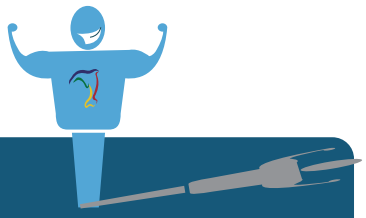
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

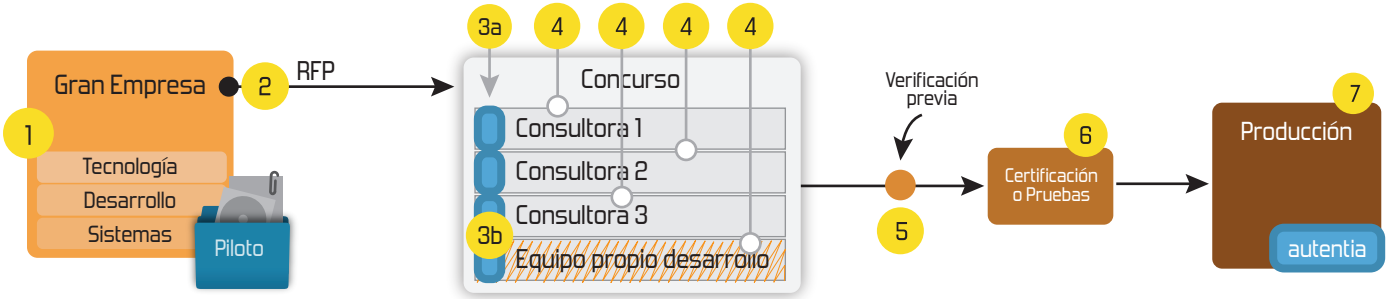
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

AdictosAlTrabajo
Terrakas 1x04
¡Estreno 6 de septiembre!
terrakas.com
autentia
Soporte a desarrollo informático
Hosting patrocinado por ENREDADOS
Entra en Adictos a través de
E-mail
Contraseña
Entrar
Deseo registrarme
Olvidé mi contraseña
Inicio Quiénes somos Formación Comparador de salarios Nuestro libro Más
» Estás en: Inicio » Tutoriales » Double Opt-In y autologin con el soporte de Spring MVC y Spring Security.

Jose Manuel Sánchez Suárez

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2012-08-24

Tutorial visitado 0 veces Descargar en PDF

Double Opt-In y autologin con el soporte de Spring MVC y Spring Security.

0. Índice de contenidos.

1. Introducción

"Double Opt-In" es un término anglosajón que hace referencia al doble consentimiento que se solicita a un usuario para ser incluido en una lista de correo electrónico o a un potencial cliente para que confirme esa dirección de email. El objetivo en ambos casos es confirmar que la dirección proporcionada se corresponde con una dirección real y, además, pertenece a la persona que se registra en nuestro site.

La secuencia temporal sería la siguiente:

- un usuario proporciona sus datos personales y dirección de correo electrónico en un formulario de nuestro site,
- nuestra lógica de negocio valida la unicidad de la dirección de correo, crea el usuario inactivo y envía un correo electrónico con un enlace de confirmación para que el usuario acceda al mismo, hasta que no acceda el usuario seguirá inactivo; el enlace incluye un token cifrado que identifica de forma unívoca el registro del usuario,
- el usuario recibe el email de confirmación, pulsa sobre el enlace y nuestra lógica de negocio valida que el token se corresponde con un usuario inactivo, lo activa y, para no volver a solicitarle usuario y contraseña, le autologa.

El ámbito de este tutorial cubre el último punto de la secuencia y vamos a analizar cómo llevarlo a cabo con el soporte de Spring MVC y Spring Security. El resto:

- el registro de usuario no tiene mayor complejidad que la recepción y validación de la información de un formulario, que podemos hacerlo también con el soporte de Spring MVC,
- al crear el usuario, se marcará como inactivo y podemos almacenar en la información del mismo el token del registro, una clave cifrada generada a partir de su email, por ejemplo, con un salt, y
- para el envío del correo electrónico podemos seguir este tutorial, para implementarlo también con el soporte de Spring.

2. Entorno.

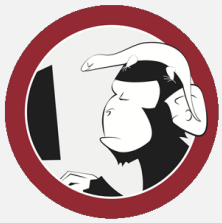
El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 15' (2.4 GHz Intel Core i7, 8GB DDR3 SDRAM).
- Sistema Operativo: Mac OS X Lion 10.7.4
- Spring MVC 3.1.2.RELEASE
- Spring Security 3.1.2.RELEASE

3. Configuración.

Para lo que nos ocupa, la configuración de nuestro proyecto no tiene relevancia; importaremos con el soporte de maven las dependencias de spring que necesitemos, a priori:

Catálogo de servicios Autentia



Síguenos a través de:

Últimas Noticias

» Estreno Terrakas 1x04: "Terraka por un día"

» Nuevos cursos de gestión de la configuración en IOS y Android

» La regla del Boy Scout y la Oxidación del Software

» Autentia conquista los Alpes

» Orientación a objetos y la importancia del "Tell, Don't Ask"

Histórico de noticias

Últimos Tutoriales

» Posicionamiento de componentes en HTML con el soporte de CSS.

» Integrando Xcode4 con GitHub

» WebSockets con Java y Tomcat 7

» Test de integración con Solr y el soporte de EmbeddedSolrServer.

```xml
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>3.1.2.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-core</artifactId>
    <version>3.1.2.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>3.1.2.RELEASE</version>
</dependency>
```

Y, a nivel de Spring Security, no tenemos más que seguir los pasos habituales para la autenticación tradicional, por ejemplo siguiendo este tutorial, puesto que no afecta al Double Opt-in, no vamos a hacer uso del authenticationManager.

4. Lógica de control.

El usuario recibiría un correo electrónico con un enlace del tipo `http://www.mi-sitio.com/registro/confirmacion-mail?t=adsg78huiomkl34` que, con el soporte de Spring MVC, podríamos mapear de la siguiente forma:

```java
@Controller
public class RegisterCtrl implements Serializable{

    @Resource
    private RegistrationCustomer registrationCustomer;

    @RequestMapping(method=RequestMethod.GET, value="registro/confirmacion-mail")
    public ModelAndView registrationCustomerConfirmation( @RequestParam(value="t") Strir
        registrationCustomer.confirm(token);
        final ModelAndView modelAndView = new ModelAndView("mailRegistrationCustomerConf
        return modelAndView;
    }

}
```

La lógica de control únicamente delega en la lógica de negocio, que veremos en el siguiente punto. La gestión de errores es genérica, si el token de registro no existe o el usuario ya está activo, devolvemos un 404, mapeando una excepción propia; veremos como más adelante.

Solo quedaría crear la vista mailRegistrationCustomerConfirmationView con el soporte de Spring MVC para informar al usuario sobre la confirmación correcta de su registro.

5. Lógica de negocio.

La lógica de negocio la implementa un servicio de Spring que podría tener un código similar al siguiente:

```java
@Service
public class RegistrationCustomer implements Serializable{

    @Resource
    private CustomerRepository customerRepository;

    public void confirm(String token) {
        final Customer customer = customerRepository.getInactiveByToken(token);
        if (customer == null){
            throw new ResourceNotFoundException("Inactive customer not found by hash coc
        }
        customer.setActive(true);
        SecurityUtils.autoLogin(customer);
    }
```

La clase Customer es un objeto de negocio propio que mapea las propiedades de la tabla de usuario.

No vamos a mostrar el contenido de la clase de repositorio (CustomerRepository) que realiza la consulta del usuario inactivo por token, asumimos que tenemos el soporte de Hibernate u otro framework de persitencia que realizaría una consulta a la tabla de usuarios en busca de uno con dicho token y marcado como inactivo.

La clase de utilidades que realmente realiza el autologin tendría un código similar al siguiente:

```java
public final class SecurityUtils {

    private static final String ROLE_CUSTOMER = "ROLE_CUSTOMER";

    private SecurityUtils(){}

    public static void autoLogin(Customer customer) {
        final Authentication authentication = getUserCredentials(customer);
        getSecurityContextHolder().setAuthentication(authentication);
    }

    private static Authentication getUserCredentials(Customer user) {
        final Set<GrantedAuthority> authorities = new HashSet<GrantedAuthority>();
        authorities.add(new SimpleGrantedAuthority(ROLE_CUSTOMER));
        return new RememberMeAuthenticationToken(user.getEmail(), user.getEmail(), authc
    }

    static SecurityContext getSecurityContextHolder() {
        return SecurityContextHolder.getContext();
    }
```

» Primeros pasos en Android.

Últimos Tutoriales del Autor

» Posicionamiento de componentes en HTML con el soporte de CSS.

» Test de integración con Solr y el soporte de EmbeddedSolrServer.

» Arrancar Solr desde un proyecto Maven con el soporte de Jetty.

» Lectura y tratamiento de ficheros Excel con Talend (II): filtros y splits

» Invocar a un servicio REST securizado, con el soporte de plantillas Spring.

Últimas ofertas de empleo

2011-09-08
Comercial - Ventas - MADRID.

2011-09-03
Comercial - Ventas - VALENCIA.

2011-08-19
Comercial - Compras - ALICANTE.

2011-07-12
Otras Sin catalogar - MADRID.

2011-07-06
Otras Sin catalogar - LUGO.

Jose Manuel Sánchez
sanchezsuarezj

sanchezsuarezj #mallorca TI, un paraíso acuático para los niños. Solo por eso el hotel merece la pena, el resto genial. #sol #meliá pic.twitter.com/q65QRsH6
5 days ago · reply · retweet · favorite

sanchezsuarezj @adictosaltrabaj tuto sencillo sobre posicionamiento de componentes en HTML con el soporte de CSS. kcy.me/afw8
8 days ago · reply · retweet · favorite

sanchezsuarezj @ruedalenticular invitado quedas!!! Eso si, no te bajes en bici que se te va a hacer un poco largo ;)
12 days ago · reply · retweet · favorite

sanchezsuarezj arrancó la feria

Join the conversation

```
}
```

Como en este punto, no se ha realizado un login al uso, esto es, no ha habido un formulario de petición de usuario y contraseña, no podemos utilizar un UsernamePasswordAuthenticationToken, es por ello, que usamos un RememberMeAuthenticationToken.

Y, como nuestros clientes tienen un único rol de usuario, asignamos a todos una única credencial "ROLE_CUSTOMER", si tuvieramos más esa informacion estaría mapeada en la propia clase Customer y la obtendríamos de ella.

6. Gestión de excepciones.

Nuestra excepción tendría un código similar al siguiente:

```
@ResponseStatus(value=HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends RuntimeException {

    private final String resource;
    private Throwable cause;

    public ResourceNotFoundException(String resource){
        this.resource = resource;
    }

    public ResourceNotFoundException(String resource,
             Throwable cause) {
        this.resource = resource;
        this.cause = cause;
    }
}
```

Con la anotación @ResponseStatus se estamos diciendo a Spring MVC que cuando se lance esta excepción se envíe al usuario un 404.

7. Conclusiones.

También hay que decir, que la técnica del "Double Opt-in" se convierte en "Single Opt-in" cuando estamos en un proceso de compra o, lo que es lo mismo, en la confirmación del contenido de un carrito. En ese caso, si el usuario no está registrado, al realizar el registro no se solicita confirmación del correo para no perder una compra ;)

Un saludo.

Jose

jmsanchez@autentia.com

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

Share | 0

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

» **Registrate** y accede a esta y otras ventajas «

SOME RIGHTS RESERVED Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

Copyright 2003-2012 © All Rights Reserved | Texto legal y condiciones de uso | Banners | Powered by Autentia | Contacto

XHTML 1.0 CSS RSS ATOM