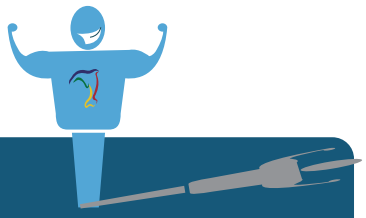
autentia

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...

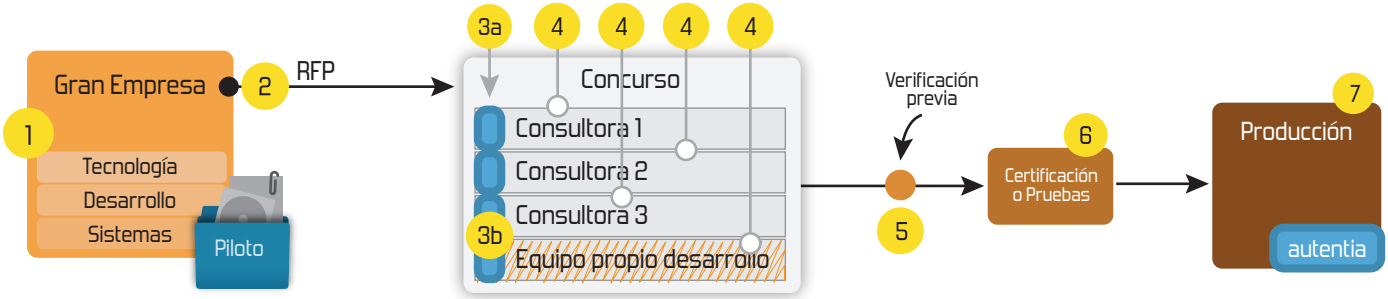
1. Desarrollo de componentes y proyectos a medida



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 conociemiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

adictos altrabajo
¡Extra, extra! Sale la SEGUNDA EDICIÓN del libro en menos de un año que lleva a la venta
autentia business solutions
patrocinado por REDados

DESARROLLADO POR:
Rubén Aguilera Díaz-Heredero

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero en Informática, especialidad en Ingeniería del Software

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

Somos expertos en Java/J2EE

Anuncios Google Java Video Tutorial Java Java Code Examples

Fecha de publicación del tutorial: 2011-03-05

1

Share | Regístrate para votar

CRUD con Spring MVC Portlet (IV): Realizando pruebas unitarias

0. Índice de contenidos.

- 1. Entorno
- 2. Introducción
- 3. Manos a la obra
- 4. Conclusiones

1. Entorno

Este tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil Mac Book Pro 17" (2,6 Ghz Intel Core i7, 8 GB DDR3)
- Sistema Operativo: Mac OS X Snow Leopard 10.6.4
- Spring MVC Portlet 3.0.4
- Maven 2.2.1
- Eclipse 3.6 (Helios) con M2Eclipse
- Spring Test 3.0.4

2. Introducción

Este tutorial es la cuarta parte de una serie, por lo que te recomiendo que antes leas:

- CRUD con Spring MVC Portlet
- CRUD con Spring MVC Portlet (II)
- CRUD con Spring MVC Portlet (III)

En esta cuarta parte vamos a ver cómo podemos probar nuestro desarrollo sin necesidad de arrancar el gestor de portales, gracias a los mocks que nos proporciona Spring en su librería de test.

3. Manos a la obra

Catálogo de servicios Autentia

Últimas Noticias
Java Database Class
XV Charla Autentia - web2py (y Google App Engine)
XIV Charla Autentia - ZK - Vídeos y Material
Hablando de coaching ágil, milagro nocturo y pruebas de vida
XIII Charla Autentia - AOS y TDD - Vídeos y Material
Las metodologías ágiles como el catalizador del cambio

Histórico de NOTICIAS

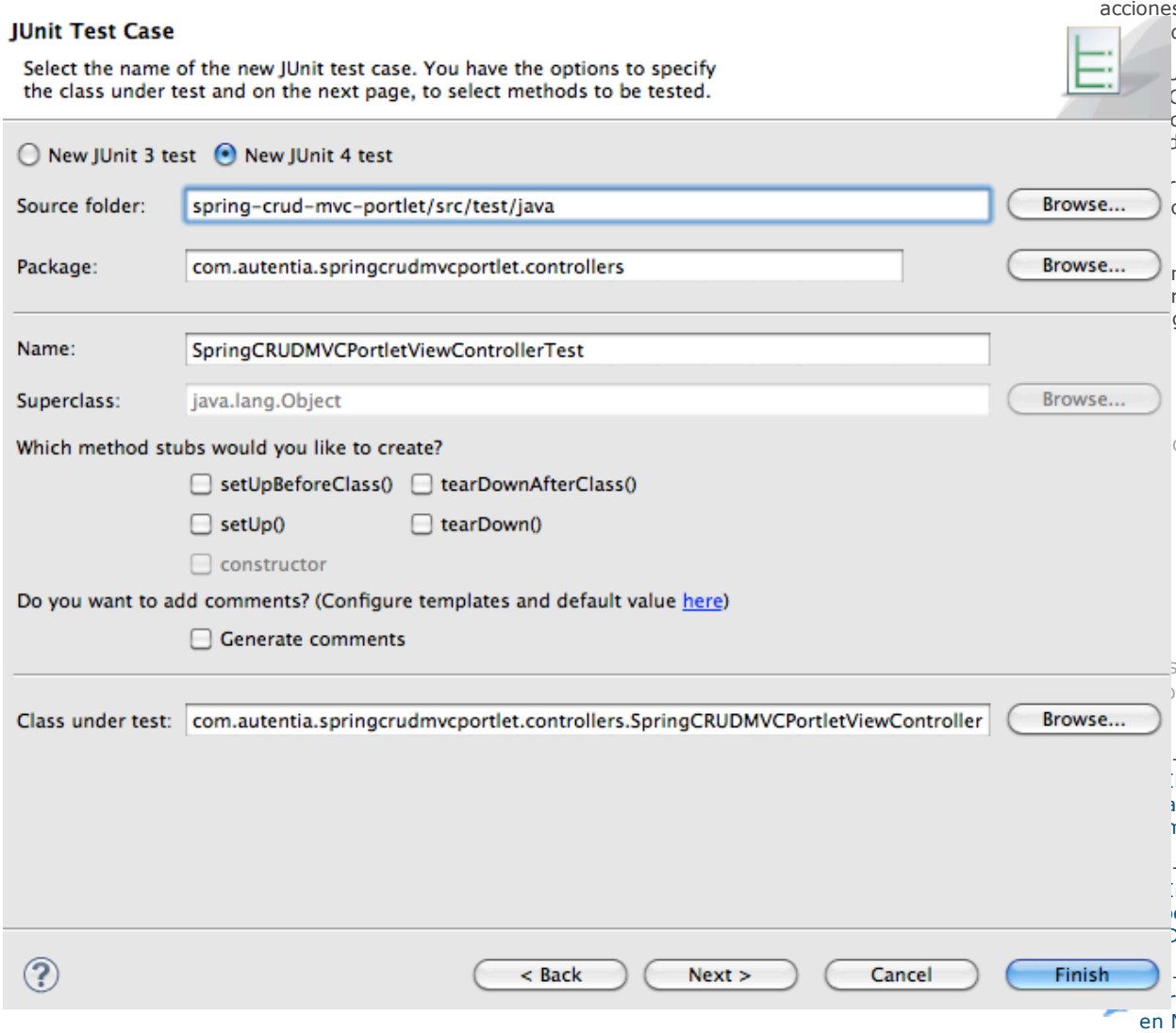
Últimos Tutoriales
Introducción a Selenium 2 y WebDriver
Dividir tu pantalla gigante en Mac con Divvy
Mi experiencia en Autentia

Lo primero que necesitamos es añadir la dependencia de spring test al pom de nuestro proyecto de esta forma:

view plain print ?

```
01. <dependency>
02.     <groupid>org.springframework</groupid>
03.     <artifactid>spring-test</artifactid>
04.     <version>3.0.4.RELEASE</version>
05.     <scope>test</scope>
06. </dependency>
07. <dependency>
08.     <groupid>junit</groupid>
09.     <artifactid>junit</artifactid>
10.     <version>4.8.2</version>
11.     <scope>test</scope>
12. </dependency>
```

Ahora creamos la clase del test con el asistente de Eclipse para ello con botón derecho sobre la clase del controlador seleccionamos New --> Other --> JUnit --> JUnit Test Case asegurándonos de crear la clase de test dentro de la carpeta src/test/java (si no existe la creamos) para que Maven tenga el cuenta este test a lo ahora de ejecutar sus goals.



Una vez creada la clase de test, lo siguiente es configurarla para que arranque el contexto de Spring de nuestra aplicación.

Patrón Intérprete

NIC Bonding, NIC Teaming, Port Trunking, Etherchannel o Ether bonding, con ifenslave en Ubuntu

Últimos Tutoriales del Autor

CRUD con Spring MVC Portlet (III): Añadiendo validación al formulario

CRUD con Spring MVC Portlet (II): Creando el listado y las acciones de modificación
do

UD con Spring
C Portlet:
o el proyecto y la
de alta

ería de acceso a
os con Spring y

mplo de Swing
rker: ¿Por qué se
gela la interfaz?

os a través de:

s ofertas de

-02
información -
alista /
nador - MALAGA.

-24
información -
ecialista CRM -
D.

-16
rketing - Experto
en Marketing -
CADIZ.

2011-02-08
Comercial - Ventas - CADIZ.

2011-01-28
Comercial - Ventas - SEVILLA.

view plain print ?

```
01. package com.autentia.springcrudmvcportlet.controllers;
02.
03. import org.junit.runner.RunWith;
04. import org.springframework.test.context.ContextConfiguration;
05. import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
06.
07. @RunWith(SpringJUnit4ClassRunner.class)
08. @ContextConfiguration(locations = { "file:src/main/webapp/WEB-
    INF/context/applicationContext.xml" })
09. public class SpringCRUDMVCPortletViewControllerTest {
10.
11. }
```

Con estas anotaciones levantamos el contexto de Spring especificado en el atributo locations, que en nuestro caso, apunta al fichero applicationContext.xml de nuestro proyecto.

Ya estamos en disposición de realizar nuestro primer test unitario con Spring MVC Portlet. Como es el primero va a ser sencillo, simplemente vamos a chequear que el método defaultView mete en el model el atributo "listaPersonas".

view plain print ?

```
01. @Resource
02. private SpringCRUDMVCPortletViewController springCRUDMVCPortletViewController;
03.
04. @Test
05. public void defaultView(){
06.     ModelMap model = new ModelMap();
07.     springCRUDMVCPortletViewController.defaultView(model);
08.     Assert.assertEquals(true, model.containsAttribute("listaPersonas"));
09. }
```

Ahora sólo basta que ejecutemos el test para ver si la condición se cumple o se produce algún tipo de error.

Lo siguiente que vamos a hacer es probar el formulario y la validación del mismo. Para ello vamos a crear al menos dos casos de prueba, uno que sea correcto y otro que contenga un error, por ejemplo, que la persona introduzca como nombre Pepe (vuelvo a repetir que no tengo nada en contra de los que se llaman Pepe)

view plain print ?

```
01. @Resource
02. private PersonaFormValidator personaFormValidator;
03.
04. @Test
05. public void validatorFormOK(){
06.     PersonaForm personaForm = new PersonaForm();
07.     personaForm.setNombre("Verónica");
08.     personaForm.setApellidos("Gómez Gil");
09.     personaForm.setDireccion("Calle de la Farmacia, 3");
10.     BindingResult result = new BeanPropertyBindingResult(personaForm, "personaForm")
11.     personaFormValidator.validate(personaForm, result);
12.     Assert.assertEquals(false, result.hasErrors());
13. }
14.
15. @Test
16. public void validatorFormNoPepes(){
17.     PersonaForm personaForm = new PersonaForm();
18.     personaForm.setNombre("Pepe");
19.     personaForm.setApellidos("Pérez");
20.     personaForm.setDireccion("Calle de los Pepes, 3");
21.     BindingResult result = new BeanPropertyBindingResult(personaForm, "personaForm")
22.     personaFormValidator.validate(personaForm, result);
23.     Assert.assertEquals(true, result.hasErrors());
24. }
```

Ahora vamos a probar la inserción, la modificación y el borrado de una persona. Fijaos como Spring nos proporciona las clases MockActionResponse y MockActionRequest para cuando necesitamos simular el contexto de la request en los métodos de acción del portlet y MockRenderResponse y MockRenderRequest para los métodos de vista. Este podría ser un ejemplo:

view plain print ?

```
01. @Test
02. public void insertarPersonaOK(){
03.     PersonaForm personaForm = new PersonaForm();
04.     personaForm.setNombre("Verónica");
05.     personaForm.setApellidos("Gómez Gil");
06.     personaForm.setDireccion("Calle de la Farmacia, 3");
07.     BindingResult result =
08.         new BeanPropertyBindingResult(personaForm, "personaForm");
09.     String cancelar = null;
10.     ModelMap model = new ModelMap();
11.     MockActionResponse response = new MockActionResponse();
12.     String accion = "crud.alta";
13.     springCRUDMVCPortletViewController.insertarPersona(personaForm,
14.         result, cancelar, model, response, accion);
15.     Assert.assertEquals(false, model.containsAttribute("futuraAccion"));
16.
17. }
18.
19. @Test
20. public void modificarPersonaOK(){
21.     PersonaForm personaForm = new PersonaForm();
22.     personaForm.setNombre("Verónica");
23.     personaForm.setApellidos("Gómez Gil");
24.     personaForm.setDireccion("Calle de la Medicina, 3");
25.     personaForm.setIdPersona(new Long(26));
26.     BindingResult result =
27.         new BeanPropertyBindingResult(personaForm, "personaForm");
28.     String cancelar = null;
29.     ModelMap model = new ModelMap();
30.     MockActionResponse response = new MockActionResponse();
31.     String accion = "crud.modificacion";
32.     springCRUDMVCPortletViewController.modificarPersona(personaForm,
33.         result, cancelar, model, response, accion);
34.     Assert.assertEquals(false, model.containsAttribute("futuraAccion"));
35. }
36.
37. @Test
38. public void borrarPersonaOK(){
39.     PersonaForm personaForm = new PersonaForm();
40.     personaForm.setIdPersona(new Long(26));
41.     String cancelar = null;
42.     ModelMap model = new ModelMap();
43.     springCRUDMVCPortletViewController.bajaPersona(personaForm,
44.         cancelar, model);
45.     Assert.assertEquals(false, model.containsAttribute("futuraAccion"));
46. }
```

4. Conclusiones

Como se demuestra es sencillo probar la lógica de nuestros portlets cuando están hechos con Spring MVC. La ventaja fundamental es que podemos realizar todas las pruebas que queramos sin necesidad de arrancar el gestro de portales, lo que provoca una reducción en el tiempo de desarrollo de los portlets y además se proporciona una forma de poder aplicar TDD para el desarrollo de portlets.

Con este tutorial doy por finalizada esta serie, y espero que os sea de utilidad.

Cualquier duda o sugerencia en la zona de comentarios.

Saludos.

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

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

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

Comentarios

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

Copyright 2003-2011 © All Rights Reserved | Texto legal | Contacto | Aviso legal | Condiciones de uso | Banners | Powered by Autentia |