



## ***“Spring Security Básico”***

### **Módulo 8 / 1**

© *Todos los logos y marcas utilizados en este documento, están registrados y pertenecen a sus respectivos dueños.*

## Objetivo

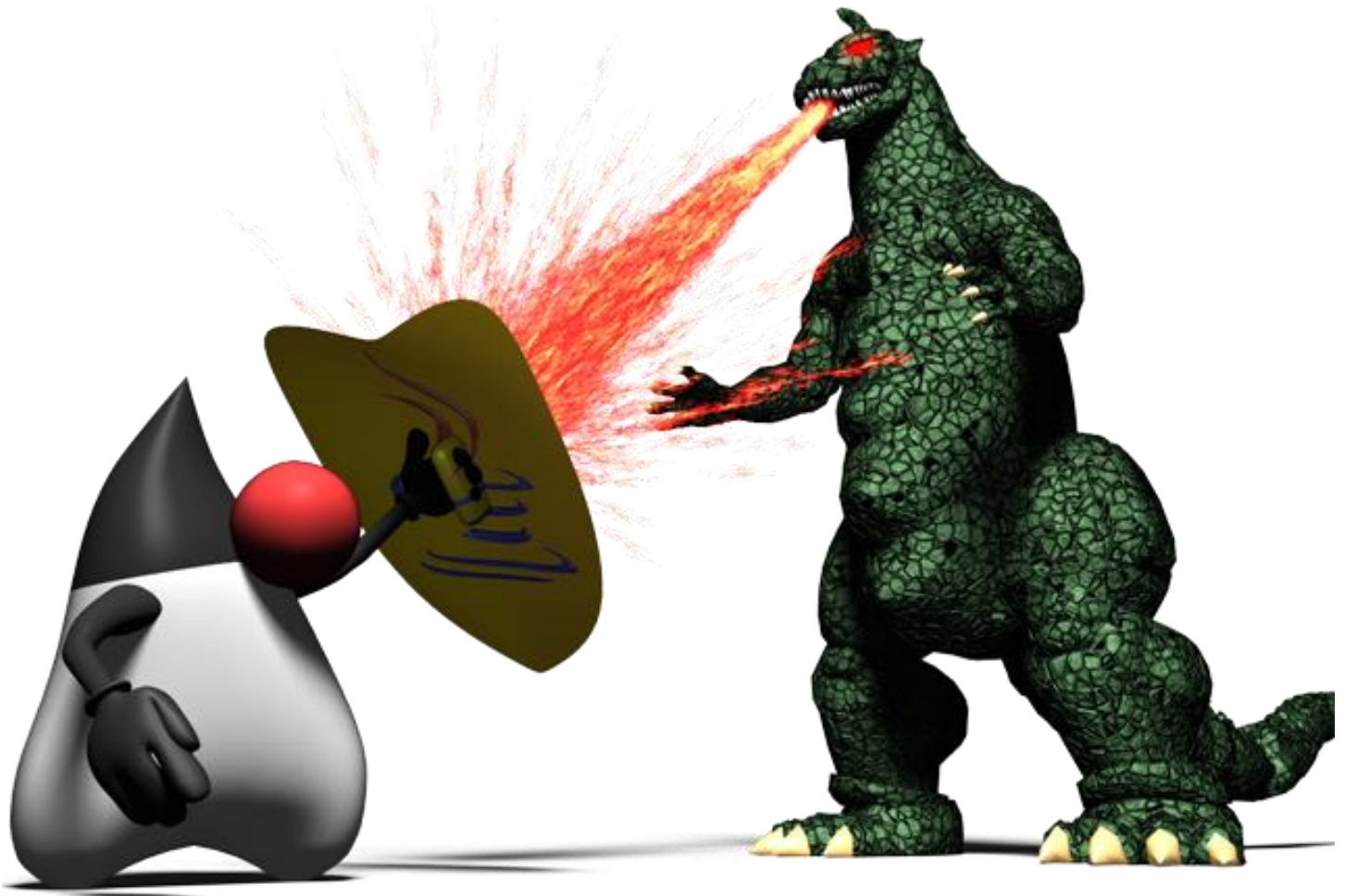
El objetivo de esta semana es entender cómo funciona el componente [Spring Security](#) y conocer una forma de implementar un sistema de autenticación con **Spring Framework**.

**Spring Security** es una potente herramienta de autenticación, altamente configurable y un framework de control de acceso. Es uno de los proyectos de Spring más maduros y ampliamente utilizados, desarrollado en 2003 y mantenido activamente por **Spring**, hoy en día se utiliza para proteger a innumerables plataformas a nivel mundial, incluyendo organismos gubernamentales, militares, retail, y en bancos e industrias financieras. Es liberado bajo licencia Apache 2.0 así que se puede utilizar con toda confianza en nuestros proyectos.

Spring Security es muy fácil de aprender, implementar y administrar. Permitiendo implementar una seguridad completa en nuestras aplicaciones tan solo con unas pocas líneas de XML/Anotaciones.

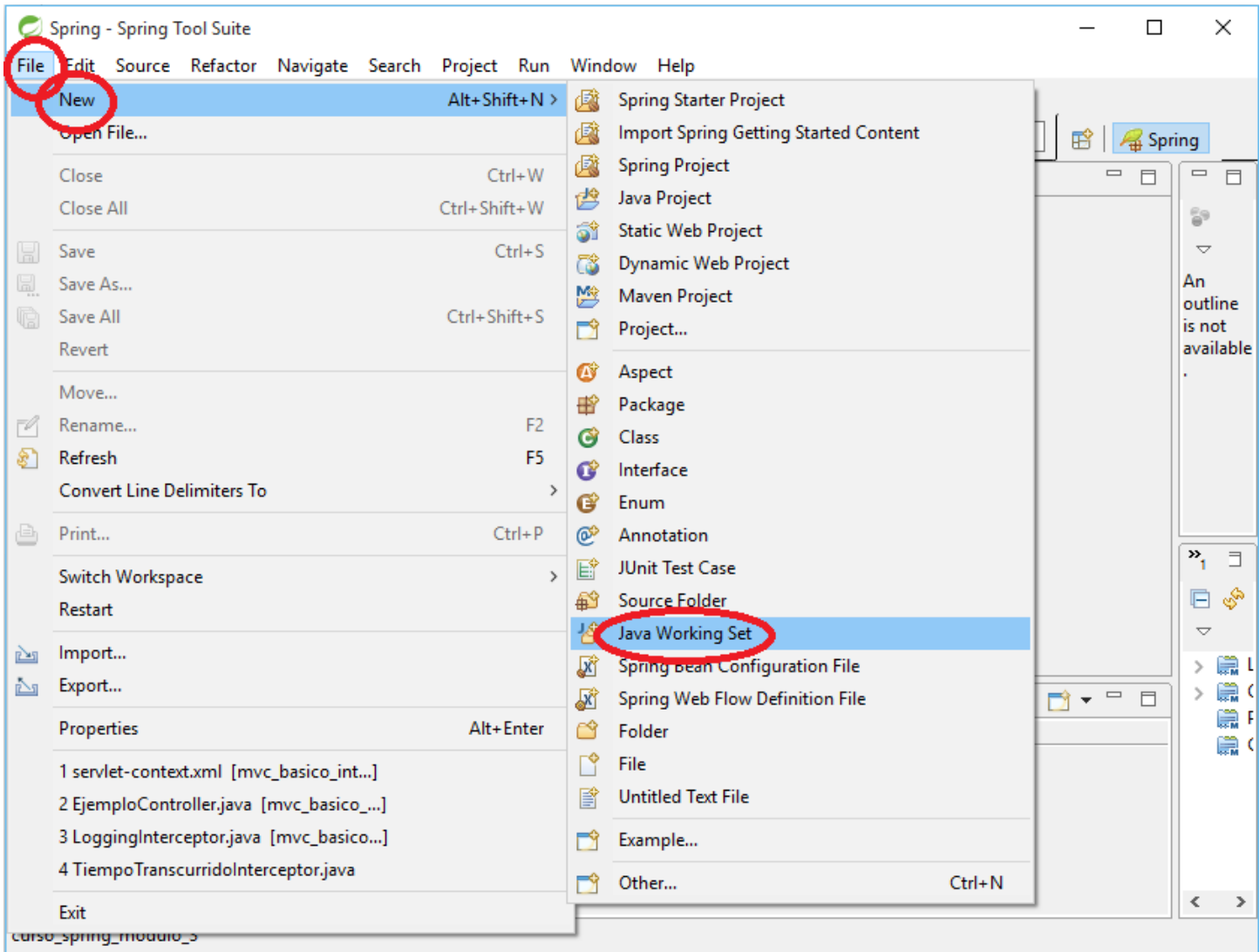
*"Quemar etapas"*

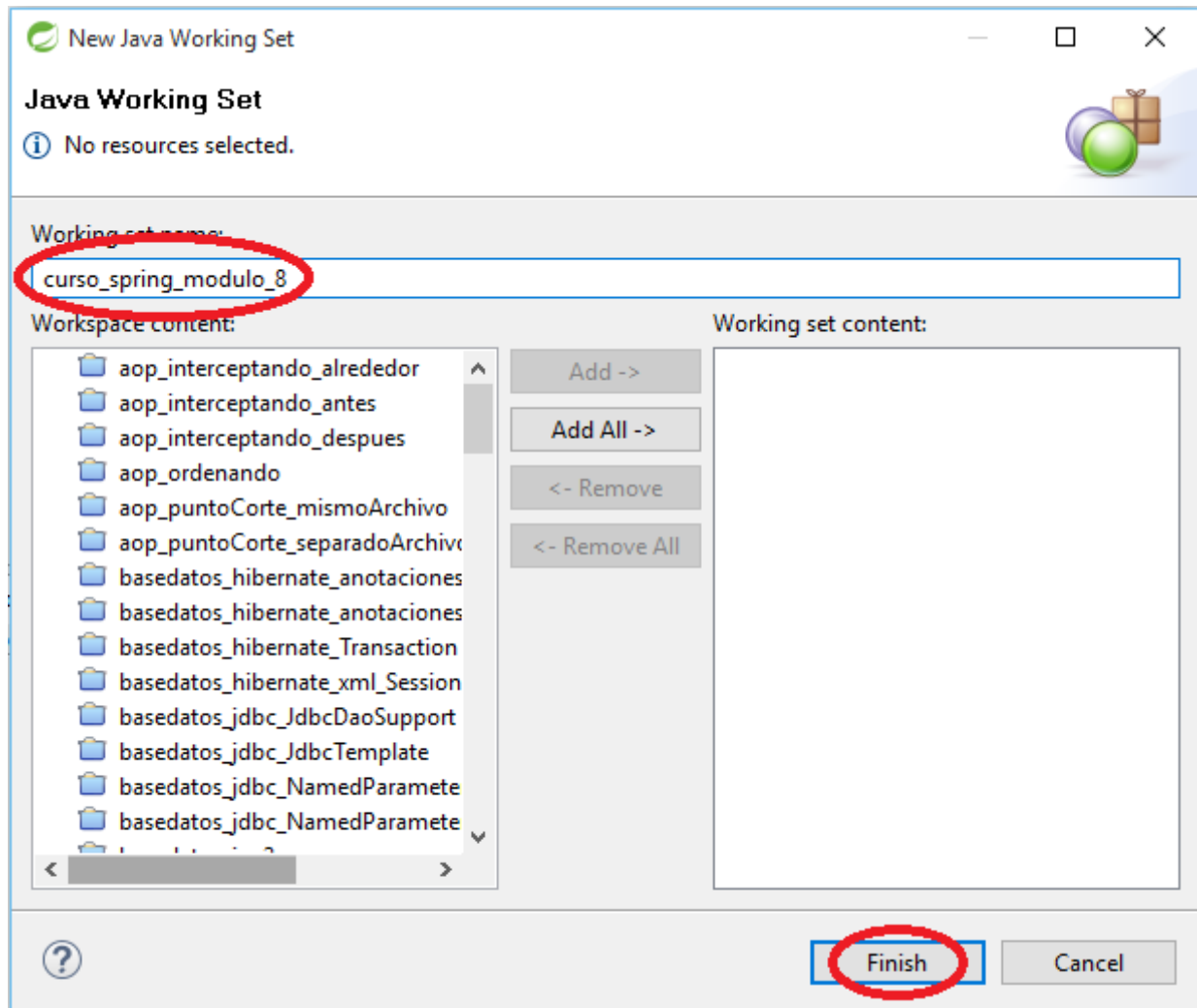
*Es importante que saques provecho de cada módulo y consultes todos los temas que se van tratando, sin adelantar etapas.*



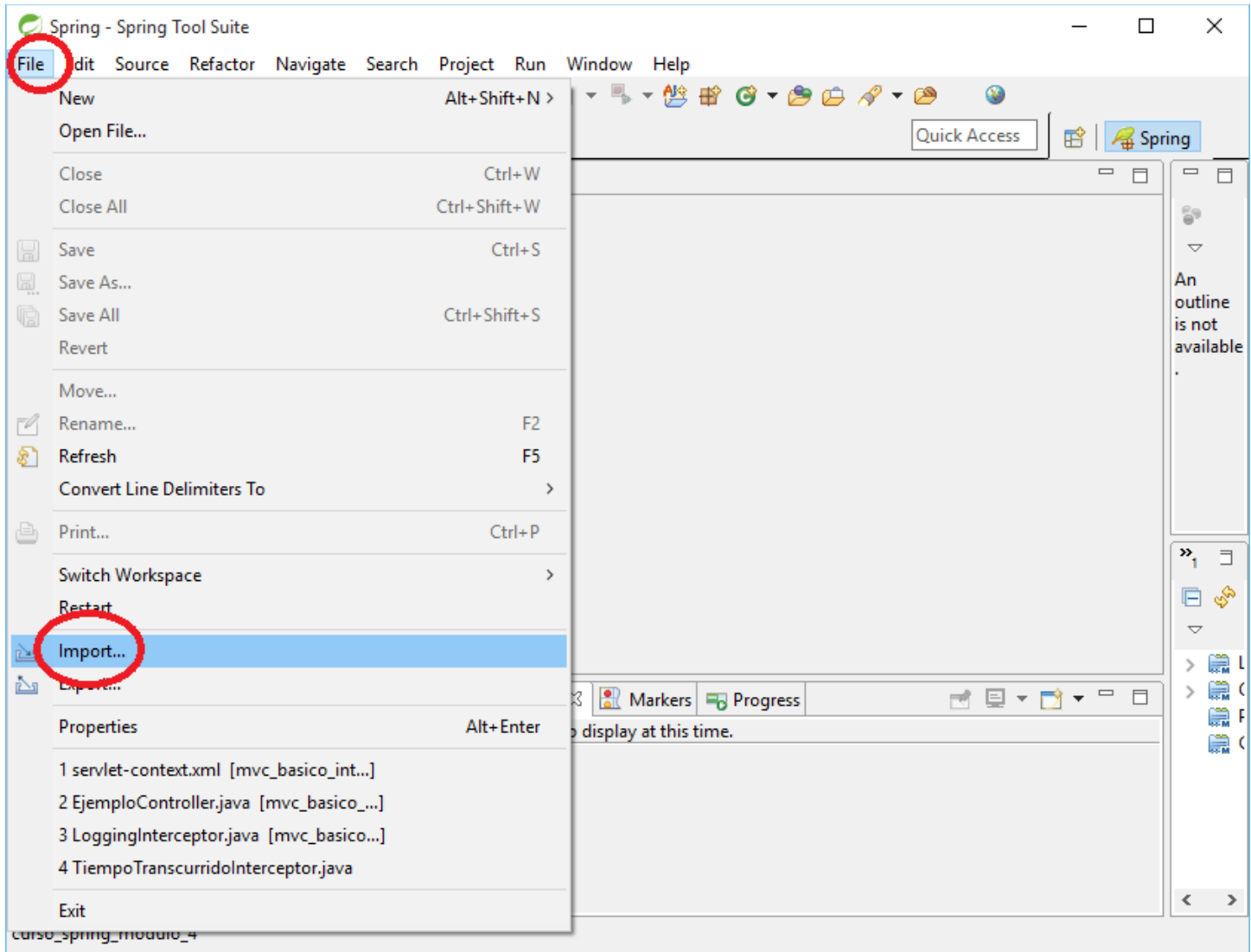
## Ejercicio 0: Importar todos los proyectos de ejemplo

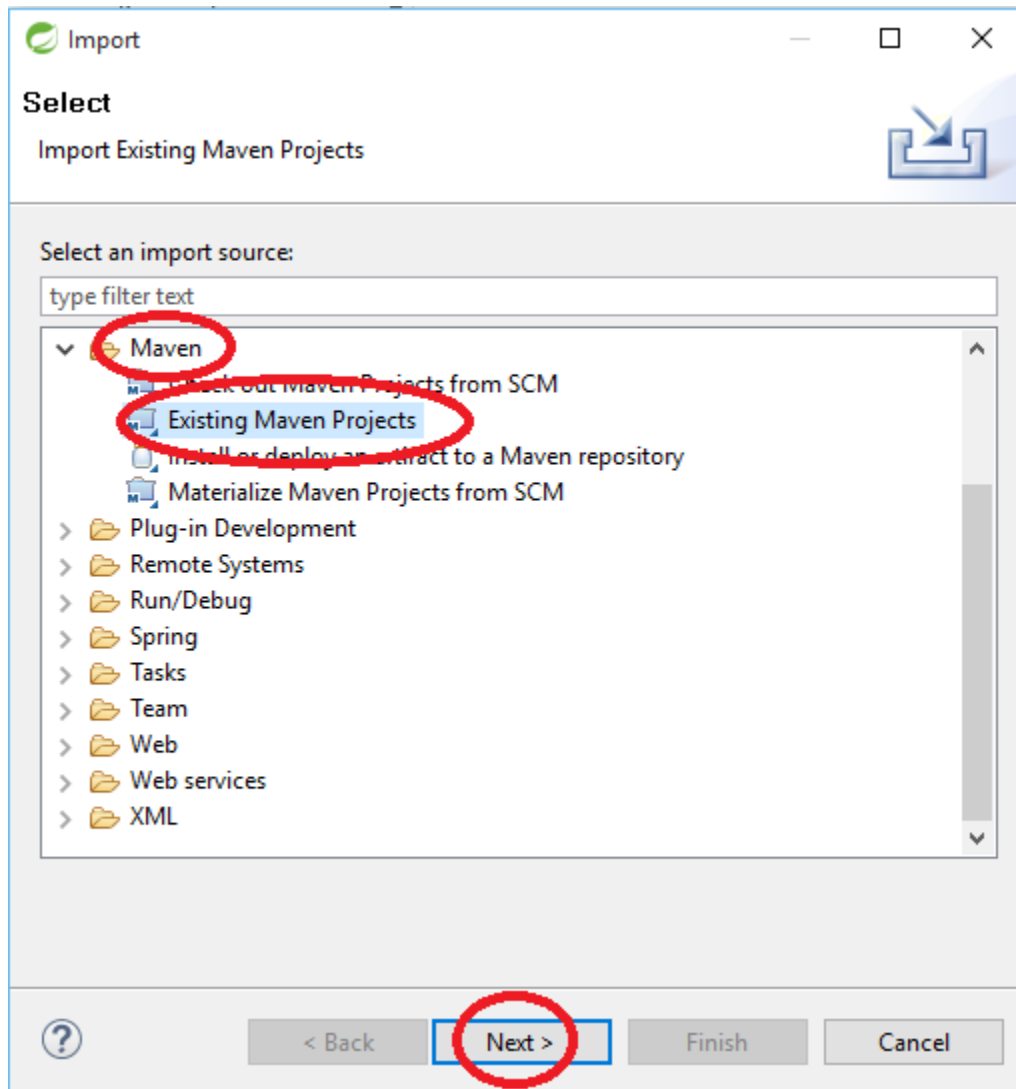
- Crear un nuevo "Java Working Set" llamado "**curso\_spring\_modulo\_8**". Esto es para organizar los proyectos bajo un esquema llamado **Working Set**, similar a como organizamos archivos en directorios.
  - Seleccionar **File->New->Java Working Set**.



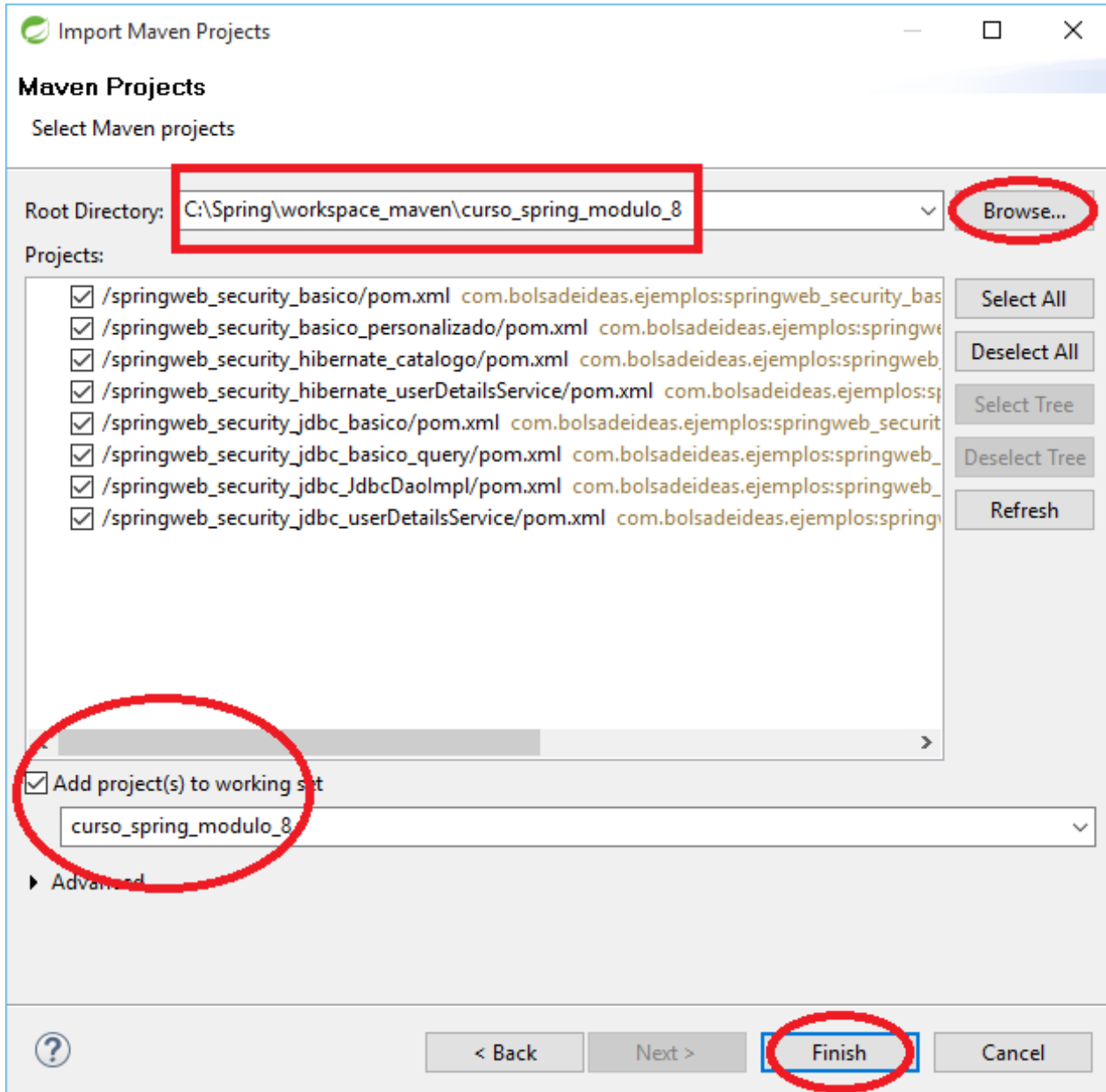


- Importar los proyectos de ejemplos en maven.
  - Seleccionar **File->Import**.

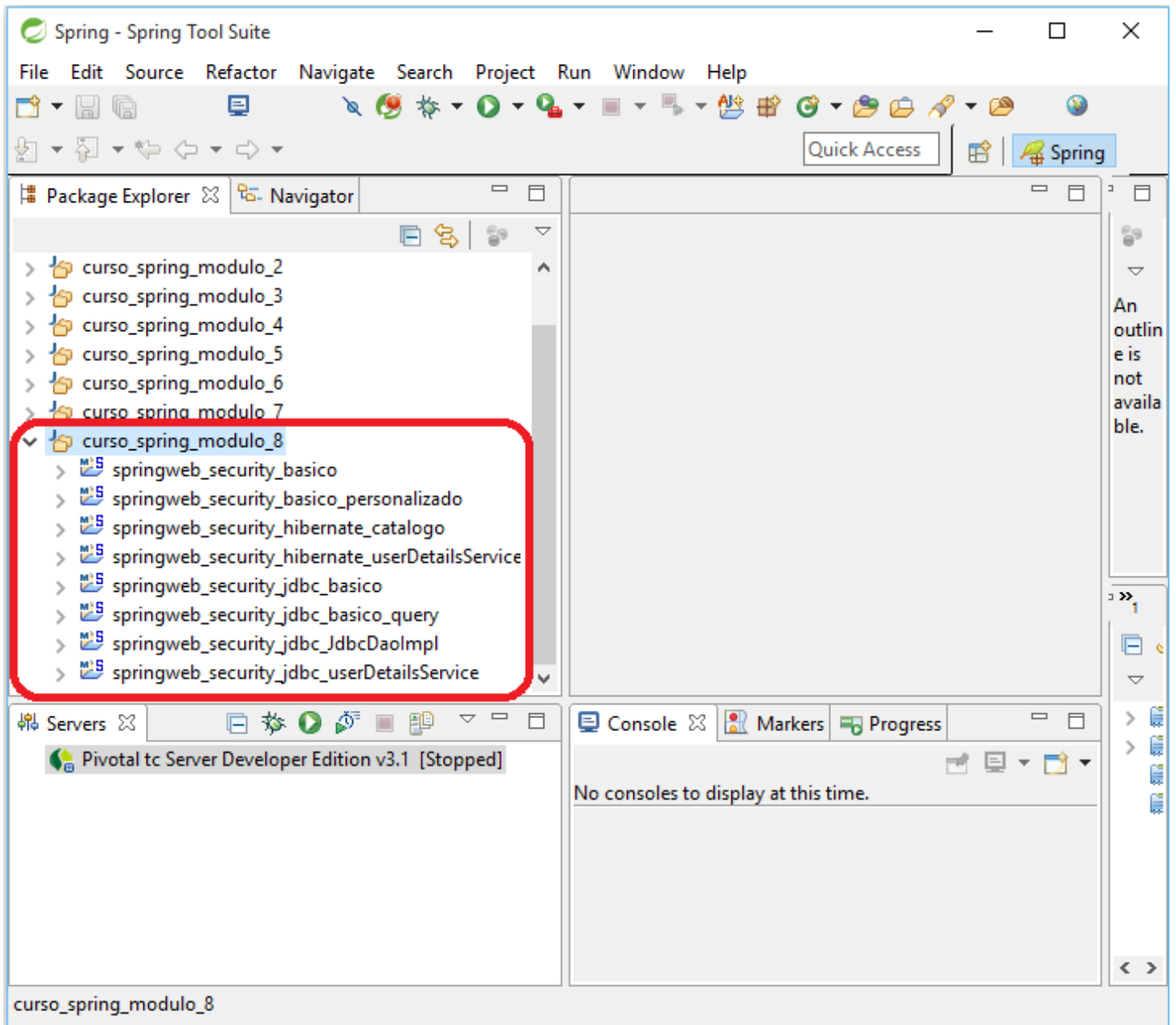




- Clic **Browse**
- Seleccionamos el directorio donde vienen los proyecto de ejemplo del laboratorio
- Agregamos los proyectos al **Working Set** **curso\_spring\_modulo\_8**
- **Finish**





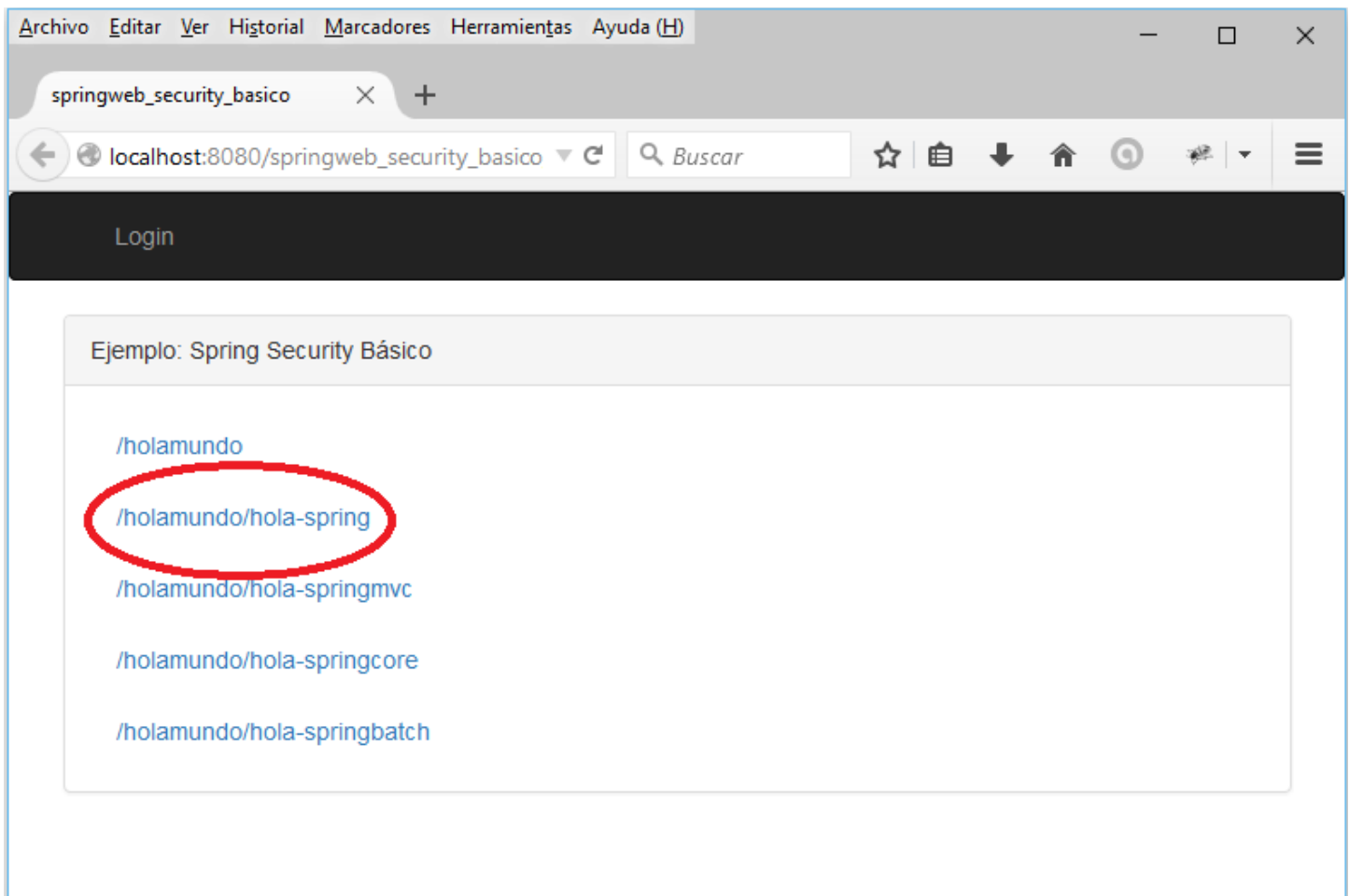


## Ejercicio 1: Implementar seguridad paso a paso a una aplicación Spring MVC

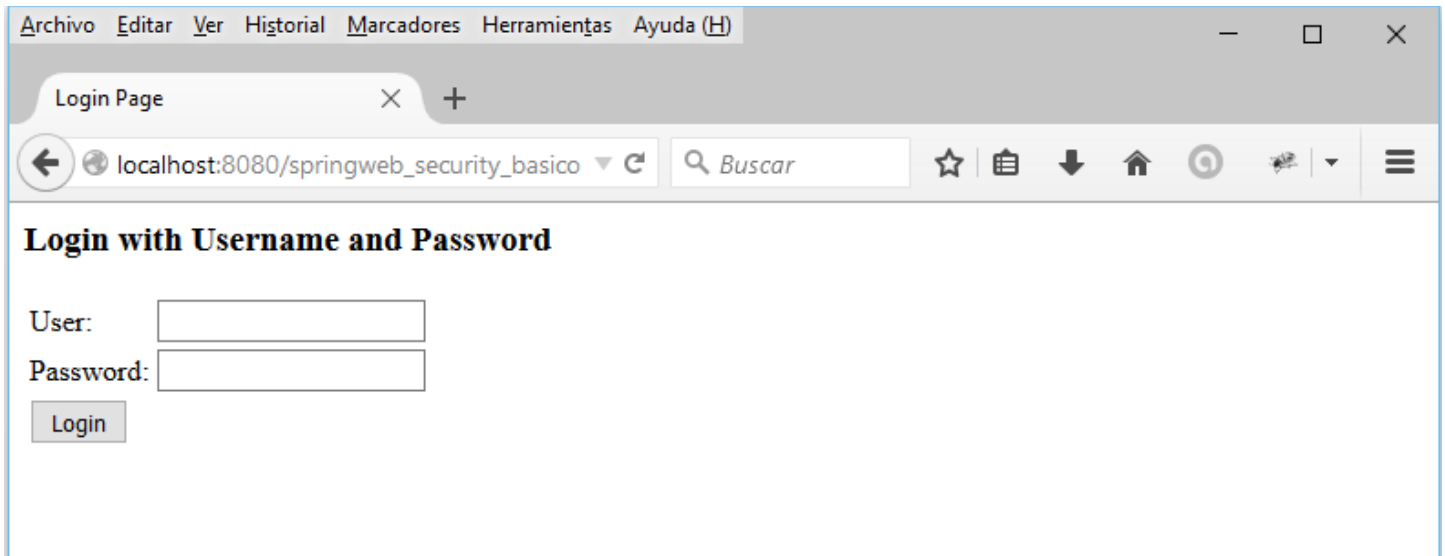
Aprenderemos a dar seguridad a una aplicación web con unos sencillos pasos, para el ejemplo usaremos como base el proyecto del módulo 3 **mvc\_basico\_RequestMapping** que renombraremos a **springweb\_security\_basico**.

También aprenderemos a usar el componente que viene incluido **logout**, que controla el cierre de sesión.

1. Clic derecho sobre el proyecto **Run As->Maven Clean** y **Run As->Maven Install**.
2. Clic derecho sobre el proyecto y **Maven->Update Project...**
3. Clic derecho sobre el proyecto **springweb\_security\_basico-> Run As on Server**
4. Observe el resultado en el navegador:
5. Clic para acceder.



- Observe la página de login (credenciales de seguridad.)



Archivo Editar Ver Historial Marcadores Herramientas Ayuda (H)

Login Page

localhost:8080/springweb\_security\_basico

Buscar

### Login with Username and Password

User:

Password:

Login

Ahora estudiemos el código para que las páginas tengas seguridad y sólo puedan ingresar las personas que tengan las credenciales adecuadas.

6. Observar el archivo pom.xml

ETC ...

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>${spring.version}</version>
</dependency>

<!-- Spring Security -->
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-core</artifactId>
  <version>${spring.security.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-web</artifactId>
  <version>${spring.security.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-config</artifactId>
  <version>${spring.security.version}</version>
</dependency>
```

ETC ...

7. Observar `/springweb_security_basico/src/main/webapp/WEB-INF/web.xml`

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-
app_3_1.xsd" version="3.1">
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            /WEB-INF/spring/root-context.xml,
            /WEB-INF/spring/applicationContext-security.xml
        </param-value>
    </context-param>
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
    <servlet>
        <servlet-name>springweb_security_basico</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>springweb_security_basico</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <!-- Spring Security -->
    <filter>
        <filter-name>springSecurityFilterChain</filter-name>
        <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-
class>
    </filter>

    <filter-mapping>
        <filter-name>springSecurityFilterChain</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>

```

## 8. Estudiar el archivo de configuración de seguridad configurado en el web.xml

### /WEB-INF/spring/applicationContext-security.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/security"
  xmlns:beans="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/security
    http://www.springframework.org/schema/security/spring-security.xsd">

  <!-- Podemos usar multiples elementos <intercept-url> para definir los diferentes
    requerimientos de accesos para el conjunto de URLs, pero serán evaluadas
    en el orden de la lista y a la primera coincidencia será usada. -->
  <http auto-config="true">
    <intercept-url pattern="/holamundo/hola*" access="hasRole('ROLE_SUPERVISOR')" />
    <intercept-url pattern="/holamundo*" access="hasRole('ROLE_USER')" />
  </http>

  <authentication-manager>
    <authentication-provider>
      <user-service>
        <user name="rod" password="rod"
          authorities="ROLE_SUPERVISOR, ROLE_USER, ROLE_TELLER" />
        <user name="bruce" password="bruce" authorities="ROLE_USER,ROLE_TELLER" />
        <user name="james" password="james" authorities="ROLE_USER" />
        <user name="andres" password="andres" authorities="ROLE_USER" />
      </user-service>
    </authentication-provider>
  </authentication-manager>

</beans:beans>
```

El **auto-config = "true"** es sólo una sintaxis abreviada de:

```
<http>
  <form-login />
  <http-basic />
  <logout />
</http>
```

Que permite generar un formulario y autenticación básica con Spring.

## 9. Estudiar vista mivista1.jsp

ETC...

```

<div class="container">
  <div class="alert alert-success" role="alert">
    <p>${mensaje1}</p>
  </div>

  <form action="${pageContext.request.contextPath}/logout" method="post">
    <input class="btn btn-primary" role="button" type="submit" value="Log out" />
    <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />
  </form>

</div>
</body>
ETC...

```

- Por defecto viene habilitada la protección **Cross Site Request Forgery (CSRF)**
- Al tenerla habilitada, necesitamos incluir un campo oculto **\_csrf.token** en cada formulario que tengamos en nuestra aplicación, sobre todo en los formularios de login y logout.

## 10. Estudiar vista mivista2.jsp

ETC...

```

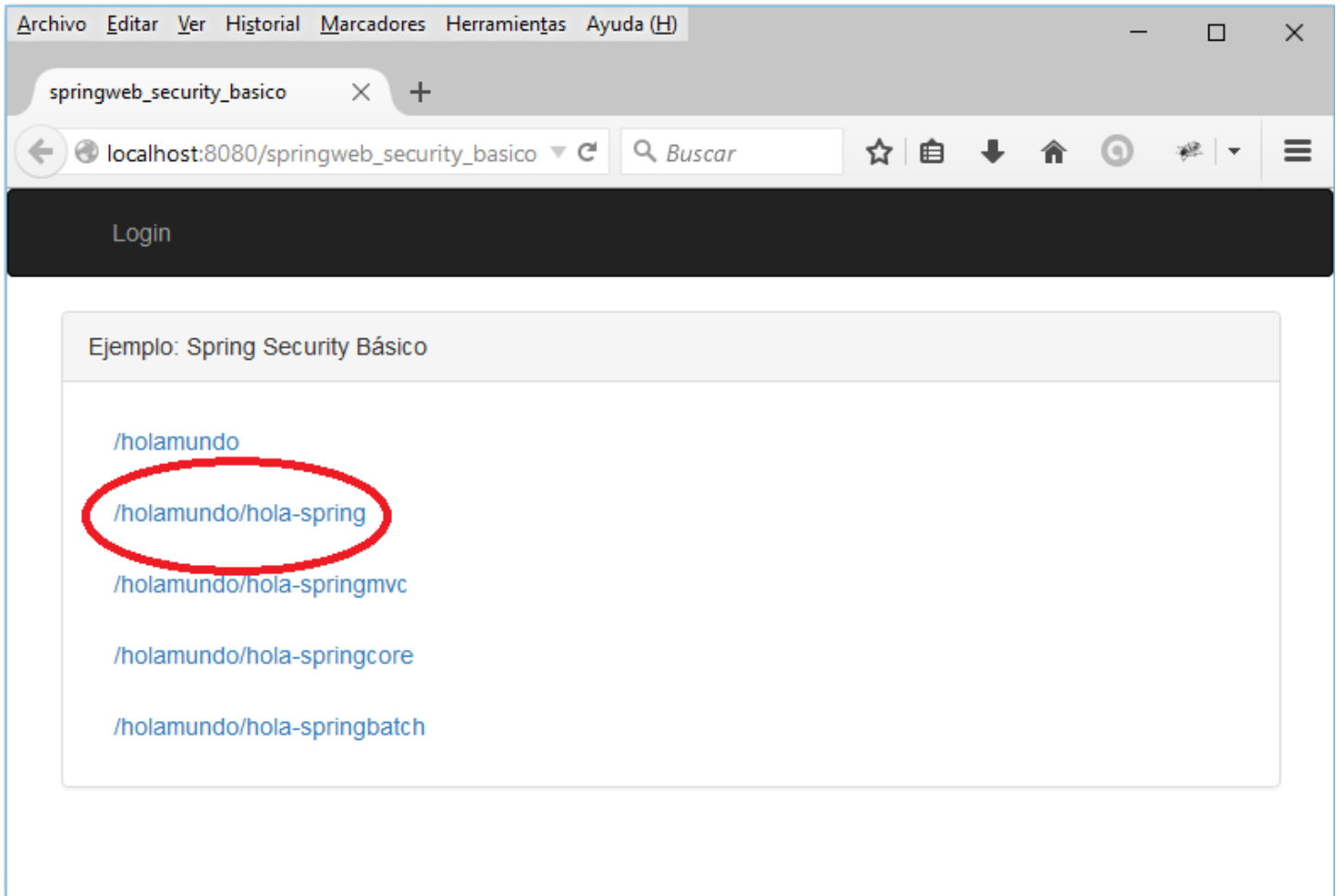
<div class="container">
  <div class="alert alert-success" role="alert">
    <p>${mensaje2}</p>
  </div>

  <form action="${pageContext.request.contextPath}/logout" method="post">
    <input class="btn btn-primary" role="button" type="submit" value="Log out" />
    <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />
  </form>

</div>
</body>
ETC...

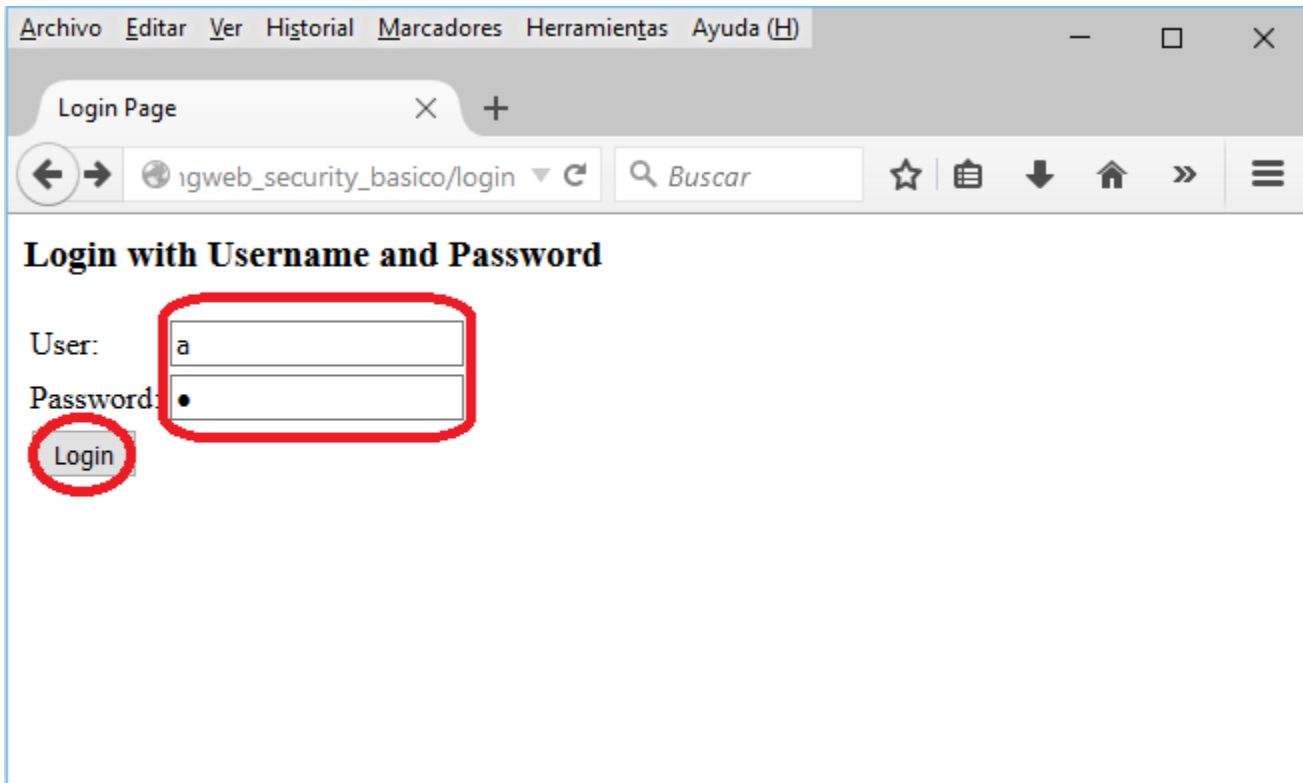
```

11. Accedemos a la página que requiere el rol **ROLE\_SUPERVISOR**



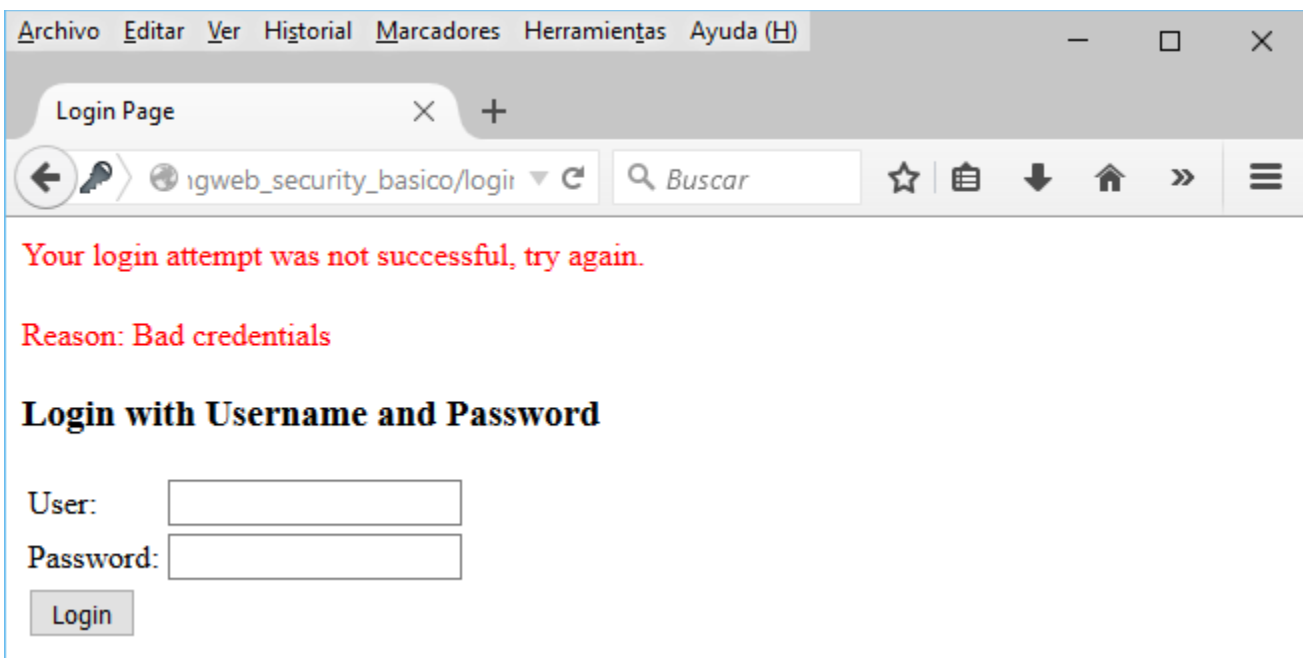


12. Ingresar un user y password incorrecto.



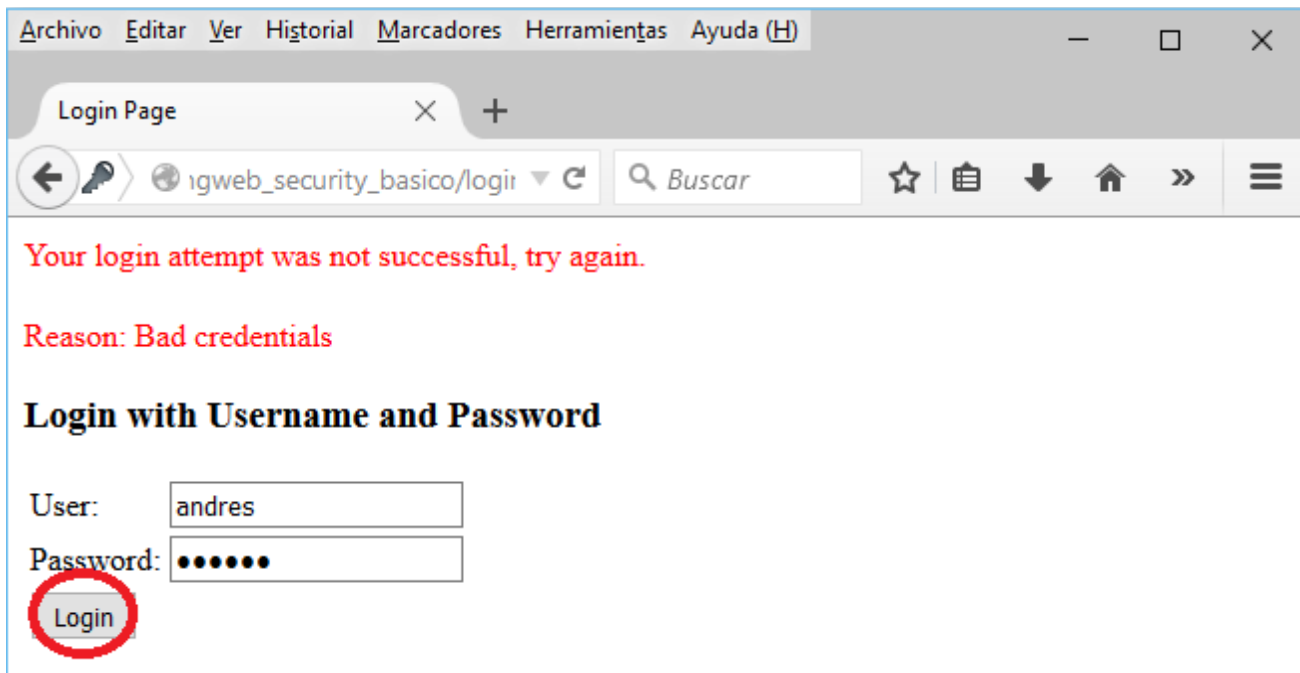
The screenshot shows a web browser window with the title "Login Page". The address bar displays "igweb\_security\_basico/login". The page content includes the heading "Login with Username and Password". Below this, there are two input fields: "User:" containing the letter "a" and "Password:" containing a single dot. A red rectangle highlights both input fields and the "Login" button below them. The "Login" button is also circled in red.

13. Observe el mensaje de error: **Bad credentials**



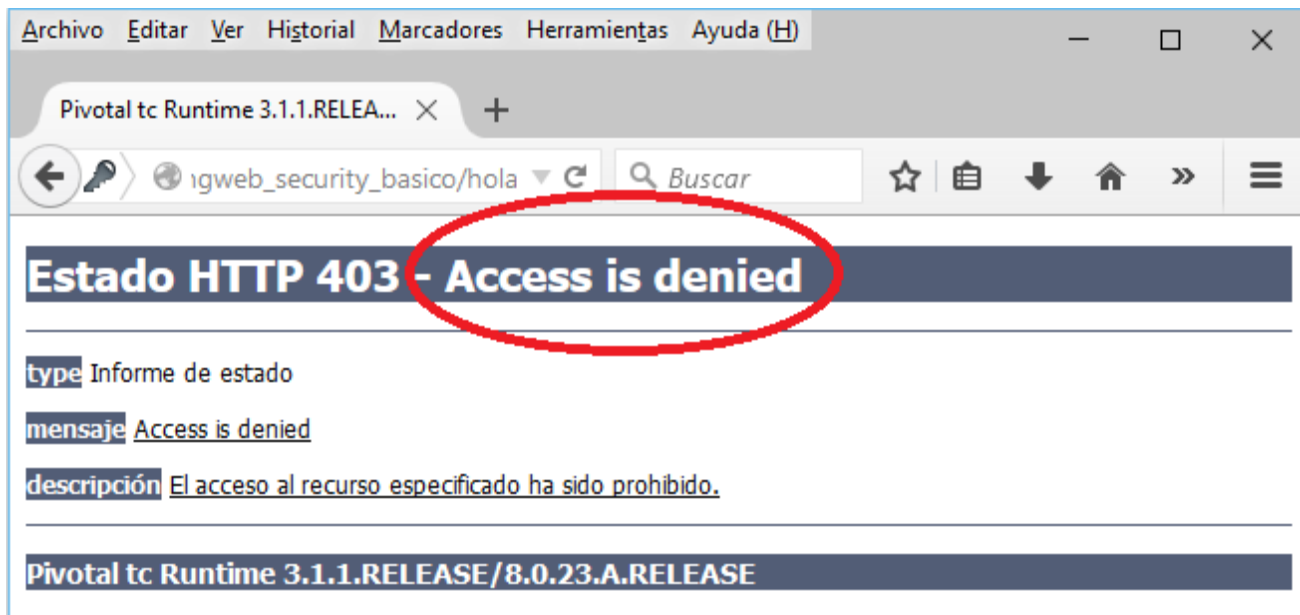
The screenshot shows the same web browser window after a failed login attempt. The page displays a red error message: "Your login attempt was not successful, try again." Below this, the reason for the failure is stated in red: "Reason: Bad credentials". The heading "Login with Username and Password" is still present, followed by empty "User:" and "Password:" input fields and a "Login" button.

14. Acceda con **andres/andres**, quien tiene credenciales de **ROLE\_USER** y NO **ROLE\_SUPERVISOR**.



The screenshot shows a web browser window with the address bar displaying `igweb_security_basico/login`. The page content includes a red error message: "Your login attempt was not successful, try again." Below this, it states "Reason: Bad credentials". The login form is titled "Login with Username and Password" and contains two input fields: "User:" with the value "andres" and "Password:" with masked characters. A "Login" button is located below the password field and is circled in red.

15. Observe que aparece el error: **HTTP 403 - Access is denied**.



The screenshot shows a web browser window with the address bar displaying `igweb_security_basico/hola`. The page content displays a large error message: "Estado HTTP 403 - Access is denied", which is circled in red. Below this, there is a table with the following information:

type	Informe de estado
mensaje	Access is denied
descripción	El acceso al recurso especificado ha sido prohibido.

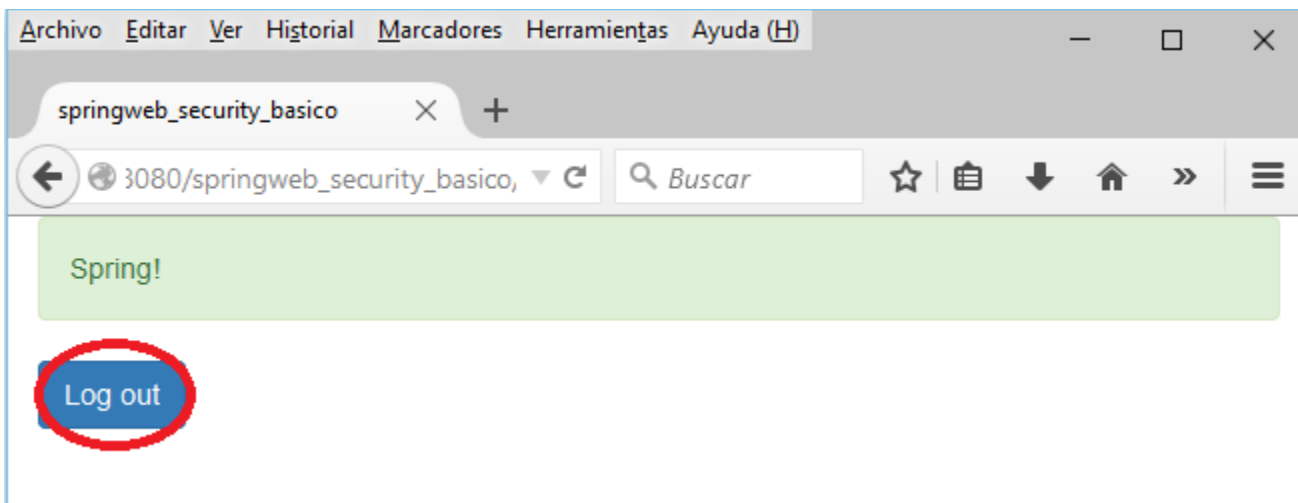
At the bottom of the page, there is a footer that reads: "Pivotal tc Runtime 3.1.1.RELEASE/8.0.23.A.RELEASE".

16. Acceda a la página con **rod/rod**. Quien tiene el privilegio de **ROLE\_SUPERVISOR**.



17. Ahora la página es accesible.

18. Logout.

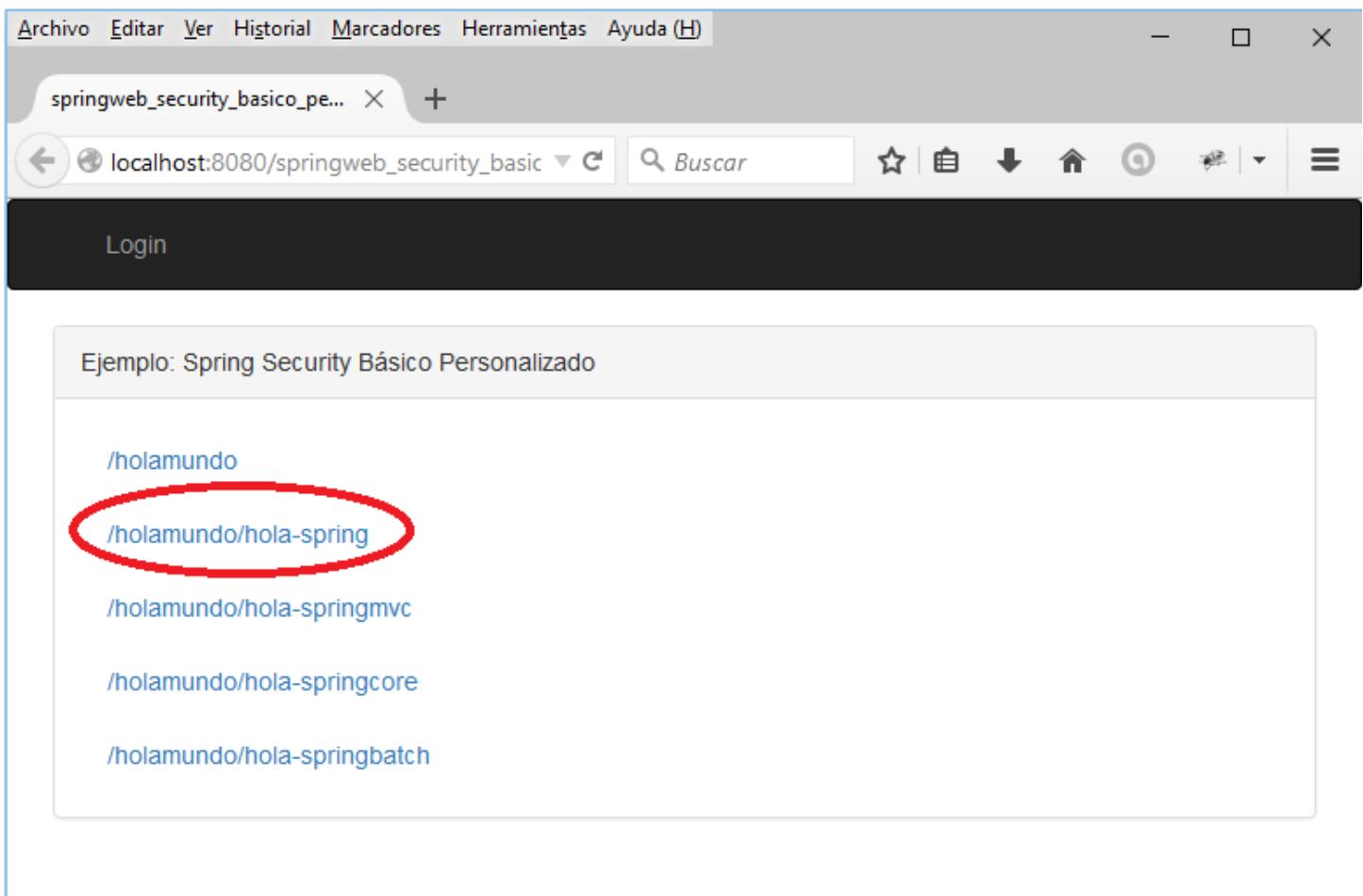


## Ejercicio 2: Customizar páginas Login y Logout

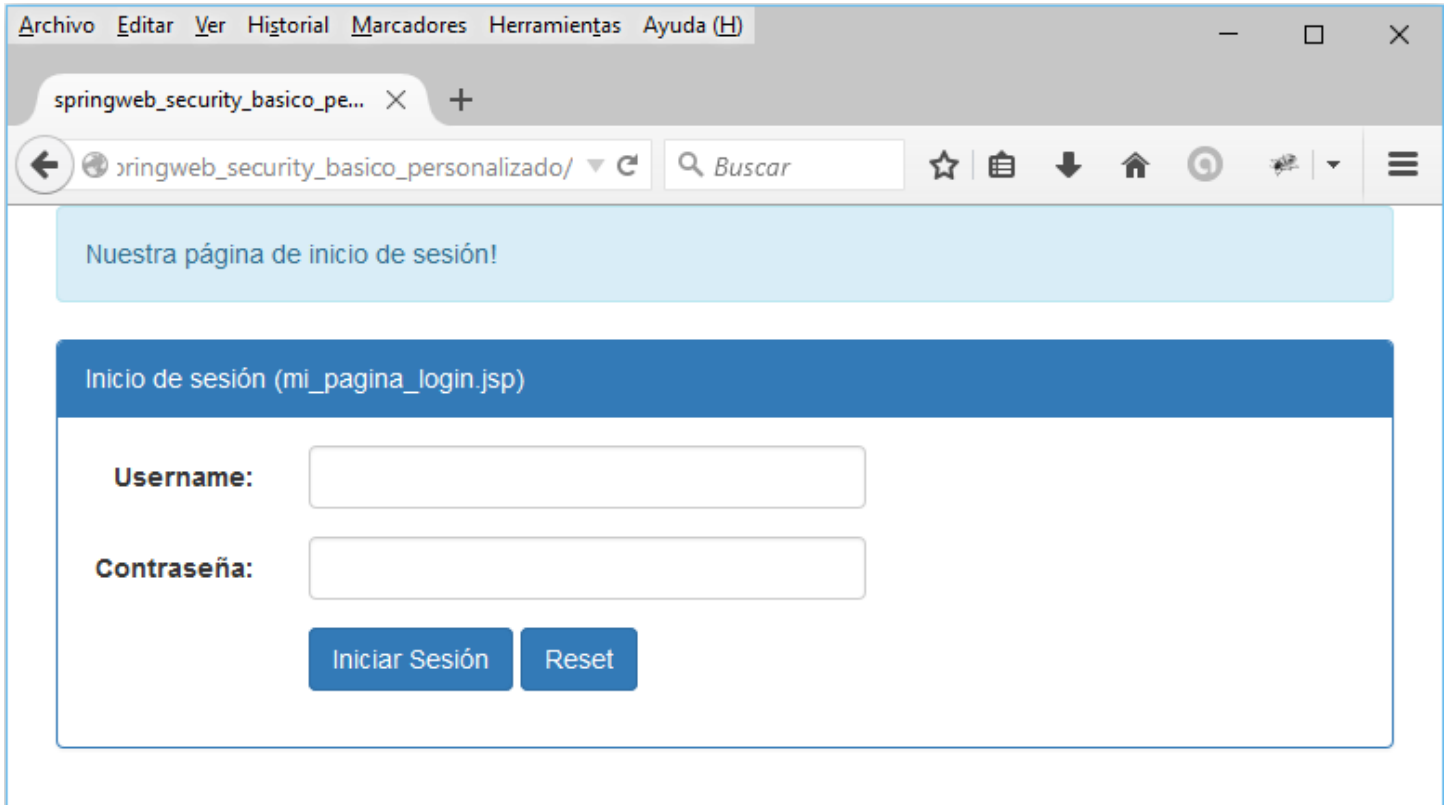
En este ejercicio, vamos a personalizar nuestro formulario de login, creando nuestra propia página de inicio y cierre de sesión, y también de errores. Necesitaremos configurar lo siguiente:

- ✓ **login-page= "/mi\_pagina\_login"** - usamos nuestra propia vista de login.
- ✓ **authentication-failure-url="/mi\_pagina\_errorlogin"** - Si falla la autenticación, cargará la vista mi\_pagina\_errorlogin.
- ✓ **logout-success-url="/mi\_pagina\_logout"** - Después de haber cerrado con éxito la sesión, cargará la vista mi\_pagina\_logout.

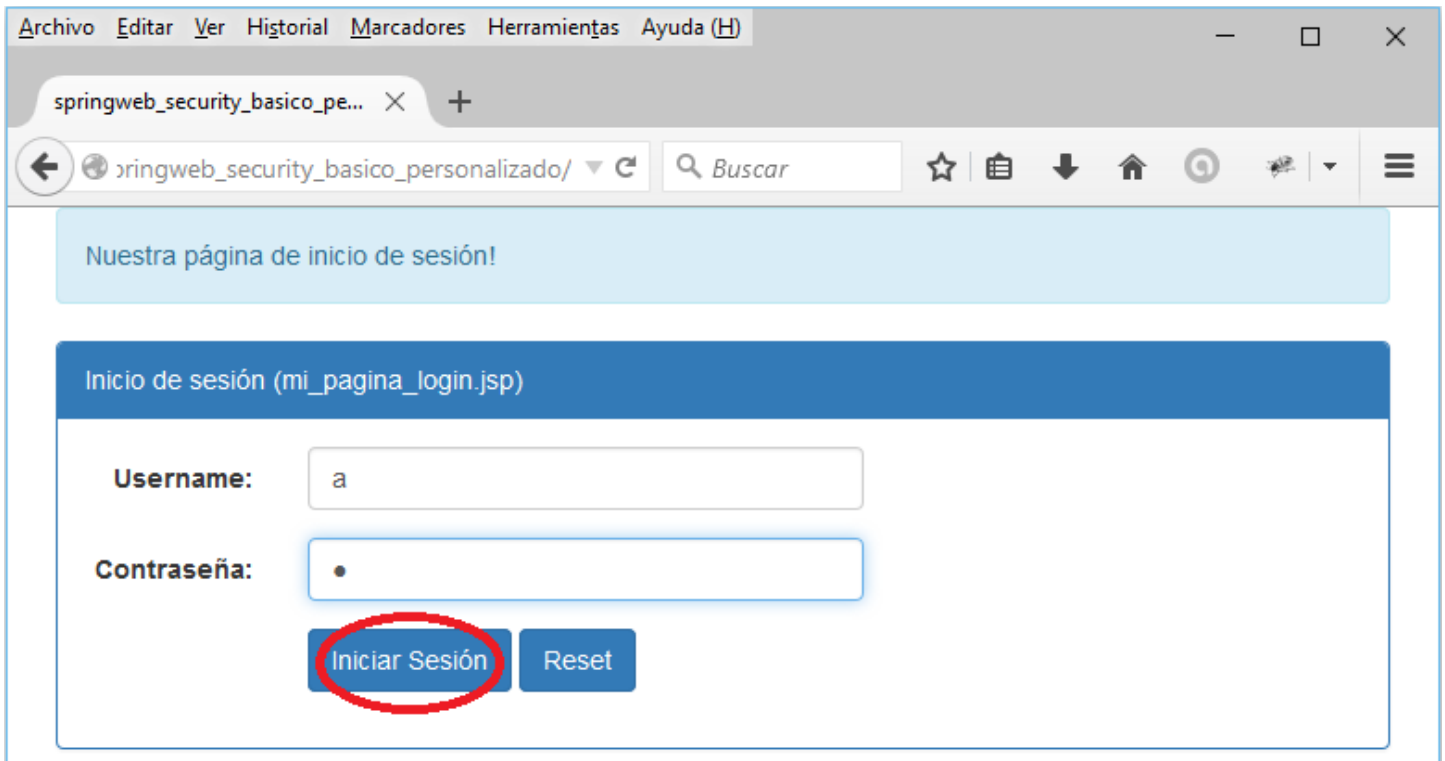
1. Clic derecho sobre el proyecto **Run As->Maven Clean** y **Run As->Maven Install**.
2. Clic derecho sobre el proyecto y **Maven->Update Project...**
3. Clic derecho sobre **springweb\_security\_basico\_personalizado-> Run As on Server**
4. Observe el resultado en el navegador:
5. Acceder a la página con privilegio **ROLE\_SUPERVISOR**.



- Observe que aparece la página personalizada de inicio de sesión (mi\_pagina\_login.jsp) (en lugar de la página por defecto de inicio de sesión de Spring).



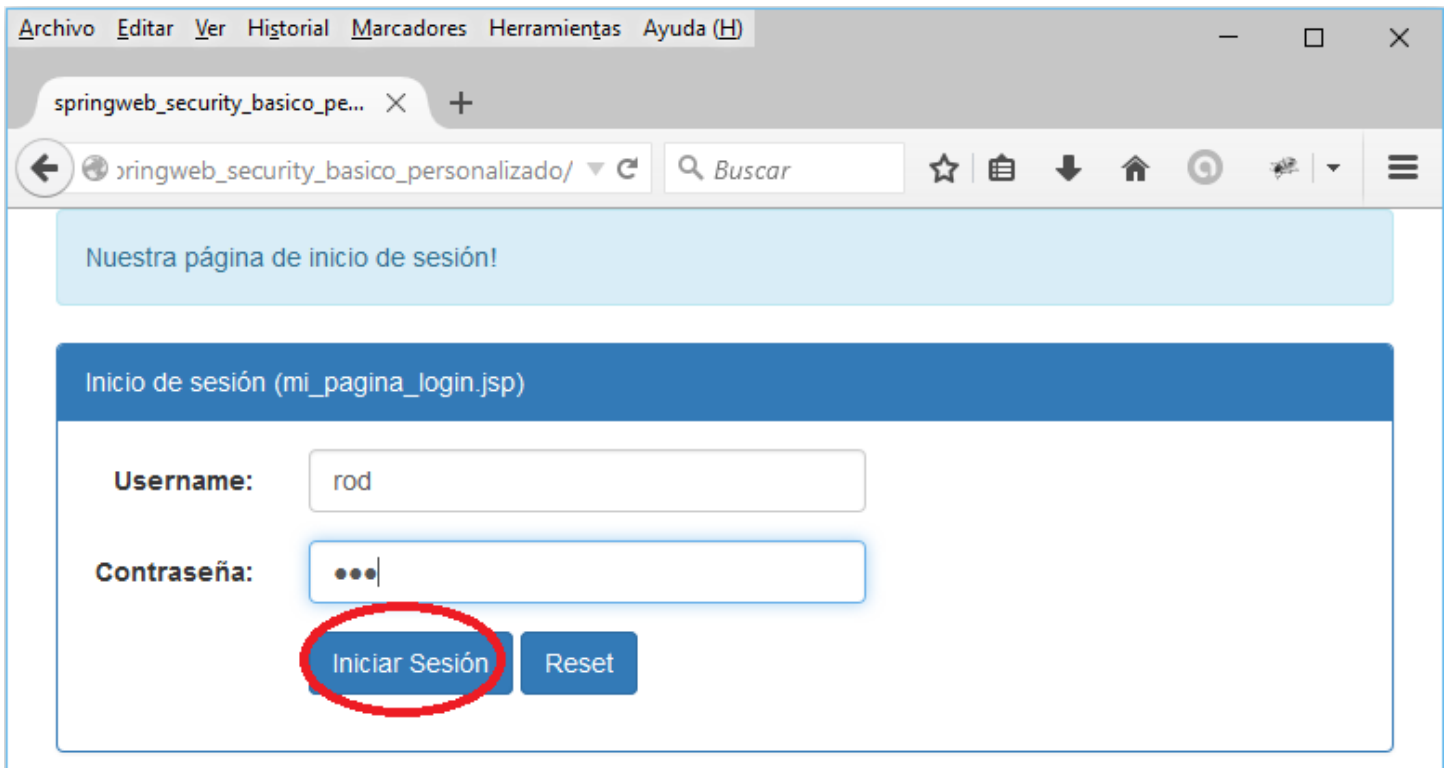
- Ingrese un incorrecto username/password



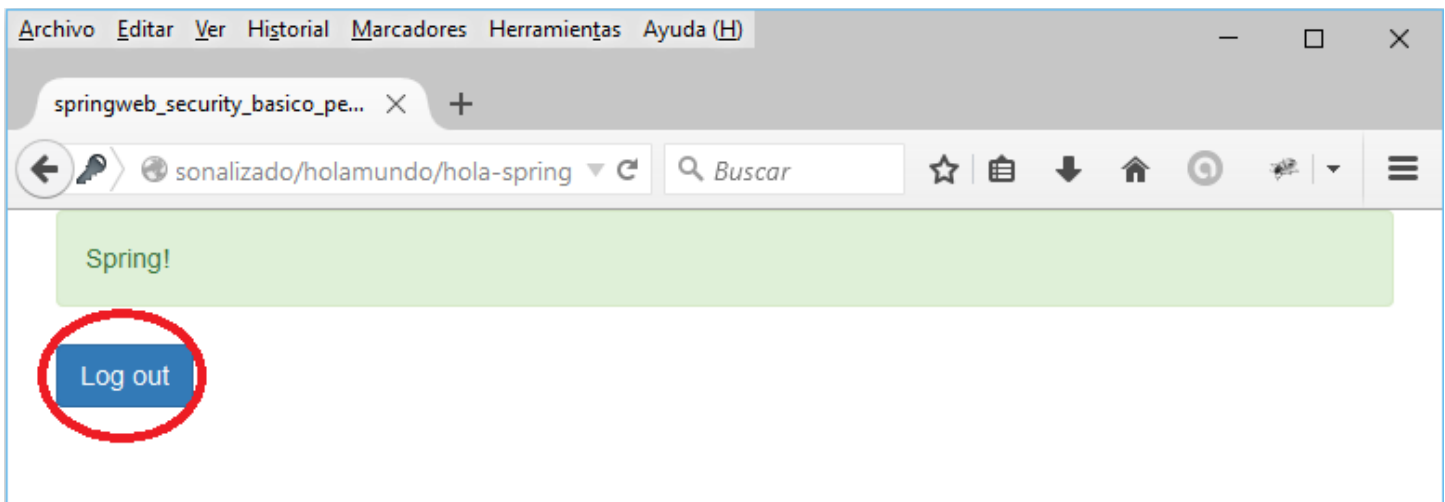
- Observe que aparece la página personalizada de error en la autenticación (mi\_pagina\_errorlogin.jsp).



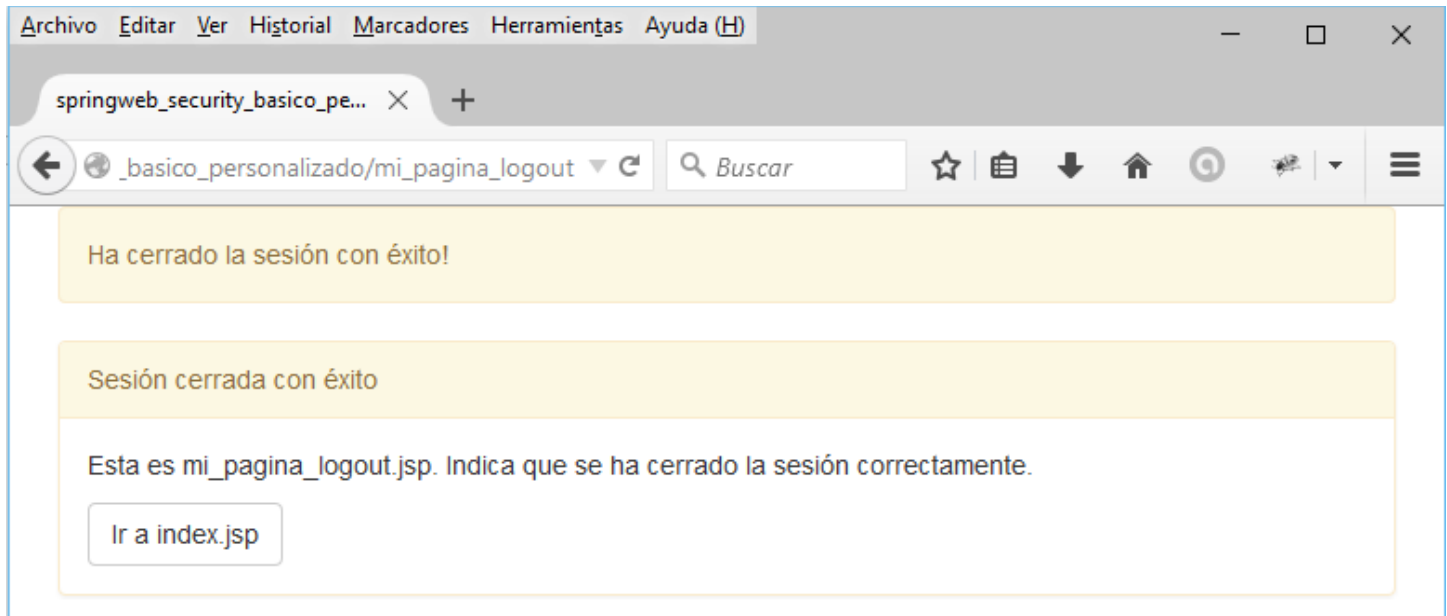
- Ingresar un correcto username/password, con **rod/rod** que tiene el acceso del privilegio **ROLE\_SUPERVISOR**.



- Observe que ahora podemos acceder a la página.



- Observe que aparece la página personalizada cierre de sesión





## 6. Estudiar cambios en el archivo **applicationContext-security.xml**

- Ahora vamos a configurar la página de login, logout y la página de falla de sesión.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/security"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/security
        http://www.springframework.org/schema/security/spring-security.xsd">

    <http auto-config="true">
        <access-denied-handler error-page="/mi_pagina_error_403" />
        <intercept-url pattern="/holamundo/hola*" access="hasRole('ROLE_SUPERVISOR')"/>
        <intercept-url pattern="/holamundo*" access="hasRole('ROLE_USER')"/>
        <form-login login-page="/mi_pagina_login"
            default-target-url="/mi_pagina_despues_login"
            authentication-failure-url="/mi_pagina_errorlogin" />
        <logout logout-success-url="/mi_pagina_logout" />
    </http>

    <authentication-manager>
        <authentication-provider>
            <user-service>
                <user name="rod" password="rod"
                    authorities="ROLE_SUPERVISOR, ROLE_USER, ROLE_TELLER" />
                <user name="bruce" password="bruce" authorities="ROLE_USER,ROLE_TELLER" />
                <user name="james" password="james" authorities="ROLE_USER" />
                <user name="andres" password="andres" authorities="ROLE_USER" />
            </user-service>
        </authentication-provider>
    </authentication-manager>
</beans:beans>
```

7. Observemos nuestra vista JSP `mi_pagina_login`.`/src/main/webapp/WEB-INF/views/mi_pagina_login.jsp`

```
... ETC
<c:if test="${not empty error}">
  <div class="alert alert-info">
    <p>Su intento de inicio de sesión ha fallado, vuelva a intentarlo!</p>
    <p>Causa: ${sessionScope["SPRING_SECURITY_LAST_EXCEPTION"].message}</p>
  </div>
</c:if>
... ETC
<form name='f' action="${pageContext.request.contextPath}/login"
  method='post' class="form-horizontal" role="form">
  <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />
  <div class="form-group">
    <label for="username" class="col-sm-2 control-label">Username:</label>
    <div class="col-sm-10">
      <input style="width: 300px;" class="form-control" type='text' name='username' />
    </div>
  </div>

  <div class="form-group">
    <label for="password" class="col-sm-2 control-label">Contraseña:</label>
    <div class="col-sm-10">
      <input style="width: 300px;" class="form-control" type='password' name='password' />
    </div>
  </div>

  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <input type="submit" value="Iniciar Sesión" class="btn btn-primary" role="button" />
      <input type="reset" value="Reset" class="btn btn-primary" role="button" />
    </div>
  </div>
</form>
... ETC
```

8. Observemos nuestra vista JSP `mi_pagina_errorlogin`.`/src/main/webapp/WEB-INF/views/mi_pagina_errorlogin.jsp`

```
... ETC
<body>
    <div class="container">
        <div class="alert alert-danger">
            <p>El login ha fallado!.</p>
        </div>

        <div class="panel panel-danger">
            <div class="panel-heading">Página de error</div>
            <div class="panel-body">
                <p>Esta es mi_pagina_errorlogin.jsp. Indica que el login ha fallado.</p>
                <a class="btn btn-default" href="${pageContext.request.contextPath}/index.jsp"
                    role="button">Ir a index.jsp</a>
            </div>
        </div>
    </div>
</body>
... ETC
```

9. Observemos nuestra vista JSP `mi_pagina_logout`.`/src/main/webapp/WEB-INF/views/mi_pagina_login.jsp`

```
... ETC
<body>
    <div class="container">
        <div class="alert alert-warning">
            <p>Ha cerrado la sesión con éxito!</p>
        </div>

        <div class="panel panel-warning">
            <div class="panel-heading">Sesión cerrada con éxito</div>
            <div class="panel-body">
                <p>Esta es mi_pagina_logout.jsp. Indica que se ha cerrado la
                    sesión correctamente.</p>
                <a class="btn btn-default" href="${pageContext.request.contextPath}/index.jsp"
                    role="button">Ir a index.jsp</a>
            </div>
        </div>
    </div>
</body>
... ETC
```

10. Luego observemos los mapping URL para las vistas jsp, las definimos en **servelt-context.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="
           http://www.springframework.org/schema/mvc
           http://www.springframework.org/schema/mvc/spring-mvc.xsd
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/context
           http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- Escanea o busca en el package base de la aplicación clases beans anotados
         con @Components, @Controller, @Service -->
    <context:component-scan base-package="com.bolsadeideas.ejemplos" />

    <!-- Habilita la anotacion de Spring MVC @Controller -->
    <mvc:annotation-driven />

    <!-- Declaramos el mapeo de las URL para cargar las vistas inmediatamente
         (sin necesidad de un controller) -->
    <mvc:view-controller path="/mi_pagina_login" view-name="mi_pagina_login" />
    <mvc:view-controller path="/mi_pagina_logout" view-name="mi_pagina_logout" />
    <mvc:view-controller path="/mi_pagina_errorlogin" view-name="mi_pagina_errorlogin" />
    <mvc:view-controller path="/mi_pagina_despues_login" view-name="mi_pagina_despues_login" />
    <mvc:view-controller path="/mi_pagina_error_403" view-name="mi_pagina_error_403" />

    <!-- View Resolvers -->
    <!-- Resuelve la ubicion de las vistas .jsp de @Controllers en la ruta /WEB-INF/views -->
    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
    </bean>
</beans>
```

### Ejercicio 3: Customizar login segunda parte.

En este ejercicio, vamos a personalizar nuestra página propia después de haber iniciado login, es útil si nuestra aplicación requiere que siempre el usuario inicie en una página de inicio por defecto.

Necesitaremos configurar lo siguiente:

- **default-target-url="/mi\_pagina\_despues\_login"** - usamos esta configuración para tener una página después de haber iniciado sesión correctamente.
- **always-use-default-target='true'** - usamos esta opción para que siempre el usuario termine en esta página después del login, independientemente si el login fue realizado en forma implícita o explícitamente.

#### 1. Modificar el archivo **applicationContext-security.xml**

- Ahora vamos a configurar la página de login, logout y la página de falla de sesión.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/security"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/security
        http://www.springframework.org/schema/security/spring-security.xsd">

    <http auto-config="true">
        <access-denied-handler error-page="/mi_pagina_error_403" />
        <intercept-url pattern="/holamundo/hola*" access="hasRole('ROLE_SUPERVISOR')"/>
        <intercept-url pattern="/holamundo*" access="hasRole('ROLE_USER')"/>
        <form-login login-page="/mi_pagina_login"
            default-target-url="/mi_pagina_despues_login"
            always-use-default-target='true'
            authentication-failure-url="/mi_pagina_errorlogin" />
        <logout logout-success-url="/mi_pagina_logout" />
    </http>

    <authentication-manager>
        <authentication-provider>
            <user-service>
                <user name="rod" password="rod"
                    authorities="ROLE_SUPERVISOR, ROLE_USER, ROLE_TELLER" />
                <user name="bruce" password="bruce" authorities="ROLE_USER,ROLE_TELLER" />
                <user name="james" password="james" authorities="ROLE_USER" />
                <user name="andres" password="andres" authorities="ROLE_USER" />
            </user-service>
        </authentication-provider>
    </authentication-manager>
</beans:beans>
```

2. Observamos nuestra vista JSP `mi_pagina_despues_login`.

`/src/main/webapp/WEB-INF/views/mi_pagina_despues_login.jsp`

```
<body>
  <div class="container">
    <div class="alert alert-success">
      <p>Ha iniciado sesión con éxito!</p>
    </div>

    <div class="panel panel-success">
      <div class="panel-heading">Primera página después del login</div>
      <div class="panel-body">
        <p>Esto es la primera página justo después del login de inicio
          de sesión (mi_pagina_despues_login.jsp). Todo el mundo después de
          iniciar sesión, verá esta página.</p>
        <a class="btn btn-default" href="${pageContext.request.contextPath}/index.jsp"
          role="button">Volver
          al index.jsp</a>
        <form action="${pageContext.request.contextPath}/logout" method="post">
          <input class="btn btn-warning" role="button" type="submit"
            value="Log out" />
          <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />
        </form>
      </div>
    </div>
  </div>
</body>
```

### 3. Observamos el mapping request URL en **servelt-context.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="
           http://www.springframework.org/schema/mvc
           http://www.springframework.org/schema/mvc/spring-mvc.xsd
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/context
           http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- Escanea o busca en el package base de la aplicación clases beans anotados
         con @Components, @Controller, @Service -->
    <context:component-scan base-package="com.bolsadeideas.ejemplos" />

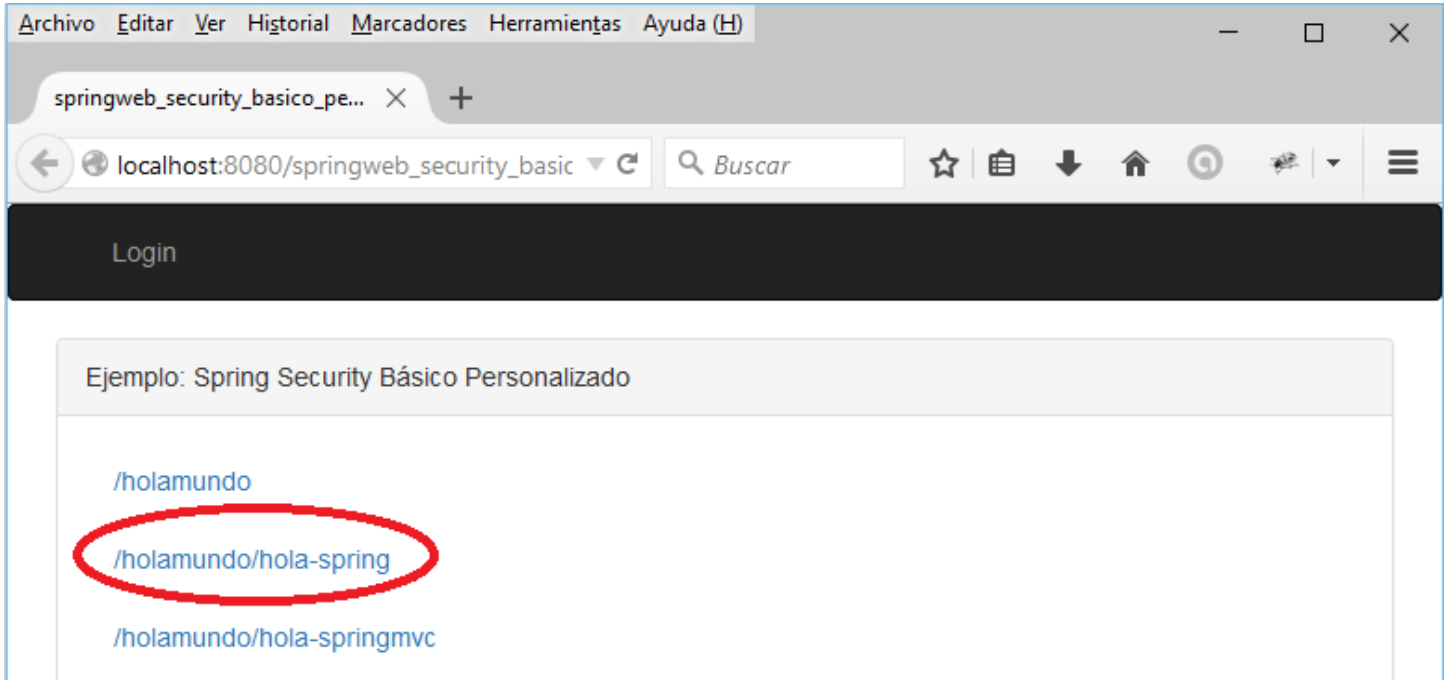
    <!-- Habilita la anotacion de Spring MVC @Controller -->
    <mvc:annotation-driven />

    <!-- Declaramos el mapeo de las URL para cargar las vistas inmediatamente
         (sin necesidad de un controller) -->
    <mvc:view-controller path="/mi_pagina_login" view-name="mi_pagina_login" />
    <mvc:view-controller path="/mi_pagina_logout" view-name="mi_pagina_logout" />
    <mvc:view-controller path="/mi_pagina_errorlogin" view-name="mi_pagina_errorlogin" />
    <mvc:view-controller path="/mi_pagina_despues_login" view-name="mi_pagina_despues_login" />
    <mvc:view-controller path="/mi_pagina_error_403" view-name="mi_pagina_error_403" />

    <!-- View Resolvers -->
    <!-- Resuelve la ubicion de las vistas .jsp de @Controllers en la ruta /WEB-INF/views -->
    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
    </bean>
</beans>
```

## 4. Re-Construir y ejecutar la aplicación

- Observe el resultado en el navegador:
- Acceder a la página que tiene acceso con privilegio ROLE\_USER, links [/holamundo](#).





5. Ingresamos un correcto username/password, con **andres/andres**.

Archivo Editar Ver Historial Marcadores Herramientas Ayuda (H)

springweb\_security\_basico\_pe... X +

localhost:8080/springweb\_security\_basic Buscar

Nuestra página de inicio de sesión!

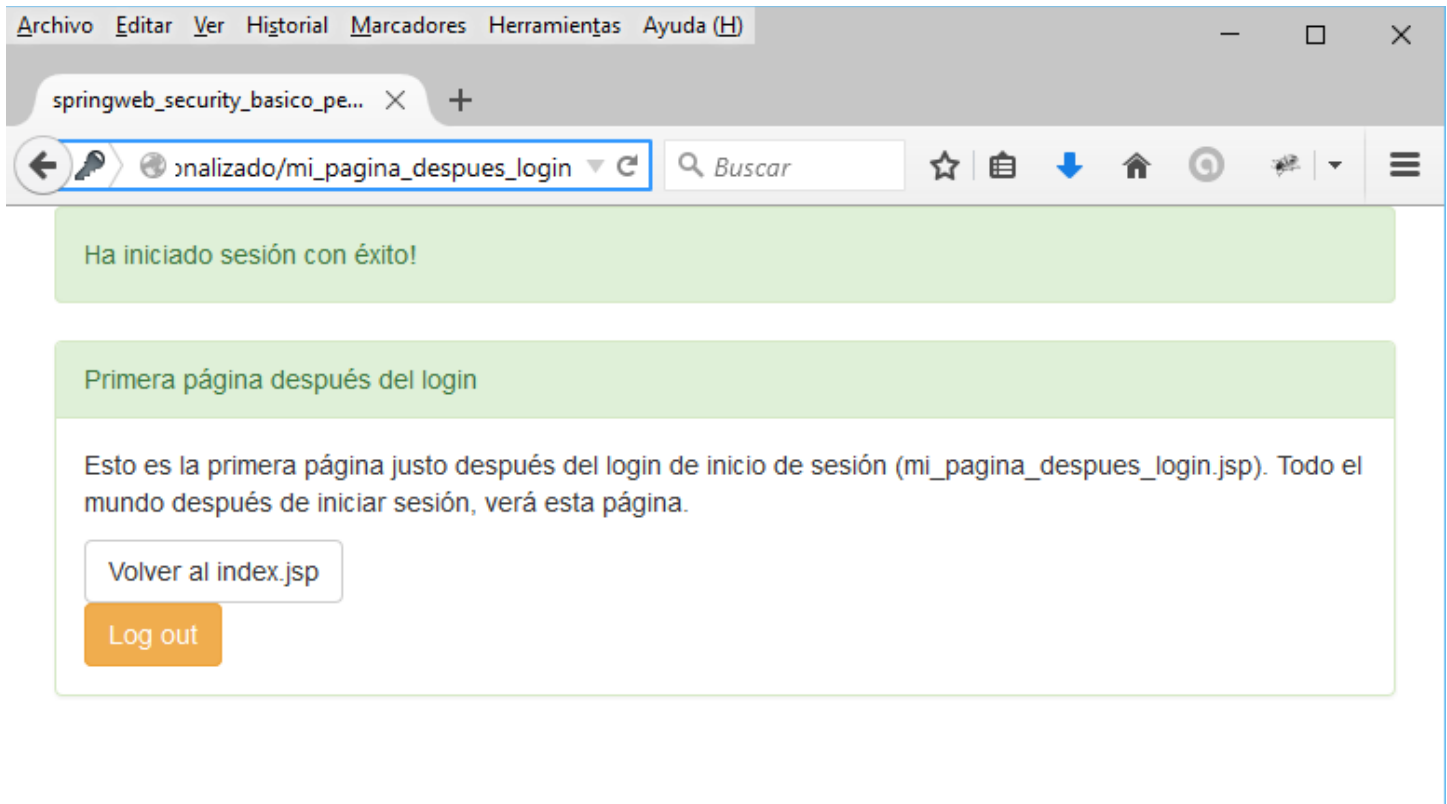
Inicio de sesión (mi\_pagina\_login.jsp)

Username: andres

Contraseña: .....

Iniciar Sesión Reset

6. Observe que ahora redirige a nuestra página de inicio de sesión.



## Resumen

En el documento se explica detalladamente como incorporar un sistema de autenticación de usuarios en nuestros proyectos con Spring MVC. Nos proporciona un sistema de login bastante simple pero potente y con un completo control de errores y configuración de los mensajes de manera sencilla.

¡Dudas, a los foros! ;-)

**FIN.**

**Envía tus consultas a los foros!**

Aquí es cuando debes sacarte todas las dudas haciendo consultas en los foros correspondientes

## Lectura Recomendada y Bibliografía

- [Spring Security](#): recomendable lectura del manual oficial para complementar con este workshop.
- [The Security Namespace](#): Recomendable lectura de esta sección del manual oficial para complementar con este workshop.
- [Sample Applications](#): manual oficial para complementar con este workshop.
- [Spring Security](#) tutorial por Mkyung.com
- [Presentación sobre Spring Security](#)
- [Introduction to Spring Security 3/3.1](#) video tutorial por Mike Wiesne