	UNIVERSIDAD DON BOSCO **FACULTAD DE ESTUDIOS TECNOLÓGICOS** **COORDINACIÓN DE COMPUTACIÓN Y MÓVILES**
Ciclo II	**Desarrollo de aplicaciones con Web Frameworks** **Guía de Laboratorio No. 12** **JSF y Spring Security**

I. Objetivos

Que el alumno utilice spring security para crear una capa de seguridad en sus aplicaciones.

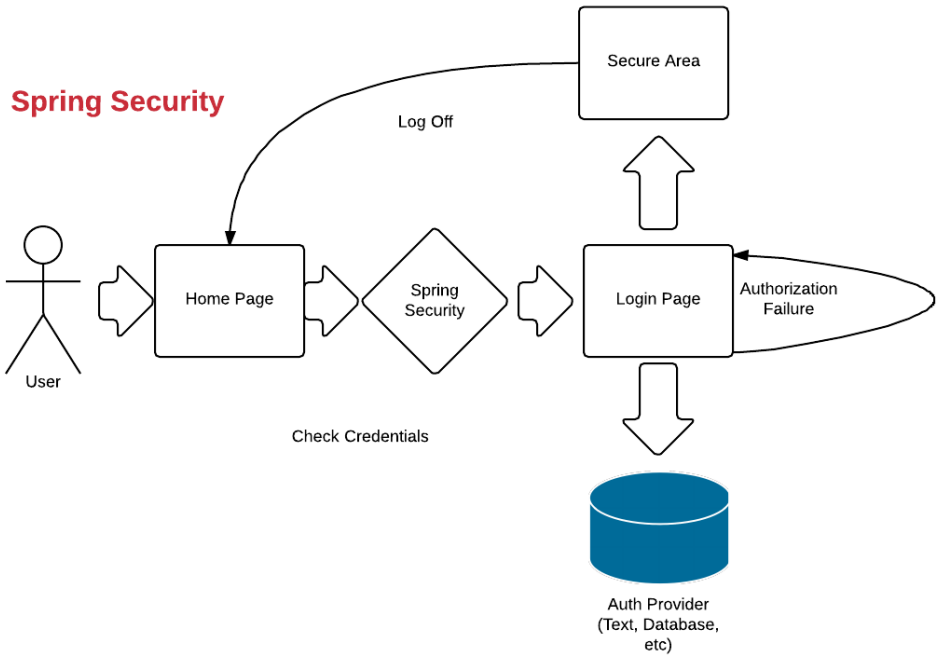
Que el alumno sea capaz de crear usuarios y roles.

II. INTRODUCCIÓN

Spring Security es un subproyecto del framework Spring, que permite gestionar completamente la seguridad de nuestras aplicaciones Java, y cuyas ventajas principales son las siguientes:

- Es capaz de gestionar seguridad en varios niveles: URLs que se solicitan al servidor, acceso a métodos y clases Java, y acceso a instancias concretas de las clases.
- Permite separar la lógica de nuestras aplicaciones del control de la seguridad, utilizando filtros para las peticiones al servidor de aplicaciones o aspectos para la seguridad en clases y métodos.
- La configuración de la seguridad es portable de un servidor a otro, ya que se encuentra dentro del WAR o el EAR de nuestras aplicaciones.
- Soporta muchos modelos de identificación de los usuarios (HTTP BASIC, HTTP Digest, basada en formulario, LDAP, OpenID, JAAS y muchos más). Además podemos ampliar estos mecanismos implementando nuestras propias clases que extiendan el modelo de Spring Security.

Aún siendo un sub proyecto de Spring, es posible utilizar las dependencias de Spring Security junto con otras herramientas JEE, tales como JSP, JSF, etc. Basta con realizar las configuraciones necesarias para su integración.



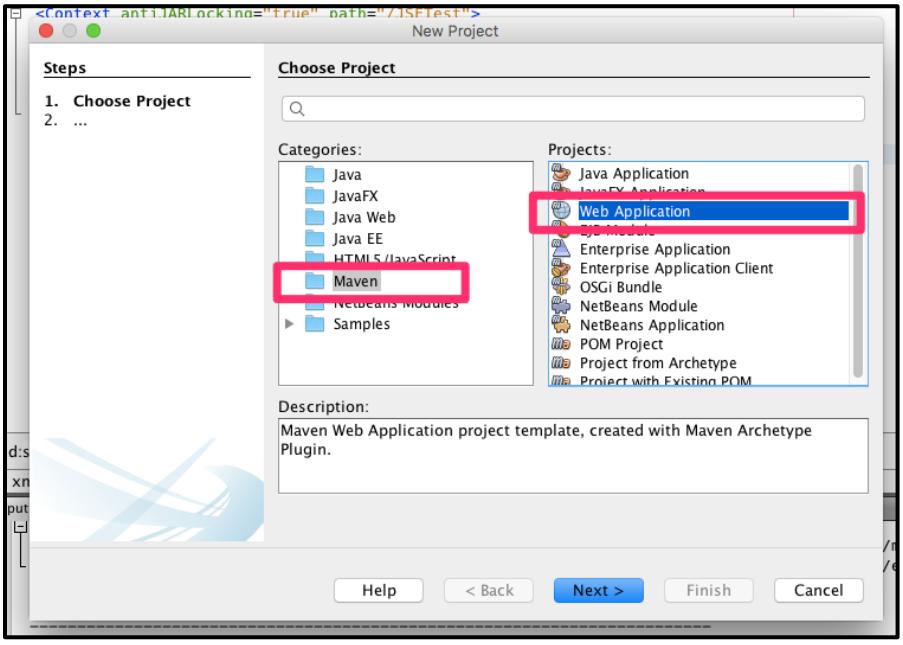
III. DESARROLLO

Cree la base de datos:

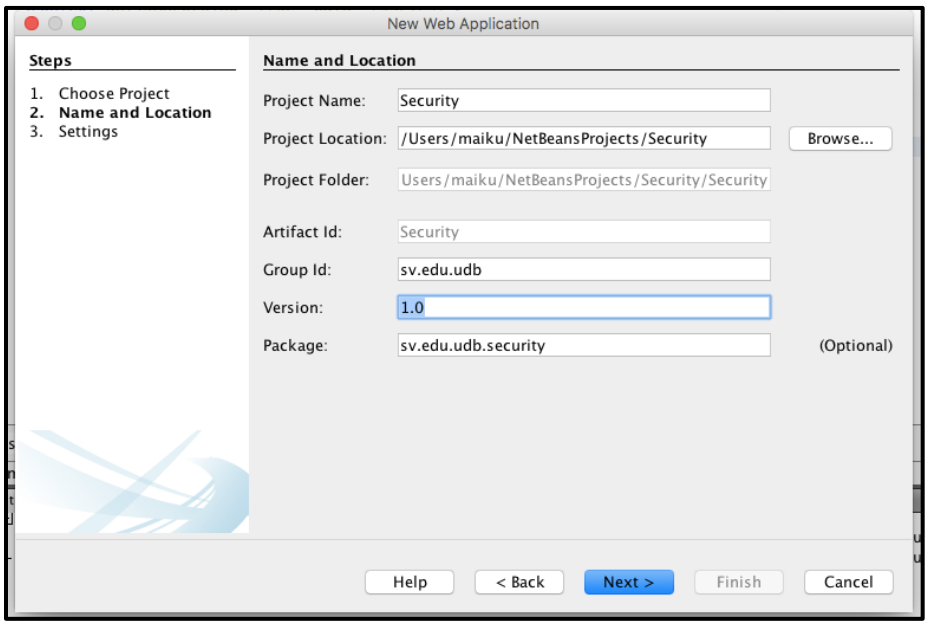
```
CREATE DATABASE SECURITY;

USE SECURITY;

CREATE  TABLE users (
  username VARCHAR(45) NOT NULL ,
  password VARCHAR(45) NOT NULL ,
  enabled TINYINT NOT NULL DEFAULT 1 ,
  PRIMARY KEY (username));



  CREATE TABLE user_roles (
  user_role_id int(11) NOT NULL AUTO_INCREMENT,
  username varchar(45) NOT NULL,
```

```
role varchar(45) NOT NULL,
PRIMARY KEY (user_role_id),
UNIQUE KEY uni_username_role (role,username),
KEY fk_username_idx (username),
CONSTRAINT fk_username FOREIGN KEY (username) REFERENCES users (username));
```

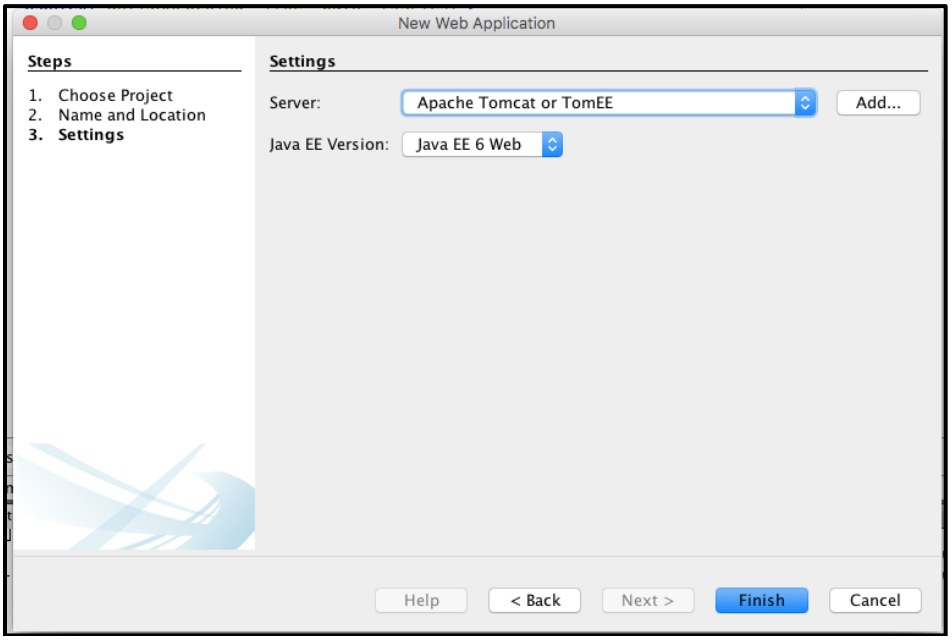
Cree un Nuevo Proyecto por medio de Maven:



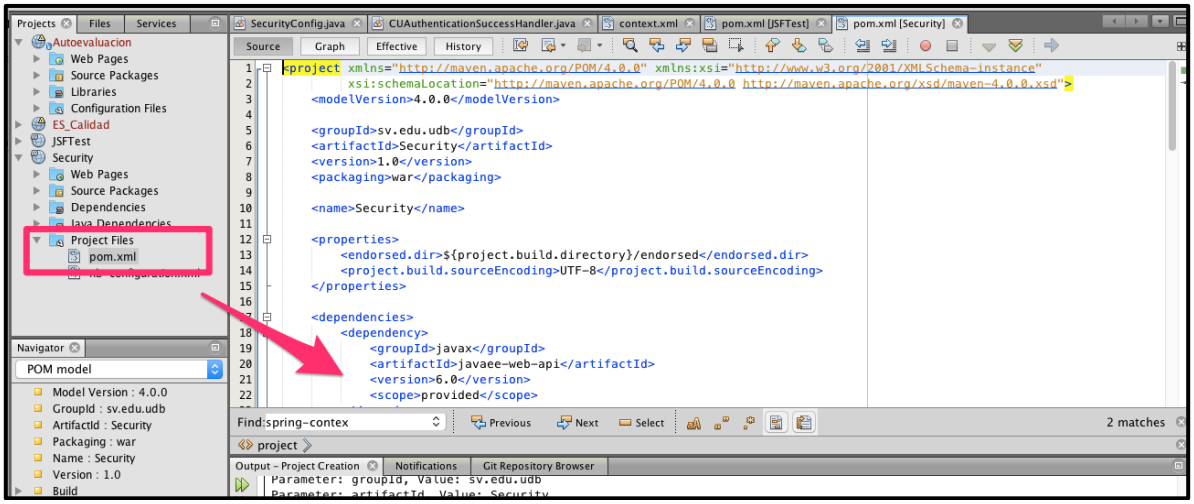
El nombre del proyecto será Security:



Para el ejemplo usará el contenedor Apache Tomcat 7 o superior.



Agregar dependencias a Maven dentro de la sección dependencias:



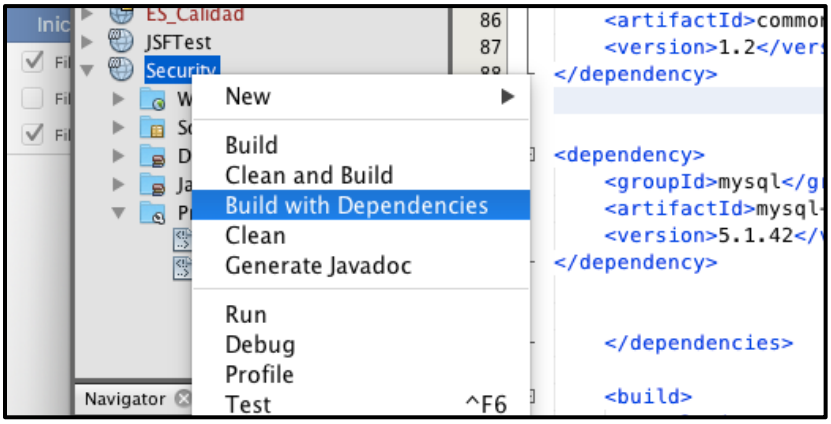
Nota: Procure agregar las dependencias dentro del par de etiquetas Dependencies como puede observarse en la ilustración la ilustración anterior.

```
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-core</artifactId>
   <version>4.0.1.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-context</artifactId>
   <version>4.0.1.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-jdbc</artifactId>
   <version>4.0.1.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-beans</artifactId>
   <version>4.0.1.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-aop</artifactId>
   <version>4.0.1.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework</groupId>
```

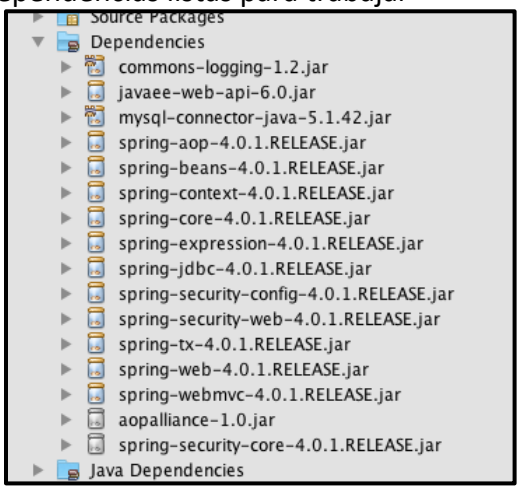
```
   <artifactId>spring-tx</artifactId>
   <version>4.0.1.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-expression</artifactId>
   <version>4.0.1.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-web</artifactId>
   <version>4.0.1.RELEASE</version>
</dependency>

<dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-web</artifactId>
      <version>4.0.1.RELEASE</version>
   </dependency>

<dependency>
   <groupId>org.springframework.security</groupId>
   <artifactId>spring-security-config</artifactId>
   <version>4.0.1.RELEASE</version>
</dependency>
<dependency>
 <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
   <version>4.0.1.RELEASE</version>
</dependency>


<dependency>
   <groupId>commons-logging</groupId>
   <artifactId>commons-logging</artifactId>
   <version>1.2</version>
</dependency>

<dependency>
   <groupId>mysql</groupId>
   <artifactId>mysql-connector-java</artifactId>
   <version>5.1.42</version>
</dependency>
```

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

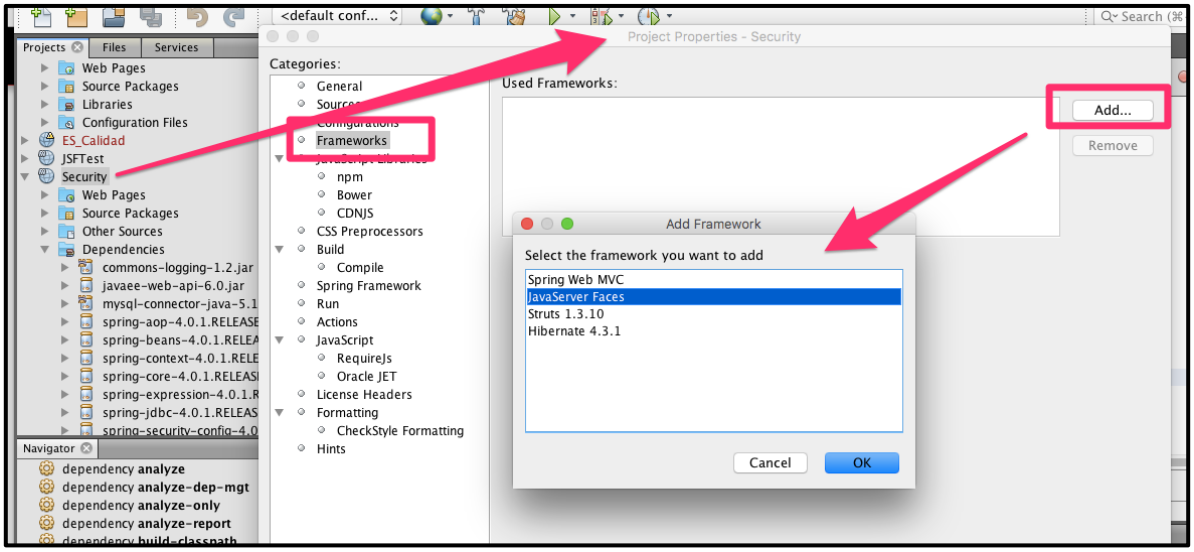
Vamos a sincronizar las dependencias:



Maven dejará todas las dependencias listas para trabajar

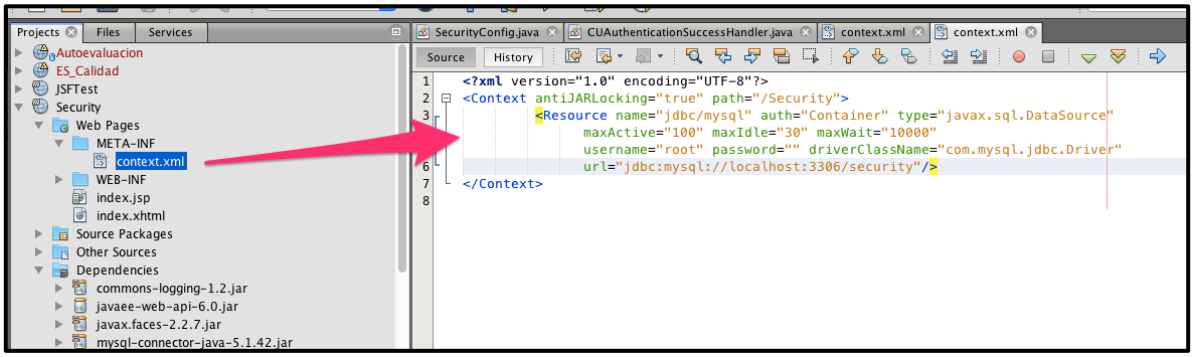


Aunque puede hacerlo directamente desde Maven, usted agregará el framework JSF 2.2 con Netbeans:



Deberá sincronizar el proyecto una vez más con “Clean and Build”.

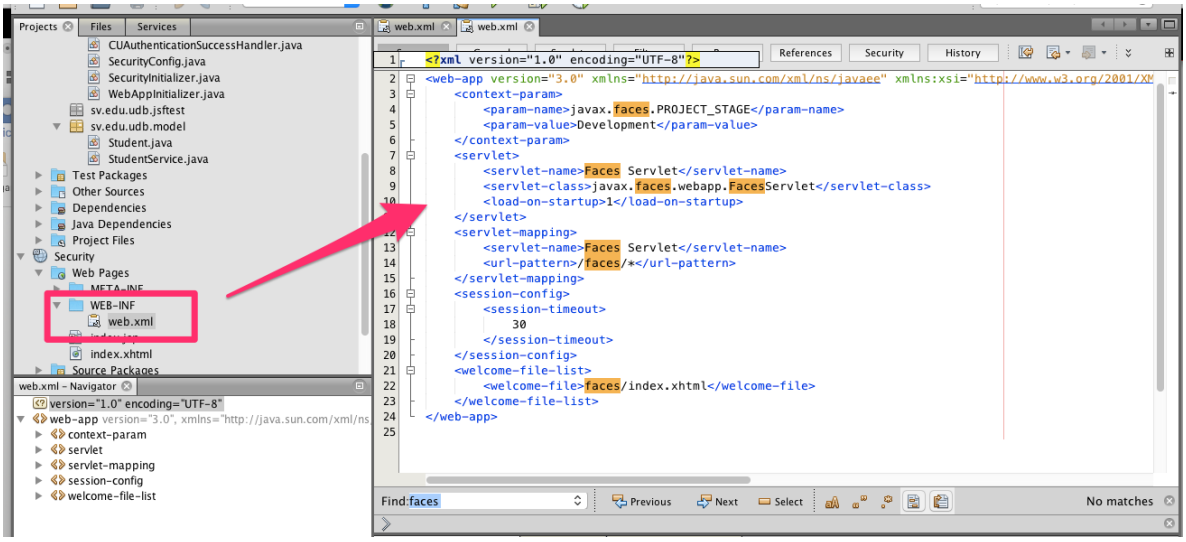
Para esta guía es requerido un DataSource con Tomcat, por lo que editaremos el fichero context y agregaremos los siguiente:



Tome en cuenta que el usuario y password pueden ser distintos dependiendo del server mysql que vaya a utilizar.

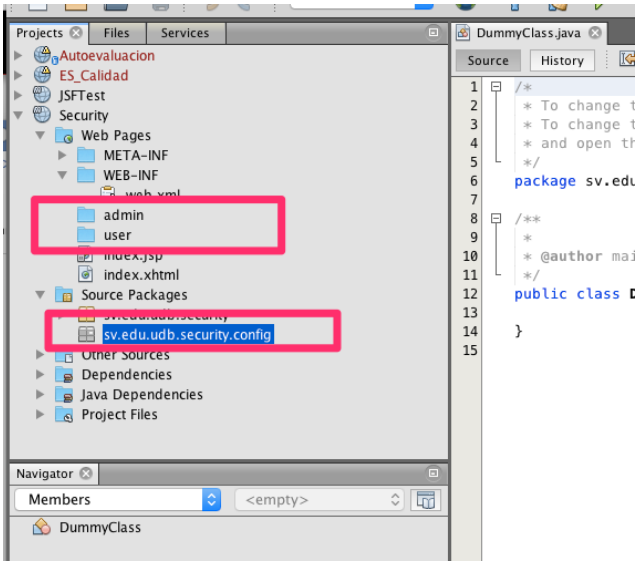
```
<Context antiJARLocking="true" path="/Security">
    <Resource name="jdbc/mysql" auth="Container" type="javax.sql.DataSource"
        maxActive="100" maxIdle="30" maxWait="10000"
        username="root" password="" driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/security"/>
</Context>
```

Corrobore que tiene las configuraciones de su proyecto correctas, desde web.xml que se ha generado automáticamente al incluir el framework JSF. Recuerde que en su proyecto el orden de los elementos puede ser diferentes.



Ahora creará dos carpetas en la raíz de Web Pages, denominadas “user” y “admin”. Además agregará un nuevo paquete denominado “sv.edu.udb.security.config”

Nota: Procure evitar que sus carpetas queden dentro de WEB-INF o META_INF. De lo contrario, no serán accesibles.



Cree el fichero denominado SecurityConfig.java dentro del paquete sv.edu.udb.security.config

```
package sv.edu.udb.security.config;
```

```
public class SecurityConfig {

}
```

Cree el fichero WebAppInitializer.java dentro del paquete sv.edu.udb.security.config

```
package sv.edu.udb.security.config;

import
org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletIn
itializer;
public class WebAppInitializer extends
AbstractAnnotationConfigDispatcherServletInitializer {
	@Override
	protected Class<?>[] getRootConfigClasses() {
		return new Class[] { SecurityConfig.class };
	}
	@Override
	protected Class<?>[] getServletConfigClasses() {
		return null;
	}
	@Override
	protected String[] getServletMappings() {
		return new String[] { "/" };
	}
}
```

Cree el fichero SecurityInitializer.java dentro del paquete sv.edu.udb.security.config

```
package sv.edu.udb.security.config;

import
org.springframework.security.web.context.AbstractSecurityWebApplicationInitializer;
public class SecurityInitializer extends AbstractSecurityWebApplicationInitializer {
}
```

Debe crear un fichero denominado CUAuthenticationSuccessHandler.java

```

```

```
package sv.edu.udb.security.config;

import java.io.IOException;
import java.util.Collection;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.web.DefaultRedirectStrategy;
import org.springframework.security.web.RedirectStrategy;
import org.springframework.security.web.WebAttributes;
import
org.springframework.security.web.authentication.AuthenticationSuccessHandler;

public class CUAuthenticationSuccessHandler implements
AuthenticationSuccessHandler{

   protected Log logger = LogFactory.getLog(this.getClass());

   private RedirectStrategy redirectStrategy = new DefaultRedirectStrategy();

   @Override
   public void onAuthenticationSuccess(HttpServletRequest hsr, HttpServletResponse
hsr1, Authentication a) throws IOException, ServletException {

      handle(hsr, hsr1, a);
      clearAuthenticationAttributes(hsr);

   }

    protected void handle(HttpServletRequest request,
     HttpServletResponse response, Authentication authentication)
     throws IOException {

      String targetUrl = determineTargetUrl(authentication);

      if (response.isCommitted()) {
         logger.debug(
           "Response has already been committed. Unable to redirect to "
           + targetUrl);
```

```
        return;
    }

    redirectStrategy.sendRedirect(request, response, targetUrl);
}

protected String determineTargetUrl(Authentication authentication) {
    boolean isUser = false;
    boolean isAdmin = false;
    Collection<? extends GrantedAuthority> authorities
     = authentication.getAuthorities();
    for (GrantedAuthority grantedAuthority : authorities) {
        if (grantedAuthority.getAuthority().equals("ROLE_USER")) {
            isUser = true;
            break;
        } else if (grantedAuthority.getAuthority().equals("ROLE_ADMIN")) {
            isAdmin = true;
            break;
        }
    }

    if (isUser) {
        return "/faces/user/user.xhtml";
    } else if (isAdmin) {
        return "/faces/admin/admin.xhtml";
    } else {
        throw new IllegalStateException();
    }
}

protected void clearAuthenticationAttributes(HttpServletRequest request) {
    HttpSession session = request.getSession(false);
    if (session == null) {
        return;
    }
    session.removeAttribute(WebAttributes.AUTHENTICATION_EXCEPTION);
}

public void setRedirectStrategy(RedirectStrategy redirectStrategy) {
    this.redirectStrategy = redirectStrategy;
}
protected RedirectStrategy getRedirectStrategy() {
    return redirectStrategy;
}
```

```
}
```

Ahora modificará SecurityConfig.java para que tenga el siguiente contenido:

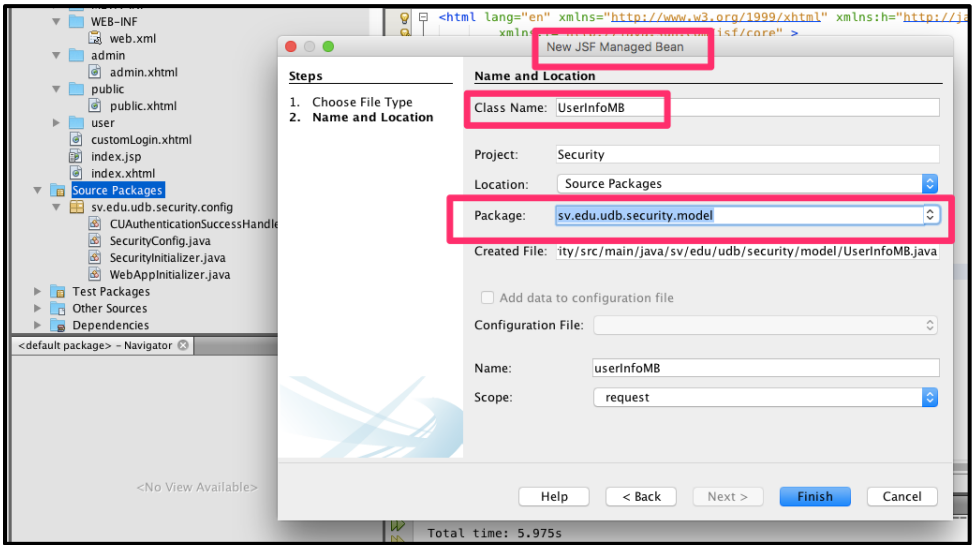
```
package sv.edu.udb.security.config;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.sql.DataSource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.Authentication
ManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigu
rerAdapter;
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

      Context ctx = null;

          @Override
          protected void configure(HttpSecurity http) throws Exception {
            http.authorizeRequests().
            antMatchers("/faces/admin/**").access("hasRole('ROLE_ADMIN')").
            antMatchers("/faces/user/**").access("hasRole('ROLE_USER')").
            and().formLogin().  //login configuration
            loginPage("/faces/customLogin.xhtml").
            loginProcessingUrl("/appLogin").
            usernameParameter("app_username").
            passwordParameter("app_password").
            successHandler(new CUAuthenticationSuccessHandler()).
                  and().logout().    //logout configuration
                  logoutUrl("/appLogout").
                  logoutSuccessUrl("/faces/customLogin.xhtml");
```

```
        }

        @Autowired
        public void configureGlobal(AuthenticationManagerBuilder auth) throws
Exception {
          ctx = new InitialContext();
          DataSource dataSource = (DataSource)
ctx.lookup("java:/comp/env/jdbc/mysql");
          auth.jdbcAuthentication().dataSource(dataSource).
               usersByUsernameQuery(
                    "select username,password, enabled from users where
username=?")
                .authoritiesByUsernameQuery(
                    "select username, role from user_roles where username=?");
                }
}
```

Cree y edite la vista customLogin.xhtml:

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Spring 4 Security  + JSF 2 </title>
  </h:head>
  <h:body>
        <h3>Spring 4 Security  + JSF 2</h3>
        <font color="red">
              <h:outputLabel
value="${SPRING_SECURITY_LAST_EXCEPTION.message}"/>
        </font>
            <form action="${request.contextPath}/appLogin" method="POST">
                <h:outputLabel value="Enter UserName:" />
                <input type="text" name="app_username"/><br/><br/>
                <h:outputLabel value="Enter Password:" />
                <input type="password" name="app_password"/> <br/><br/>

                <input type="submit" value="Login"/>
                <input type="hidden"
                  name="${_csrf.parameterName}" value="${_csrf.token}"/>
            </form>
  </h:body>
</html>
```

Ahora cree un ManagedBean llamado “UserInfoMB.java”




```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package sv.edu.udb.security.model;

import java.util.ArrayList;
import java.util.Collection;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.context.SecurityContextHolder;

/**
 *
 * @author maiku
 */
@ManagedBean
@RequestScoped
public class UserInfoMB {


   Authentication auth;
```

```
    public UserInfoMB() {
       auth= SecurityContextHolder.getContext().getAuthentication();
    }


    public String getUserName(){

     String name = auth.getName(); //get logged in username
     return name;
    }

    public ArrayList<String> getUserRole(){
     ArrayList<String> list = new ArrayList();
     Collection<? extends GrantedAuthority> authorities
       = auth.getAuthorities();

     for (GrantedAuthority grantedAuthority : authorities) {
         list.add(grantedAuthority.getAuthority());
      }

     return list;
    }

}
```

Cree un archivo denominado admin.xhtml en la carpeta admin:

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://java.sun.com/jsf/html"
     xmlns:f="http://java.sun.com/jsf/core"
     xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
    <h:head>
       <title>Spring 4 Security  + JSF 2</title>
    </h:head>
    <h:body>
      <form action="${request.contextPath}/appLogout" method="POST">
        <input type="submit" value="Logout"/>
        <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}"/>

      </form>
      <h3>Spring Security y JSF</h3>
          <h:form>
          <H1>ESTA PÁGINA ES PARA ADMINISTRACIÓN</H1>
```

```
        <h2> Usuario #{userInfoMB.userName} </h2>
        <p>
           Permisos
           <ui:repeat value="#{userInfoMB.userRole}" var="value">
              #{value} <br />
           </ui:repeat>
        </p>
        <P>
           Esta página sólo debe ser accedida por el rol Administrador.
        </P>
         <input type="hidden" name="${_csrf.parameterName}"
value="${_csrf.token}"/>
         </h:form>
   </h:body>
</html>
```

Cree user.xhtml en la carpeta user:

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://java.sun.com/jsf/html"
     xmlns:f="http://java.sun.com/jsf/core"
     xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
   <h:head>
      <title>Spring 4 Security  + JSF 2</title>
   </h:head>
   <h:body>
      <form action="${request.contextPath}/appLogout" method="POST">
        <input type="submit" value="Logout"/>
        <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}"/>

      </form>
      <h3>Spring Security y JSF</h3>
          <h:form>
        <H1>ESTA PÁGINA ES PARA USUARIO</H1>
        <h2> Usuario #{userInfoMB.userName} </h2>
        <p>
           Permisos
           <ui:repeat value="#{userInfoMB.userRole}" var="value">
              #{value} <br />
           </ui:repeat>
        </p>
        <P>
           Esta página sólo debe ser accedida por un usuario.
        </P>
```

```
        <input type="hidden" name="${_csrf.parameterName}"
value="${_csrf.token}"/>
        </h:form>
   </h:body>
</html>
```

A continuación cree una carpeta denominada “public” con un fichero public.xhtml

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://java.sun.com/jsf/html"
        xmlns:f="http://java.sun.com/jsf/core" >
   <h:head>
      <title>Spring 4 Security  + JSF 2</title>
   </h:head>
   <h:body>

     <h3>Spring Security y JSF</h3>
         <h:form>
        <H1>ESTA PÁGINA ES DE ACCESO PUBLICO</H1>
        <P>
           Acceso público
        </P>

         </h:form>
   </h:body>
</html>
```

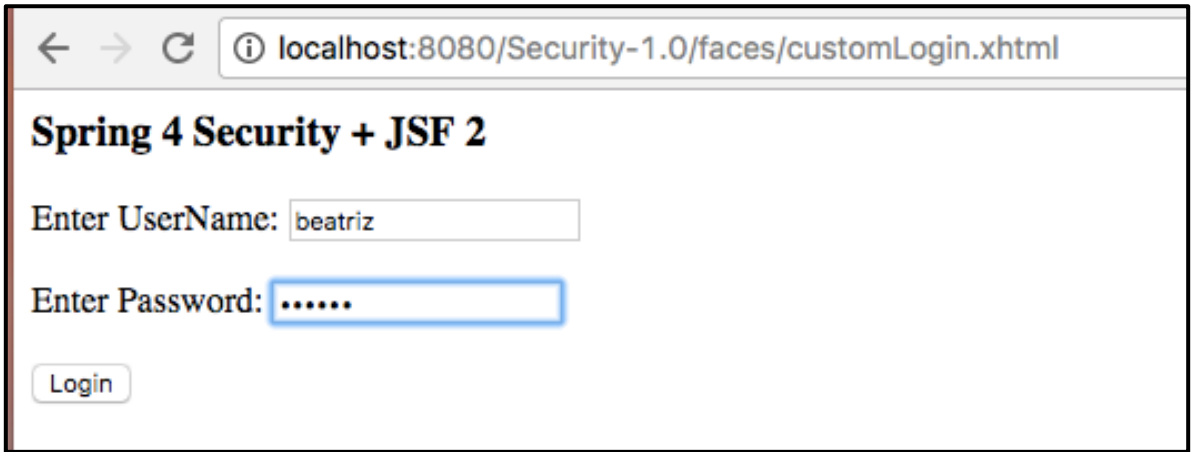
Cuando pruebe la aplicación podrá darse cuenta de lo siguiente:

Login usuario: udb
Password: udb

La redirección será hacia admin/admin.xhtml

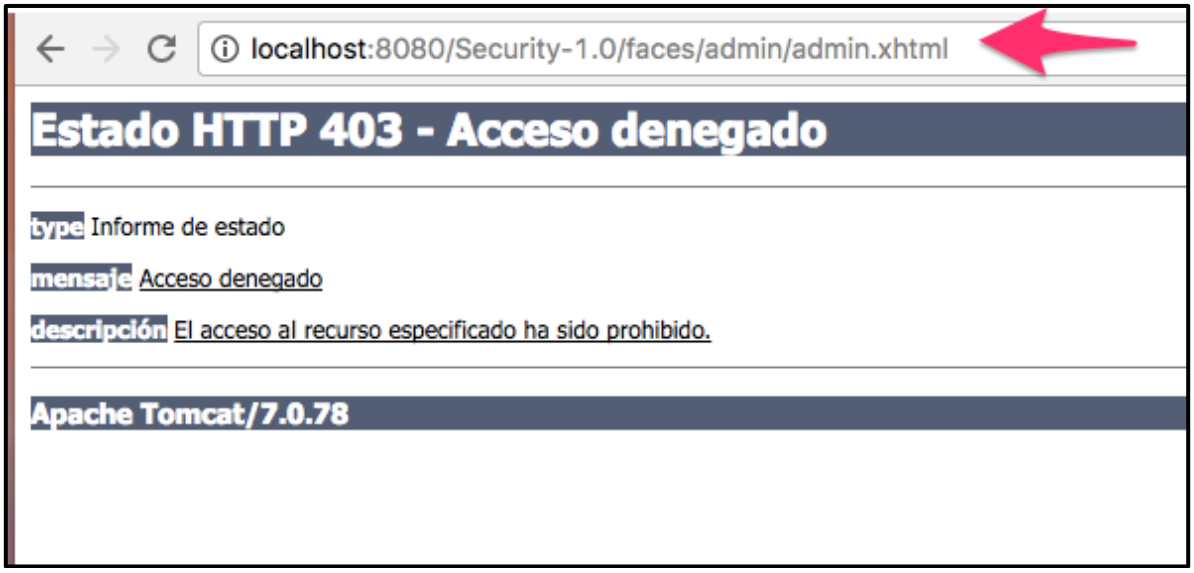
Al intentar ingresar por user/user.xhtml tendrá un acceso denegado:

Al ingresar con
Usuario: beatriz
Password: 123456



Obtendrá este resultado:

Y no podrá acceder a admin



La carpeta public tiene acceso disponible para cualquier usuario, logueado o no.

IV. INVESTIGACIÓN COMPLEMENTARIA

Elija una de sus guías de laboratorio anteriores e implemente seguridad con 2 roles de usuario utilizando Spring Security.

Cree un sistema de login para ser implementado en su proyecto de cátedra.

Investigue si es posible utilizar HttpSession para implementar seguridad en una aplicación con JSF.

V. BIBLIOGRAFIA RECOMENDADA

Paraschiv, E. (2017, April 07). Redirect to different pages after Login with Spring Security. Accedido 12 julio 2017, de http://www.baeldung.com/spring_redirect_after_login

Utilización de grupos en Spring Security. (n.d.). Accedido 12 de julio 2017, de https://www.adictosaltrabajo.com/tutoriales/utilizaciondegruposenspringsecurity/