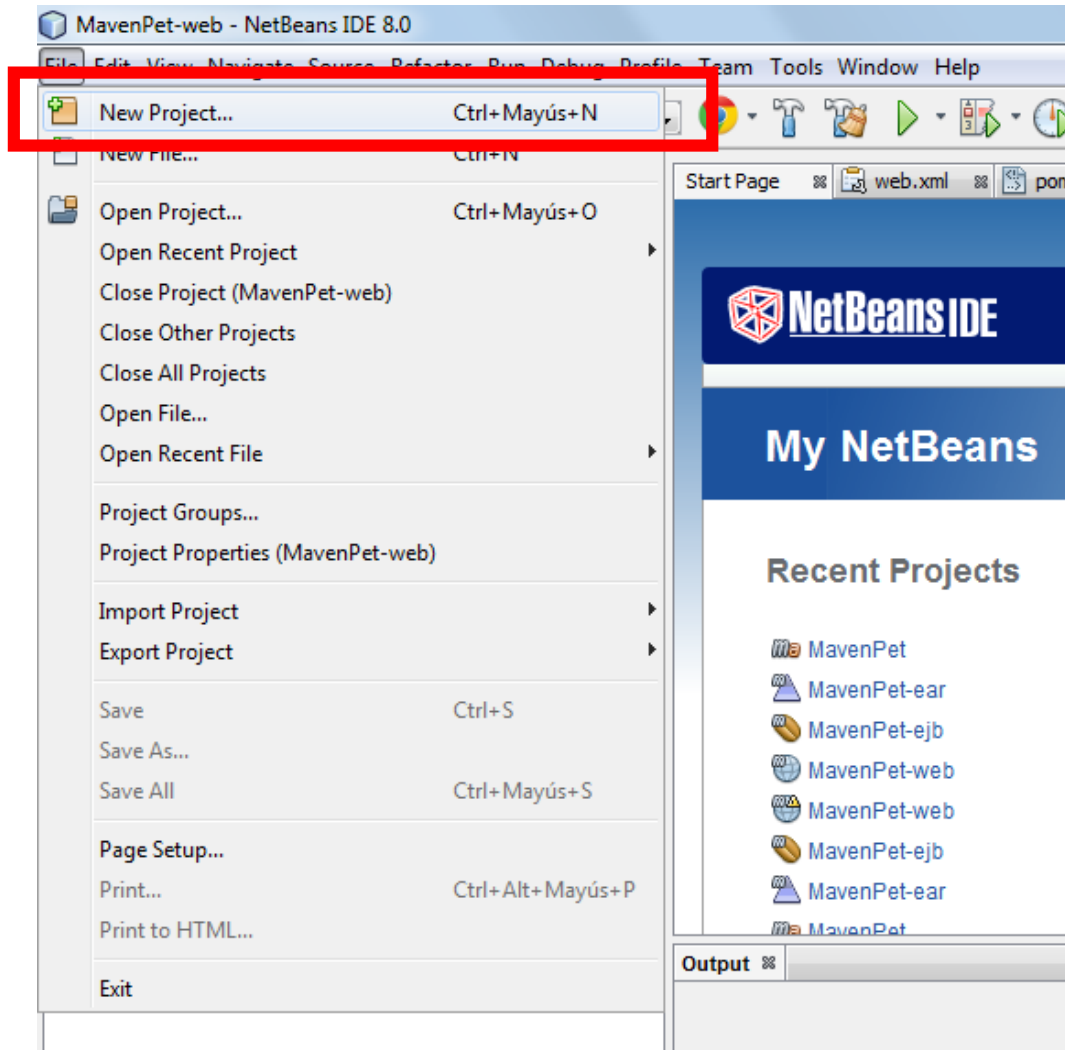
Links:
[server] <http://gianella09:8080/cas-server-webapp-3.5.2>
[server] <https://gianella09:8443/cas-server-webapp-3.5.2>

- Obteniendo la siguiente página

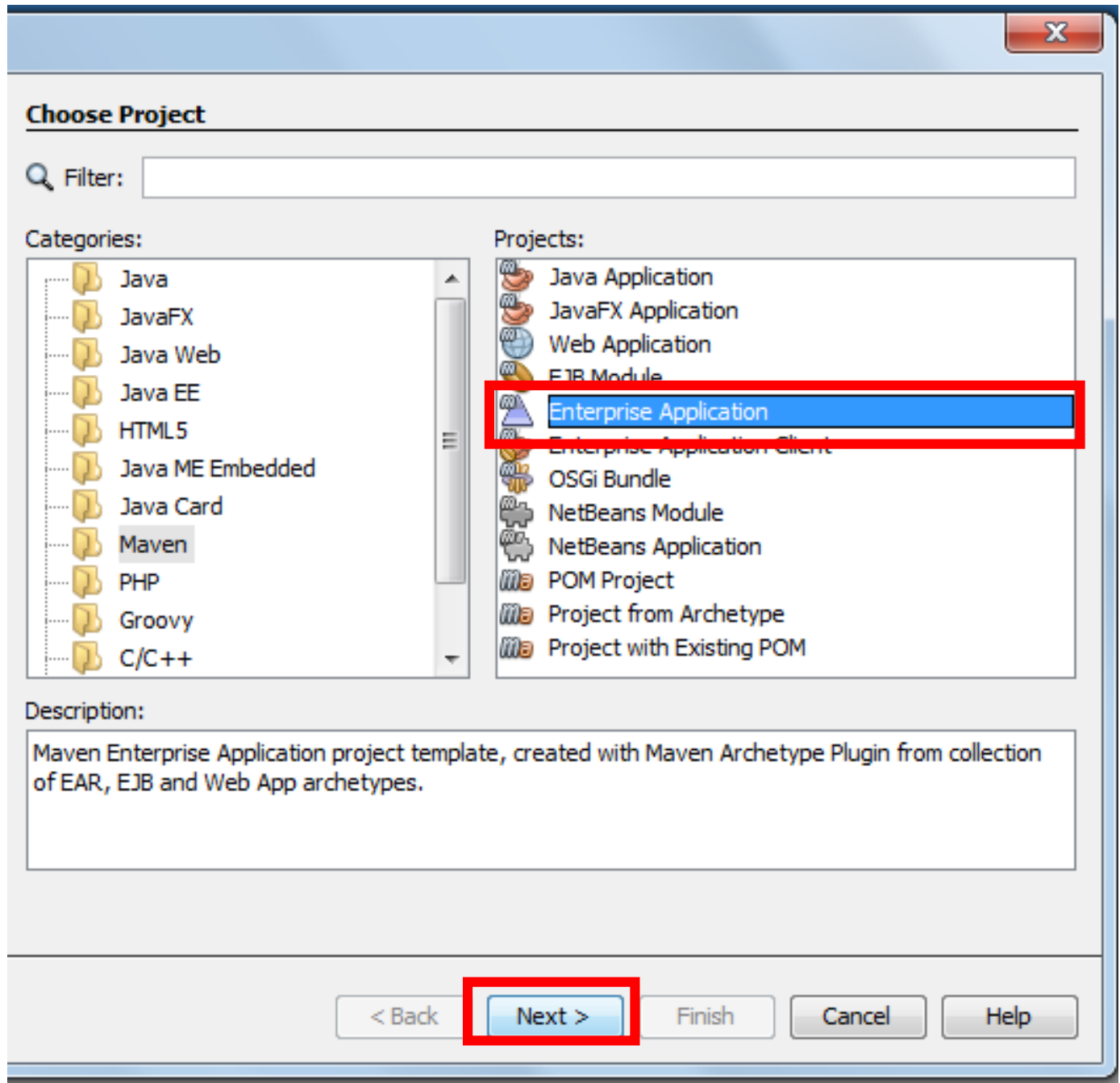


INTEGRACION DEL CAS JASIG A UN PROYECTO CREADO EN NEBEANS CON MAVEN

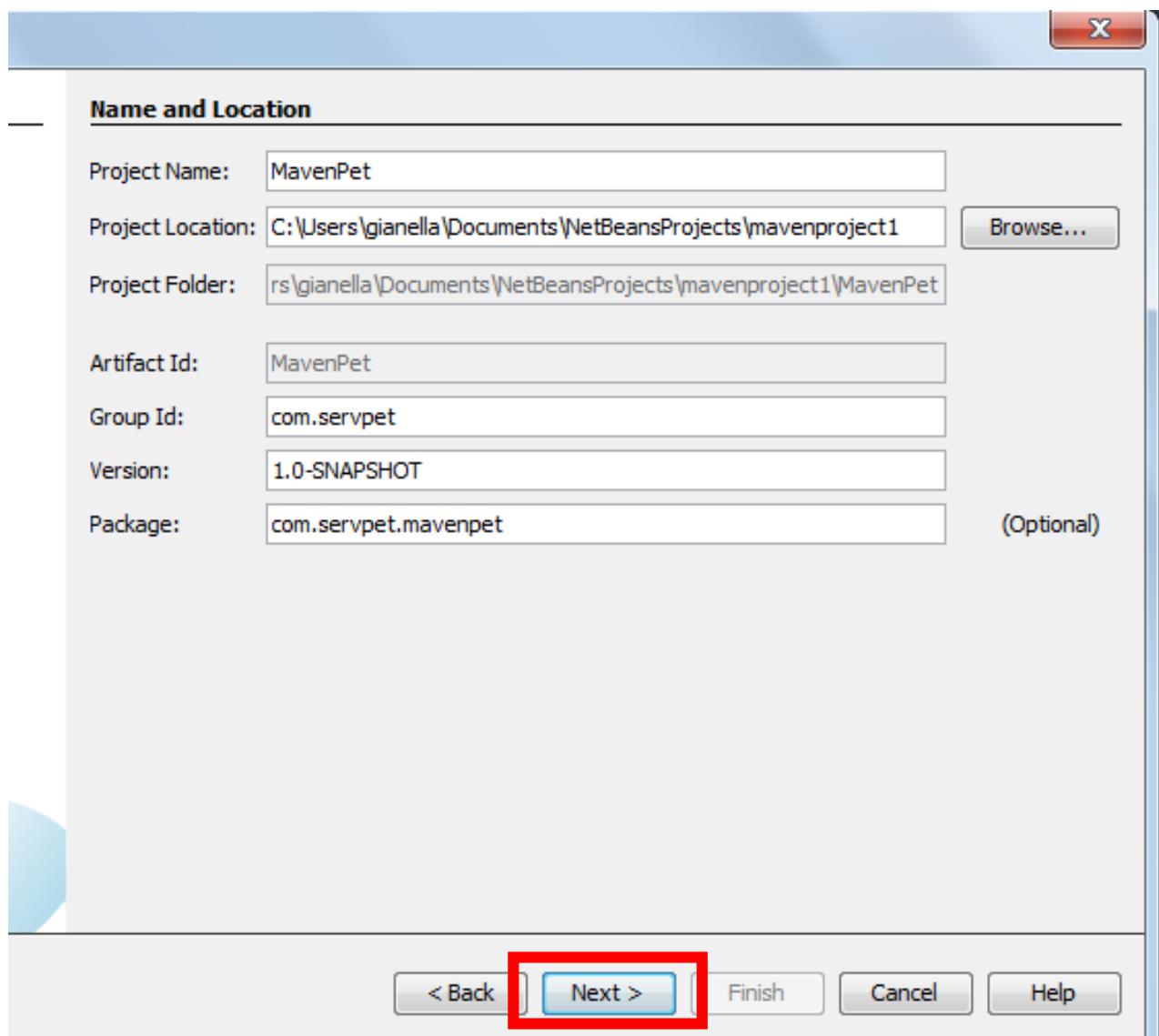
1. Se creará un proyecto en MAVEN en nuestro NETBEANS 8.0
 - Click en *File -> New Project*



- Se escoge *Maven -> Enterprise Application*



- Se escribe los datos correspondientes



The image shows a 'Name and Location' dialog box from NetBeans. It contains several text input fields for project configuration. The 'Project Name' field is filled with 'MavenPet'. The 'Project Location' field is filled with 'C:\Users\gianella\Documents\NetBeansProjects\mavenproject1', and a 'Browse...' button is to its right. The 'Project Folder' field is filled with 'rs\gianella\Documents\NetBeansProjects\mavenproject1\MavenPet'. Below these are fields for 'Artifact Id' (MavenPet), 'Group Id' (com.servpet), 'Version' (1.0-SNAPSHOT), and 'Package' (com.servpet.mavenpet), with a '(Optional)' label to the right of the Package field. At the bottom, there are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'. The 'Next >' button is highlighted with a red rectangular box.

Name and Location

Project Name:

Project Location:

Project Folder:

Artifact Id:

Group Id:

Version:

Package: (Optional)

< Back **Next >** Finish Cancel Help

- Se elige el servidor en nuestro caso utilizamos Glassfish 4.0 y con el JDK 7 debido a que se utilizara Primefaces y por ultimo *Finish*

Settings

Server: GlassFish Server 4.0 Add...

Java EE Version: Java EE 7

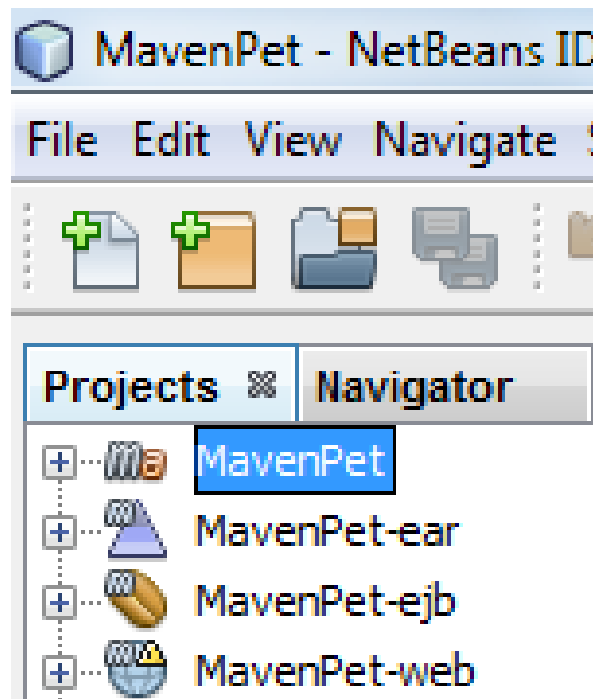
☒ Create EJB Module: MavenPet-ejb

☒ Create Web App Module: MavenPet-web

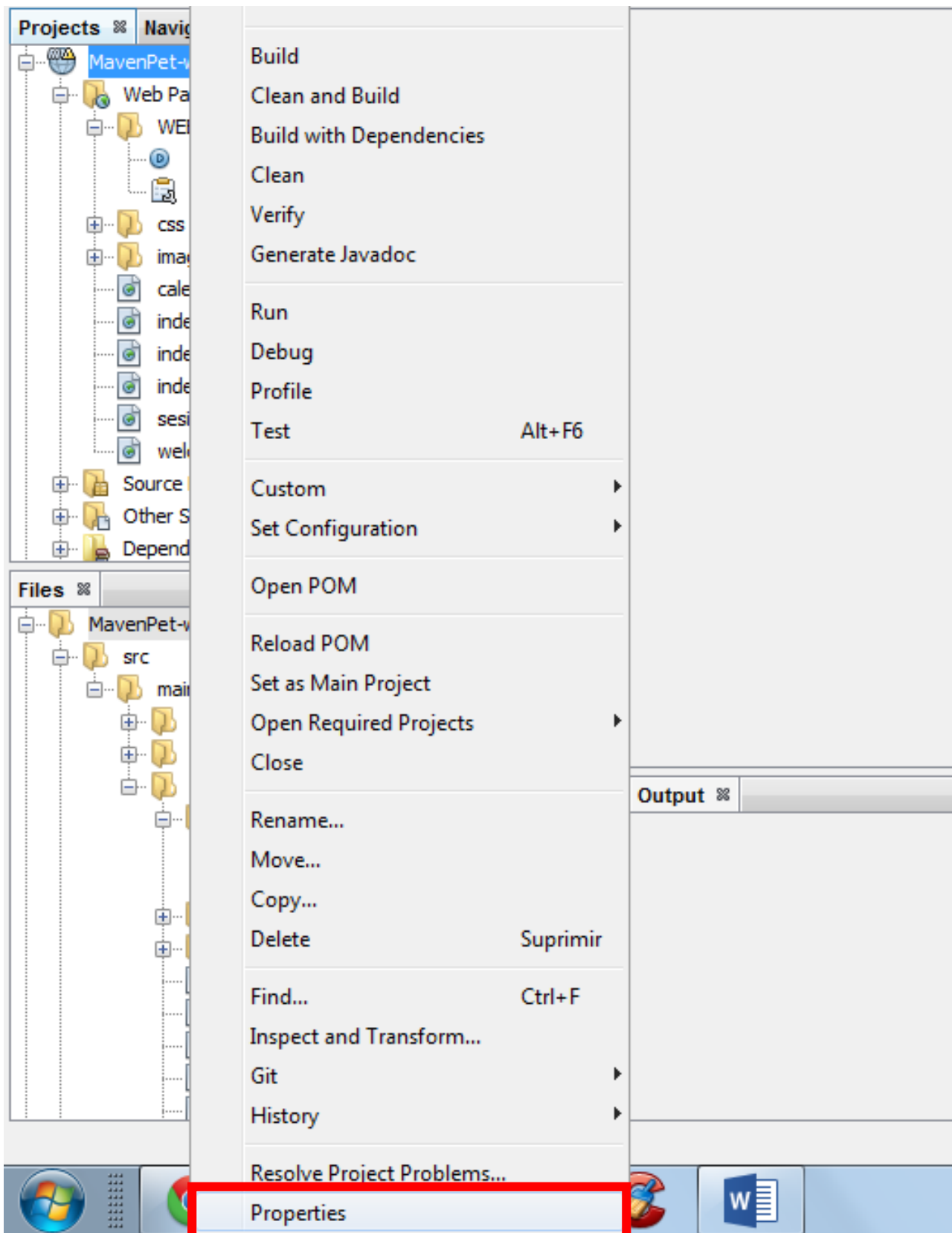
Assembly module: MavenPet-ear

< Back Next > **Finish** Cancel Help

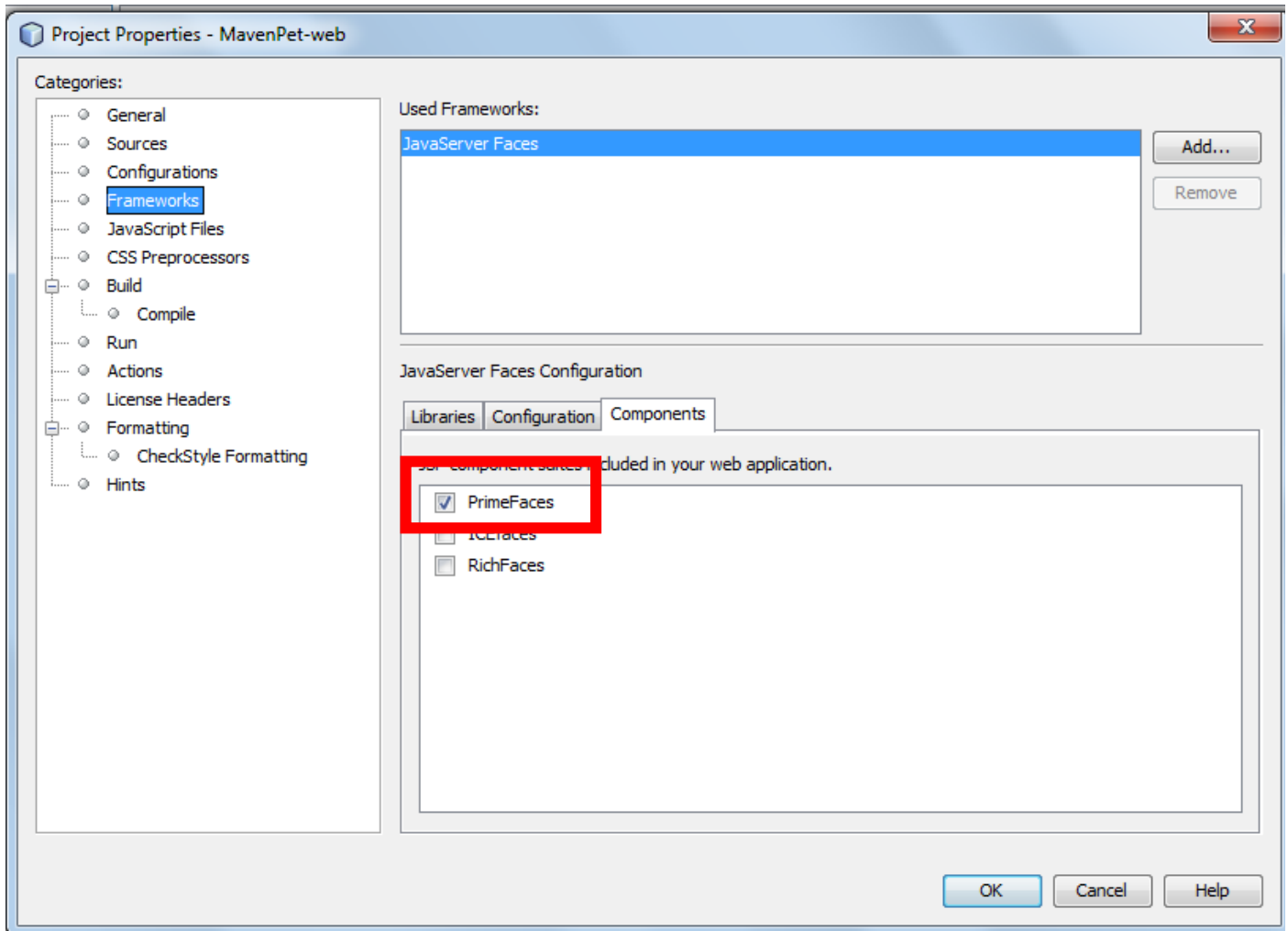
2. Se debe tener el siguiente esquema



3. Para trabajar con el componente *primefaces* se debe activarlo, para ello se realizara la siguiente configuración:
- Clic Derecho en MavenPet-web -> Propiedades



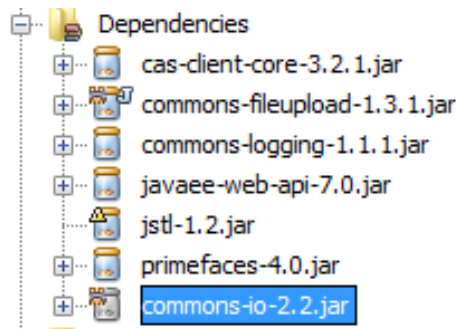
- Frameworks - > Components [*Se da un visto a Primefaces*] -> OK



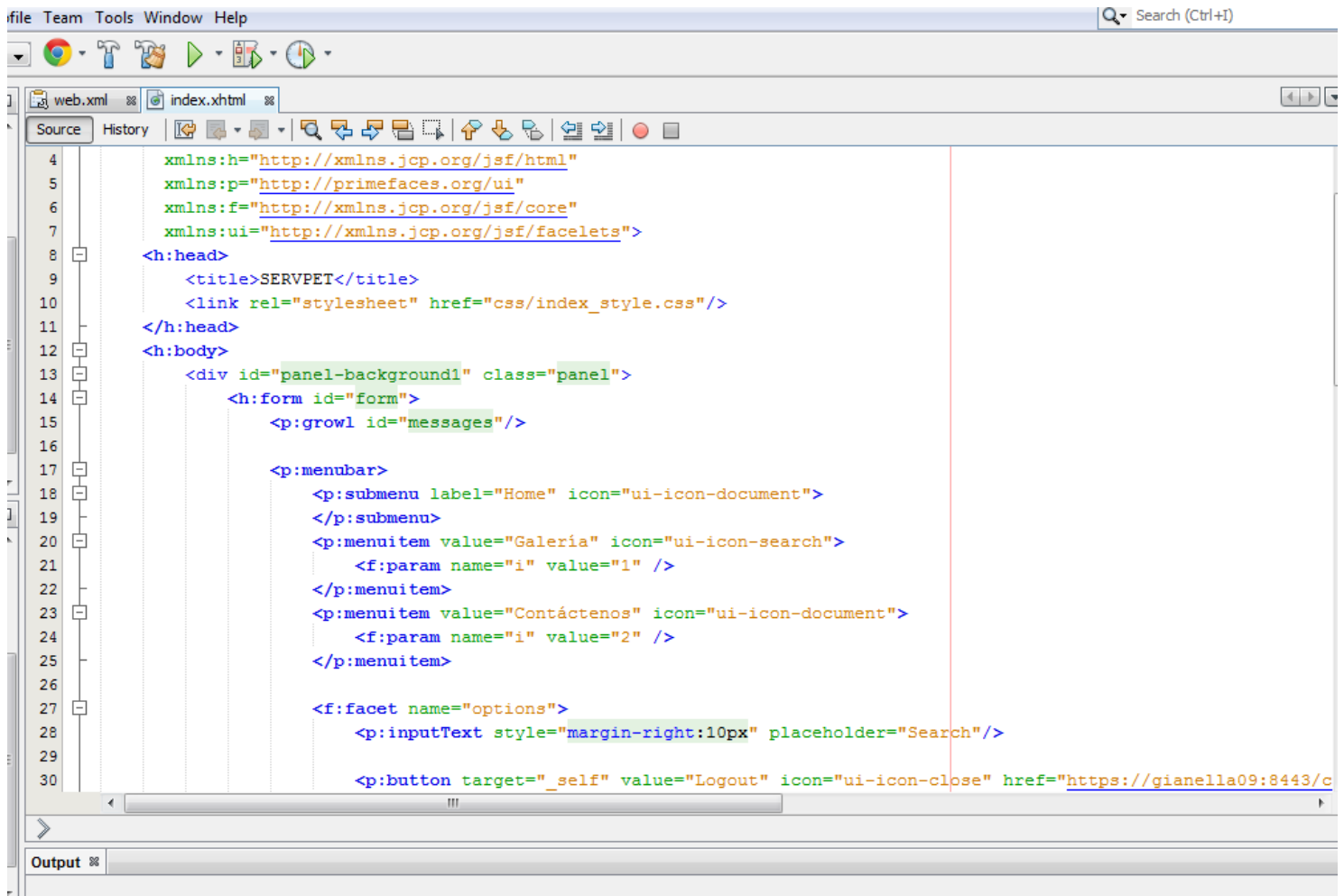
SE DEBERÁ TENER LOS SIGUIENTES JAR

Estos .jar deberán estar ubicados en la carpeta Dependencias del MavenPet-web

org.jasig.cas.client.jar
commons-fileupload-1.3.1.jar
commons-logging-1.1.1.jar
primefaces-4.0.jar
commons-io-2.2.jar



4. Se creará una pequeña página para la integración del cas y nuestra aplicación



5. Para que nuestra aplicación pueda usar del Cas se hará la siguiente configuración agregando en el web.xml las siguientes líneas en el mismo orden

```
<filter>
  <filter-name>CAS Authentication Filter</filter-name>
  <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>
  <init-param>
    <param-name>casServerLoginUrl</param-name>
    <param-value>https://gianella09:8443/cas-server-webapp-3.5.2/login</param-
value>
  </init-param>
  <init-param>
    <param-name>serverName</param-name>
    <param-value>https://gianella09:8443</param-value>
  </init-param>
</filter>

<filter>
  <filter-name>CAS Validation Filter</filter-name>
  <filter-class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-class>
  <init-param>
    <param-name>casServerUrlPrefix</param-name>
    <param-value>https://gianella09:8443/cas-server-webapp-3.5.2</param-value>
  </init-param>
  <init-param>
    <param-name>serverName</param-name>
    <param-value>https://gianella09:8443</param-value>
  </init-param>
</filter>

<filter>
  <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
  <filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>
</filter>
<filter>
  <filter-name>CAS Single Sign Out Filter</filter-name>
  <filter-class>org.jasig.cas.client.session.SingleSignOutFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>CAS Authentication Filter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

<filter-mapping>
  <filter-name>CAS Validation Filter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

<filter-mapping>
  <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

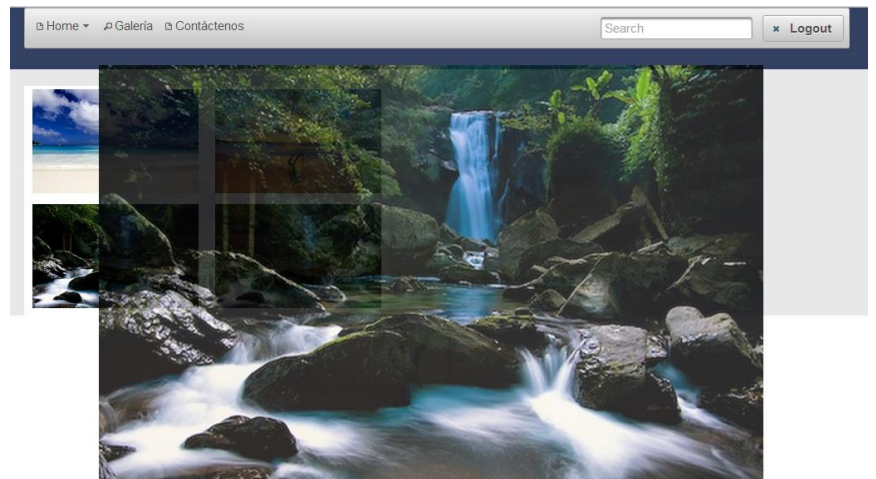
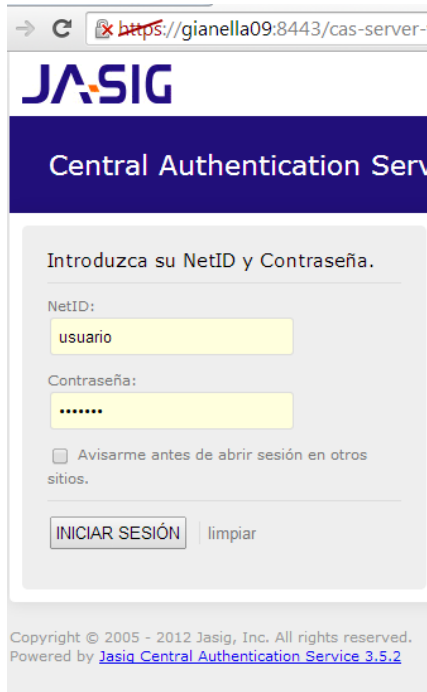
```
<filter-mapping>
  <filter-name>CAS Single Sign Out Filter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

<listener>
  <listener-class>
    org.jasig.cas.client.session.SingleSignOutHttpSessionListener
  </listener-class>
</listener>
```

6. Luego de agregar esas líneas podrá correr su proyecto y se abrirá directamente la página del Cas que para ingresar a la página deberá ingresar como:

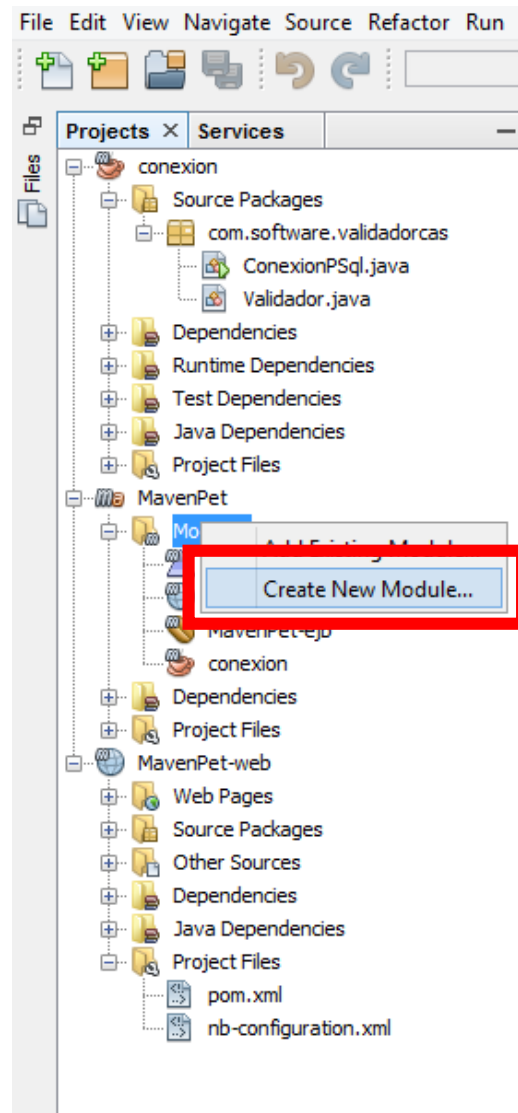
Usuario: usuario
Password: usuario

[Palabras indenticas tanto en usuarios como en password]

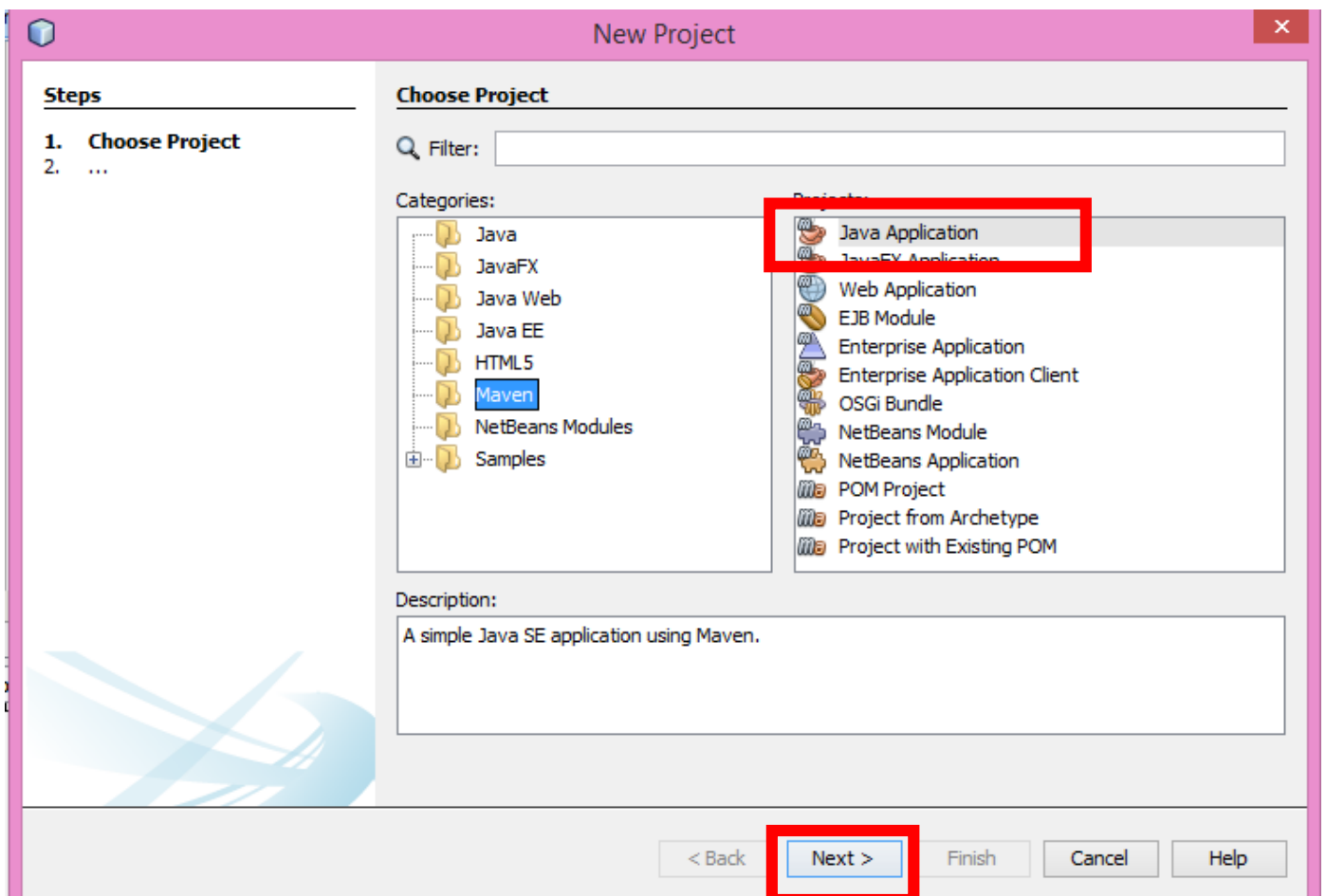


INTEGRANDO EL CAS CON NUESTRA BASE DE DATOS POSTGRESQL

1. Se crea una Java Application para nuestras clases quien se encargara de validar y crear la conexión a nuestra base de datos PostgreSQL
 - Clic Derecho en *Modulos* -> *Create New Modulo*



- Maven -> Java Application



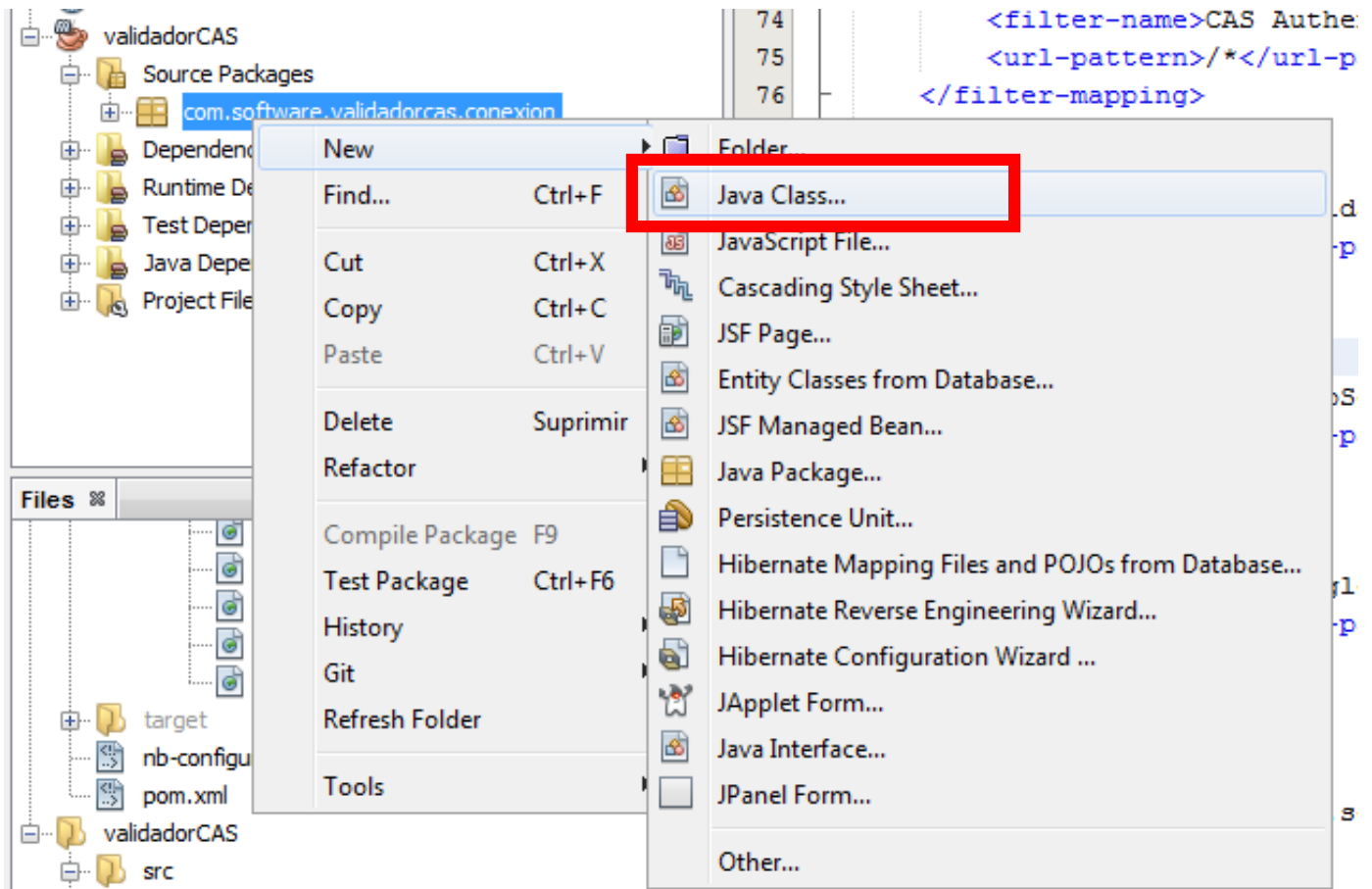
- Agrega los datos correspondientes [Nombre del Proyecto: validadorCAS]

Name and Location

Project Name:	<input type="text" value="validadorCAS"/>	
Project Location:	<input type="text" value="ments\ESPOL\2014-I TERMINO\SOFTWARE II\MavenPet\MavenPet"/>	<input type="button" value="Browse..."/>
Project Folder:	<input type="text" value="2014-I TERMINO\SOFTWARE II\MavenPet\MavenPet\validadorCAS"/>	
Artifact Id:	<input type="text" value="validadorCAS"/>	
Group Id:	<input type="text" value="com.servpet"/>	
Version:	<input type="text" value="1.0-SNAPSHOT"/>	
Package:	<input type="text" value="com.servpet.validadorcas.conexion"/>	(Optional)

< Back Next > **Finish** Cancel Help

2. Se crea 2 clases .java



3. En la primera clase con nombre : ConexionPSQL se debe copiar el siguiente código

```
package com.software.validadorcas.conexion;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;

public class ConexionPsql {
    private Connection connection = null;
    private final String url = "jdbc:postgresql://";
    private final String serverName = "localhost";
    private final String portNumber = "5432";
    private final String databaseName = "veterinaria";
    private final String userName = "postgres";
    private final String password = "gianella";

    private final String selectMethod = "Direct";
```

```
// Constructor
public ConexionPsql() {
}

private String getConnectionUrl() {
    String conexionSQL = url + serverName + ":" + portNumber +
";databaseName="+ databaseName ;
    String conexionPostgrasql =url + serverName + ":" + portNumber +
"/"+databaseName ;

    return conexionPostgrasql;
}

public java.sql.Connection getConnection() {
    try {
        Class.forName("org.postgresql.Driver");
        connection = java.sql.DriverManager.getConnection(getConnectionUrl(),
            userName, password);
        if (connection != null)
            System.out.println("Connection Successful!");
    } catch (Exception e) {
        e.printStackTrace();
        System.out.println("Error Trace in getConnection() : "
            + e.getMessage());
    }
    return connection;
}
}
```

4. En la segunda clase con nombre : Validador se debe copiar el siguiente código

```
package com.software.validadorcas.conexion;

import java.sql.ResultSet;
import java.sql.Statement;
import org.jasig.cas.authentication.handler.support.AbstractUsernamePasswordAuthenticationHandler;
import org.jasig.cas.authentication.principal.UsernamePasswordCredentials;

public class Validador extends AbstractUsernamePasswordAuthenticationHandler {

    public boolean authenticateUsernamePasswordInternal(UsernamePasswordCredentials
credentials)
    {
        String username = credentials.getUsername();
        String password = credentials.getPassword();
        boolean valid = false;
        String statementPostgrasql = "select * from \"usuario\" where usua =
\""+username+"\" and password = '"+password+'\"";
```

```

try{
    ConexionPsql myDbTest = new ConexionPsql();
    Statement select = myDbTest.getConnection().createStatement();
    ResultSet result = select.executeQuery(statementPostgrasql);
    while (result.next()) {
        valid=true;
    }
}
catch(Exception Ex){
}

return valid;
}
}

```

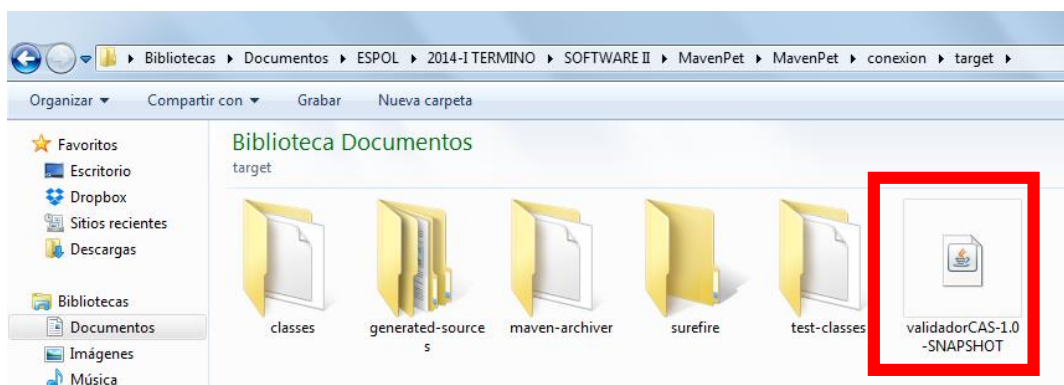
5. En la pom.xml deben estar las siguientes dependencias

```

<dependency>
    <groupId>org.jasig.cas</groupId>
    <artifactId>cas-server-core</artifactId>
    <version>3.5.2</version>
</dependency>
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>9.3-1100-jdbc41</version>
</dependency>

```

6. Ahora ya teniendo tu tabla en la base de datos puedes enviar a correr el Java Application
7. Ahora como hacer el Cas utilice nuestra base de datos pues no es fácil pero debe hacer los siguientes pasos:
- Buscas el JAR que se crear al correr tu Java Application, por lo general se encuentra en esta ubicación.
 ...\\MavenPet\\MavenPet\\conexion\\target\\



8. Ese .jar debe ser copiado en la carpeta lib del cas + el .jar del postgresql que tiene la siguiente ruta:

C:\glassfish4\glassfish\domains\domain1\applications\cas-server-webapp-3.5.2\WEB-INF\lib

9. Y por último se agrega una línea a el siguiente doc: deployerConfigContext.xml ubicado en:

C:\glassfish4\glassfish\domains\domain1\applications\cas-server-webapp-3.5.2\WEB-INF\ deployerConfigContext.xml

10. Se debe buscar la siguiente línea en el documento :

```
<bean  
    class="org.jasig.cas.authentication.handler.support.SimpleTestUser  
    namePasswordAuthenticationHandler" />
```

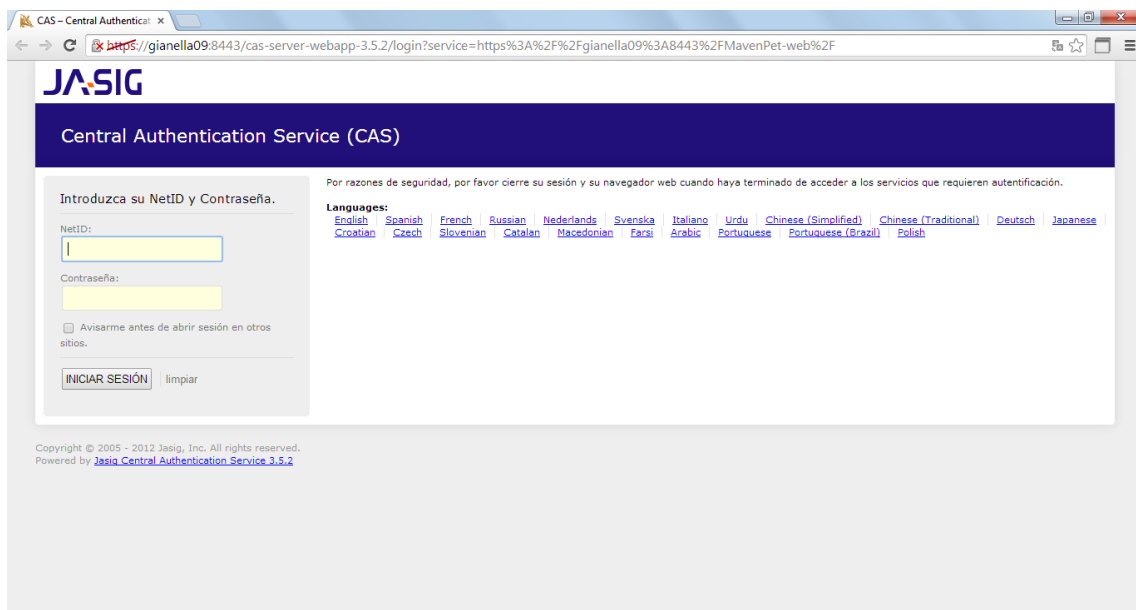
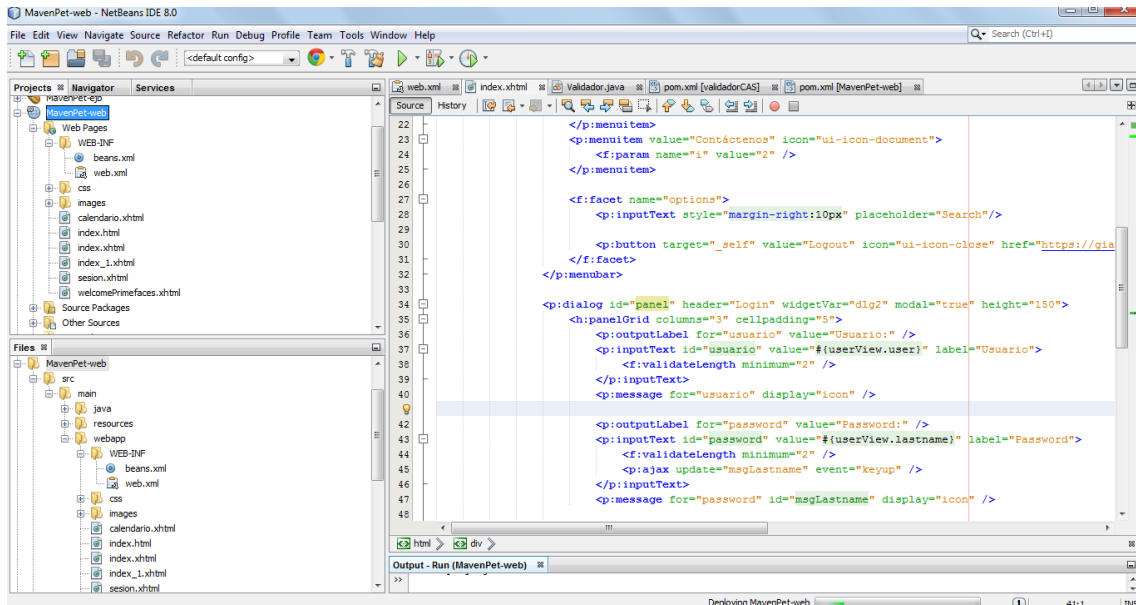
- Comentarla y cambiarla por la siguiente, teniendo en cuenta que es la ruta compuesta con el nombre del paquete de su Java Application + el nombre de la Clase Validador

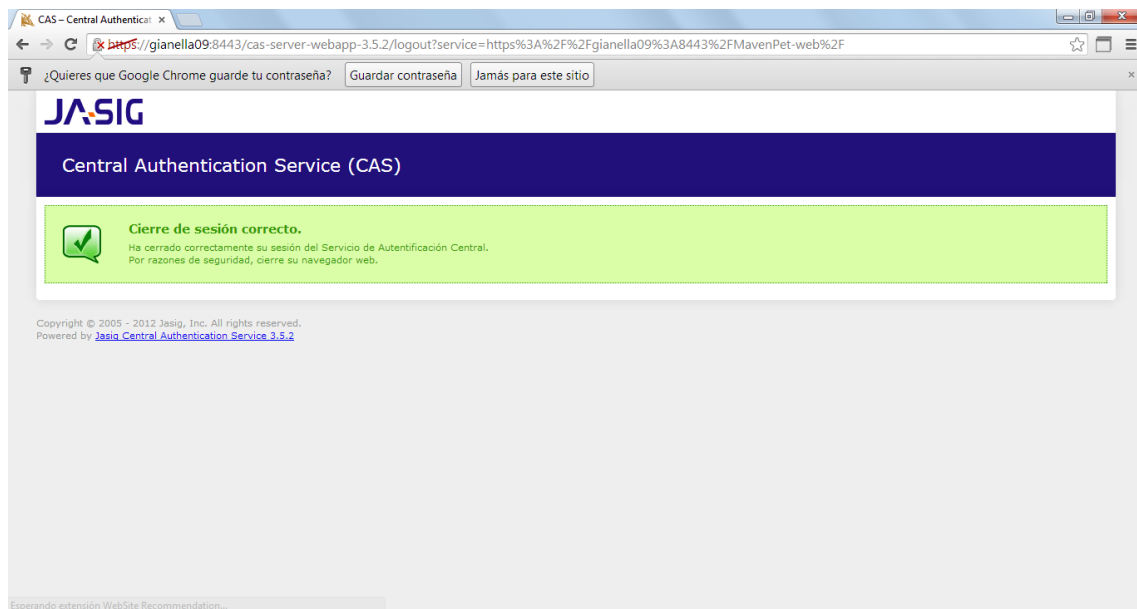
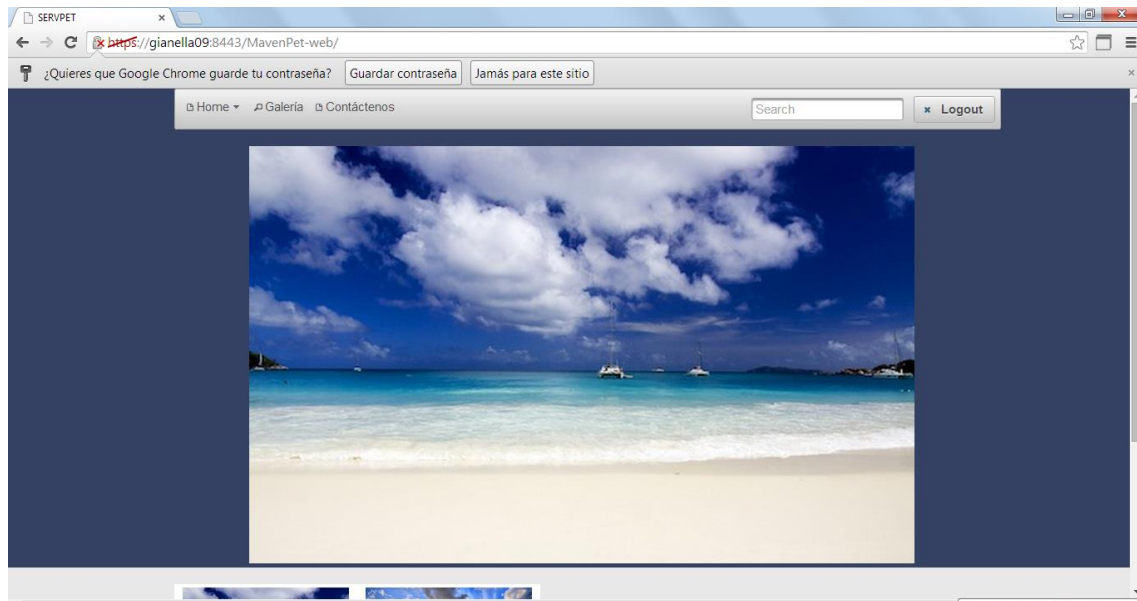
```
<bean  
  
    class="com.software.validadorcas.conexion.Validador" />
```

11. Para el LOGOUT se crea un botón en la aplicación y se agrega el mismo URL del cas de login cambiando el LOGIN por LOGOUT

```
<p: button target="_self" value="Logout" href="https://gianella09:8443/cas-  
server-webapp-  
3.5.2/logout?service=https%3A%2F%2Fgianella09%3A8443%2FMavenPet-  
web%2F"/>
```

DEMO





**LISTO AHORA YA PUEDE DISFRUTAR DE SU APLICACIÓN MAS EL CAS
JASIG**