Una aplicación web debe tener un conjunto de aspectos "minimos" para su operación.

Aspecto de manejo de sesión

* Limita el ingreso a recursos protegidos/privados
* Permitir iniciar/cerrar sesión
* Implementa la captura del recurso al que quiero acceder

Pasos:

1. Crear la pantalla de inicio de sesión
2. Principio del formulario

|  |  |
| --- | --- |
| Usuario |  |
| Clave |  |
|  |  |
| He olvidado mi clave | |

1. Final del formulario

**form sin action**

**método es post**

<form **method="post"**>

<table>

<tbody>

<tr>

<td>Usuario</td>

<td><input name="usuario" /></td> **este atributo se utiliza en el aspecto**

</tr>

<tr>

<td>Clave</td>

<td><input type="password" name="clave" /></td> **este atributo se utiliza en el aspecto**

</tr>

<tr>

<td></td>

<td><input type="submit" value="Ingresar"/></td>

</tr>

<tr>

<td colspan="2"><a href="">He olvidado mi clave</a></td>

</tr>

</tbody>

</table>

</form>

1. Crear el servicio de autenticación

public interface LoginService {

boolean authenticate(String user, String password);

}

public class LoginServiceImpl implements LoginService {

public boolean authenticate(String user, String password) {

return user.equals("carlos");

}

}

1. Crear el aspecto de control (Filtro)

@WebFilter(filterName = "SecurityFilter", urlPatterns = {"/my/\*"})

public class SecurityFilter implements Filter {

//...

/\*\*

\*

\* @param request The servlet request we are processing

\* @param response The servlet response we are creating

\* @param chain The filter chain we are processing

\*

\* @exception IOException if an input/output error occurs

\* @exception ServletException if a servlet error occurs

\*/

@Override

public void doFilter(ServletRequest request, ServletResponse response,

FilterChain chain)

throws IOException, ServletException {

Throwable problem = null;

try {

HttpServletRequest r = (HttpServletRequest) request;

final HttpSession session = r.getSession();

**//Al acceder a un recurso protegido validamos si hay session**

if (session.getAttribute("authenticated") == null) {

boolean loggedIn = false;

final String usuario = request.getParameter("usuario");

final String clave = request.getParameter("clave");

if (usuario != null && clave != null) {

loggedIn = this.loginService.authenticate(usuario, clave);

}

if (loggedIn) {

session.setAttribute("authenticated", Boolean.TRUE);

//otras cosas que puede hacer con el usuario autenticado

}

}

if (session.getAttribute("authenticated") != null) {

chain.doFilter(request, response);

} else {

//no estas autenticado y debo redirigirte a otro lado

HttpServletResponse rs = (HttpServletResponse) response;

rs.sendError(403);

}

} catch (Throwable t) {

// If an exception is thrown somewhere down the filter chain,

// we still want to execute our after processing, and then

// rethrow the problem after that.

problem = t;

t.printStackTrace();

}

// If there was a problem, we want to rethrow it if it is

// a known type, otherwise log it.

if (problem != null) {

if (problem instanceof ServletException) {

throw (ServletException) problem;

}

if (problem instanceof IOException) {

throw (IOException) problem;

}

sendProcessingError(problem, response);

}

}

//...

}

Aspecto de autorización

Esperamos es restringir el acceso a metodos/operaciones basados en permisos entregados al usuario.

* semántica para definir los permisos/accesos
* control de los permisos asignados a un usuario sobre un recurso
* un método para autorizar de manera local una operación

1) Definir el Pointcut

"Construir una anotación y asociarla al método a proteger"

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.METHOD, ElementType.TYPE})

@Inherited

@Documented

public @interface Requires {

String[] value() default {};

}

....

**@Requires({"operation(blog)", "operation(addEntry)"})**

public void addEntry(@PathParam("user") String user, BlogEntry content) {

....

2) Definir el interceptor//aspecto

@Inherited

@InterceptorBinding

@Retention(RUNTIME)

@Target({METHOD, TYPE})

public @interface Restrict {

}

3) Enlazar el aspecto con el código

Hay dos formas de amarrar el aspecto

* Forma 1 al método (Óptima... nos permite escoger exactamente en que métodos/operaciones corre el interceptor)

....

**@Restrict**

**@Requires({"operation(blog)", "operation(addEntry)"})**

public void addEntry(@PathParam("user") String user, BlogEntry content) {

....

* Forma 2 a la clase (ejecuta el interceptor en todos los métodos de la clase, siempre)

....

**@Restrict**

public class BlogResource implements Serializable {

....

4) Implementamos el aspecto

@Interceptor //define este bean como la implementacion de un aspecto

@Restrict //define que este interceptor se asocia al aspecto Restrict

public class RestrictInterceptor implements Serializable {

@Inject

HttpServletRequest request;

@AroundInvoke

public Object validate(InvocationContext ic) throws Exception {

Requires r = ic.getMethod().getAnnotation(Requires.class); //lectura de la anotación requires

if (r == null || r.value().length == 0 ||

|| request.getSession().getAttribute(r.value()[0]) != null) {

//la lectura y ubicación de los permisos es una convención

return ic.proceed();

}

throw new RuntimeException(

"Permiso no satisfecho " + r.value());

}

}

En el archivo beans.xml activar el interceptor

<interceptors>

<class>pe.edu.cibertec.blog.RestrictInterceptor</class>

</interceptors>