Estructura del formulario de login personalizado.

```
<form name='form' action='/login' method='POST'>
     Usuario:<br>
     <input type='text' name='username' ><br>
     Password:<br>
     <input type='password' name='password'><br>
     <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}"/>
     <input name="submit" type="submit" value="Ingresar" >
</form>
```

La configuración de color **AZUL** es obligatoria como está (NO ES OPCIONAL).

- **action='/login':** URL del atributo action del formulario por default.
- **method='POST':** Por seguridad, nunca poner GET.
- **name='username':** El nombre del input tiene que ser **username** porque así lo espera Spring Security.
- **name='password':** El nombre del input tiene que ser **password** porque así lo espera Spring Security.
- **type='password':** Para que no sea visible el password al ingresarse.
- **type='hidden':** Input de tipo hidden para incluir el CSRF Token para prevenir los ataques de tipo **C**ross **S**ite **R**equest **F**orgery (el valor del atributo name y value es generado por Spring Security). El CSRF Token es necesario en peticiones tipo POST, DELETE, PUT y PATCH.
- **type="submit":** El botón tiene que ser tipo SUBMIT. No dejarlo por ejemplo type="button".

NOTA: Se puede agregar cualquier código HTML, CSS e IMAGENES para darle formato. En nuestro caso utilizaremos el diseño del archivo formLogin.html que viene en la carpeta de la plantilla del proyecto.

Configurar formulario de login personalizado.

```
<http auto-config="true">
   <!-- Declaramos todos los recursos que estaran protegidos -->
   <intercept-url pattern="/peliculas/*" access="hasAnyAuthority('EDITOR')" />
   <intercept-url pattern="/horarios/*" access="hasAnyAuthority('EDITOR')" />
   <intercept-url pattern="/noticias/*" access="hasAnyAuthority('EDITOR')" />
   <intercept-url pattern="/banners/*" access="hasAnyAuthority('GERENTE')" />

   <!-- Custom login form -->
   <form-login login-page="/formLogin" />

</http>
```

security.xml

```
@RequestMapping(value = "/formLogin" )
public String mostrarLogin() {
    return "formLogin";
}
```

Controller

Archivo JSP (view) con el formulario HTML de login.

Notificación de usuario/contraseña incorrectos (1).

- Por default cuando es proporcionado un usuario/contraseña incorrectos Spring Security realiza un REDIRECCIONAMIENTO a la URL del formulario de login y le pasa un parámetro llamado error. Ejemplo:

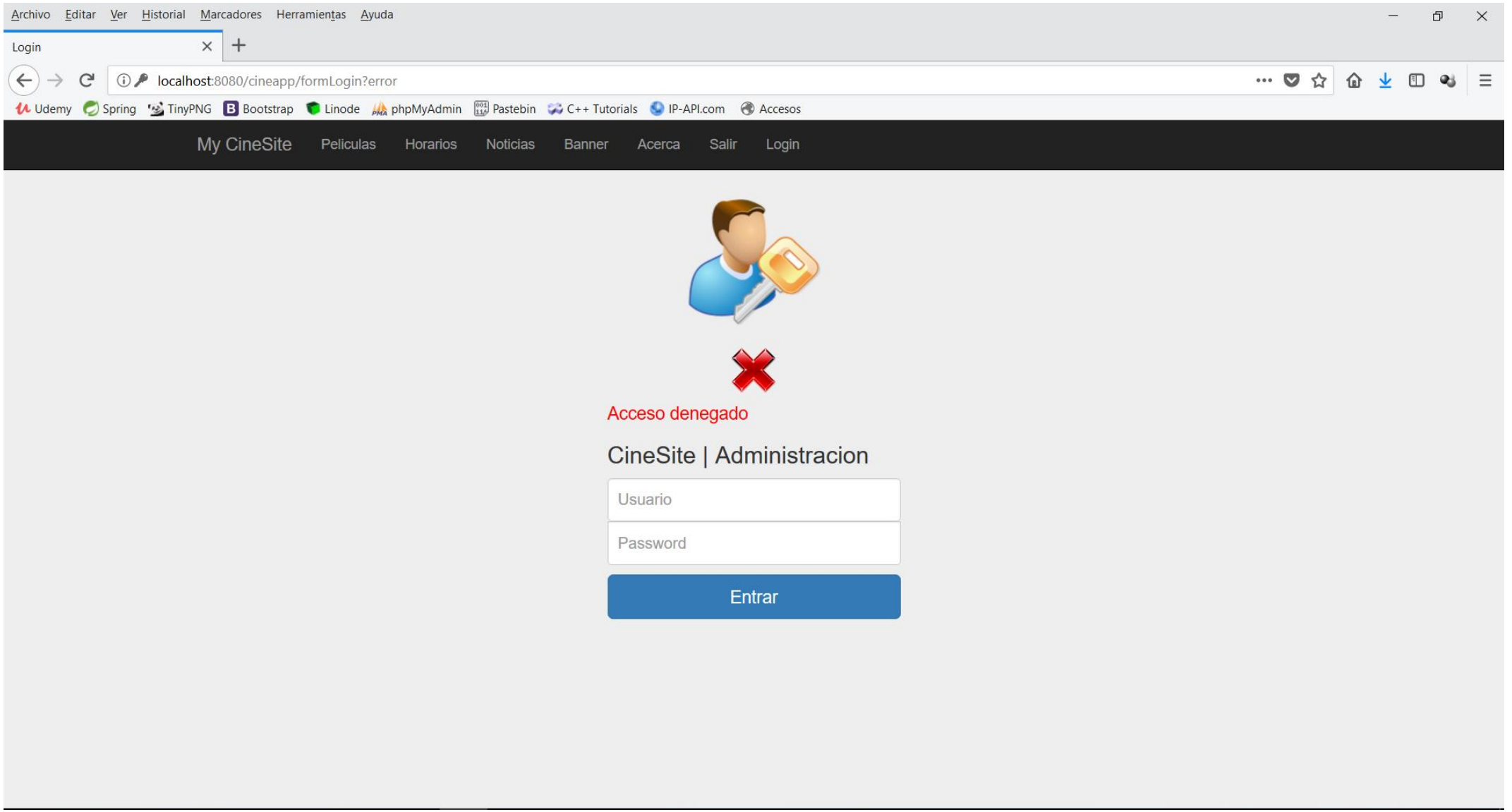
  http://localhost:8080/cineapp/formLogin**?error**

- Este parámetro puede ser usado en la vista del formulario de login para mostrar una notificación al usuario.

```
<form name='form' action='/login' method='POST'>

   <c:if test="${param.error != null}">
      <img src="${urlPublic}/images/error.png" width="48" height="48" class="center">
      <h4 class="form-signin-heading" style="color:red">Acceso denegado</h4>
   </c:if>
   . . .
</form>
```

Notificación de usuario/contraseña incorrectos (2).



Spring Security: JSP Tag Libraries

- Spring Security tiene sus propios Taglibs que pueden ser usados en los archivos JSP para proporcionar soporte básico para aplicar seguridad en la vista.
- Para usar cualquiera de los Tags, debes incluir la siguiente declaración en los archivos JSP:
  - ✓ <%@taglib uri="http://www.springframework.org/security/tags" prefix="sec" %>

El tag authorize

Este tag es utilizado para determinar si parte de la vista será renderizada o no. Un ejemplo clásico es renderizar un menú de opciones, dependiendo del rol del usuario.

```
<sec:authorize access="isAnonymous()">
   <a href="/about">Acerca</a>
   <a href="/formLogin">Login</a>
</sec:authorize>

<sec:authorize access="hasAnyAuthority('EDITOR')">
   <a href="/peliculas/indexPaginate?page=0">Peliculas</a>
   <a href="/horarios/indexPaginate?page=0">Horarios</a>
   <a href="/noticias/index">Noticias</a>
   <a href="/admin/logout">Salir</a>
</sec:authorize>
```

La expresión **isAnonymous()** es tipo boolean y básicamente regresa **true** si un usuario es anónimo (no ha sido autenticado en la aplicación).

La expresión **hasAnyAuthority('EDITOR')** es tipo boolean y básicamente regresa **true** si el usuario autenticado tiene el rol EDITOR asignado.

Configurar la URL destino después de hacer login.

```
<http auto-config="true">
   <!-- Declaramos todos los recursos que estaran protegidos -->
   <intercept-url pattern="/peliculas/*" access="hasAnyAuthority('EDITOR')" />
   <intercept-url pattern="/horarios/*" access="hasAnyAuthority('EDITOR')" />
   <intercept-url pattern="/noticias/*" access="hasAnyAuthority('EDITOR')" />
   <intercept-url pattern="/banners/*" access="hasAnyAuthority('GERENTE')" />

   <!-- Custom login form -->
   <form-login login-page="/formLogin"
       default-target-url="/admin/index"/>

</http>
```

security.xml

```
@GetMapping(value="/admin/index")
public String mostrarPrincipalAdmin() {
     return "admin";
}
```

Controller

Archivo JSP (view) con algún mensaje de bienvenida.