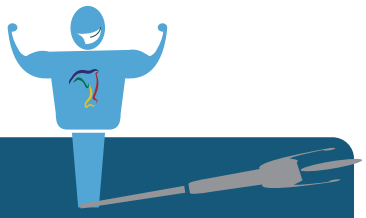
autentia

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...

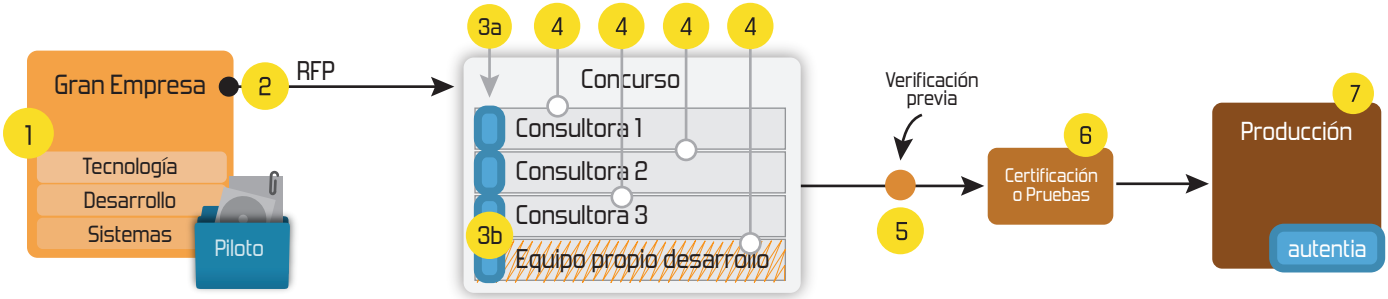
1. Desarrollo de componentes y proyectos a medida



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

Lanzado TNTConcept versión 0.8 (10/12/2007)

¿Gestionas tu empresa con hojas de cálculo? ¿No crees que puede haber un modo mejor?

Desde Autentia ponemos a vuestra disposición el software que hemos construido (100% gratuito, con código fuente disponible y sin restricciones funcionales) para nuestra gestión interna, llamado TNTConcept (au**T**e**NT**ia). Construida con las últimas tecnologías de desarrollo Java/J2EE (Spring, JSF, Acegi, Hibernate, Maven, Subversion, etc.) y disponible en licencia GPL, seguro que a muchos profesionales independientes y PYMES os ayudará a organizar mejor vuestra operativa.

Las cosas grandes empiezan siendo algo pequeño ..... Saber más en: http://tntconcept.sourceforge.net/

Tutorial desarrollado por: Javier Antoniucci

Puedes encontrarme en Autentia
Somos expertos en Java/J2EE
Contacta en:
javier.antoniucci@autentia.com

NUEVO CATÁLOGO DE SERVICIOS DE AUTENTIA (PDF 6,2MB)

www.adictosaltrabajo.com es el Web de difusión de conocimiento de www.autentia.com

autentia real business solutions

Catálogo de cursos

Descargar este documento en formato PDF springwebvalidator.pdf

Firma en nuestro libro de Visitas **<-----> Asociarme al grupo AdictosAlTrabajo en eConozco**

Fecha de creación del tutorial: 2007-12-11

Spring WebFlow con Validator

En el tutorial "Manual Básico de Spring WebFlow" hemos visto cómo poner a funcionar este maravilloso controlador. Cuando comenzamos a desarrollar una aplicación, descubrimos inmediatamente que la interfaz de usuario involucra validaciones básicas de los datos que introduce el usuario. Habitualmente utilizamos validaciones específicas en las acciones del controlador o en componentes especializados pero a medida que crece la complejidad de la aplicación esta solución no es escalable ni mantenible. Surgen entonces propuestas como Apache Validator donde mediante uno o más ficheros de configuración se definen las reglas de validación y la aplicación de dichas reglas a los campos de los formularios.

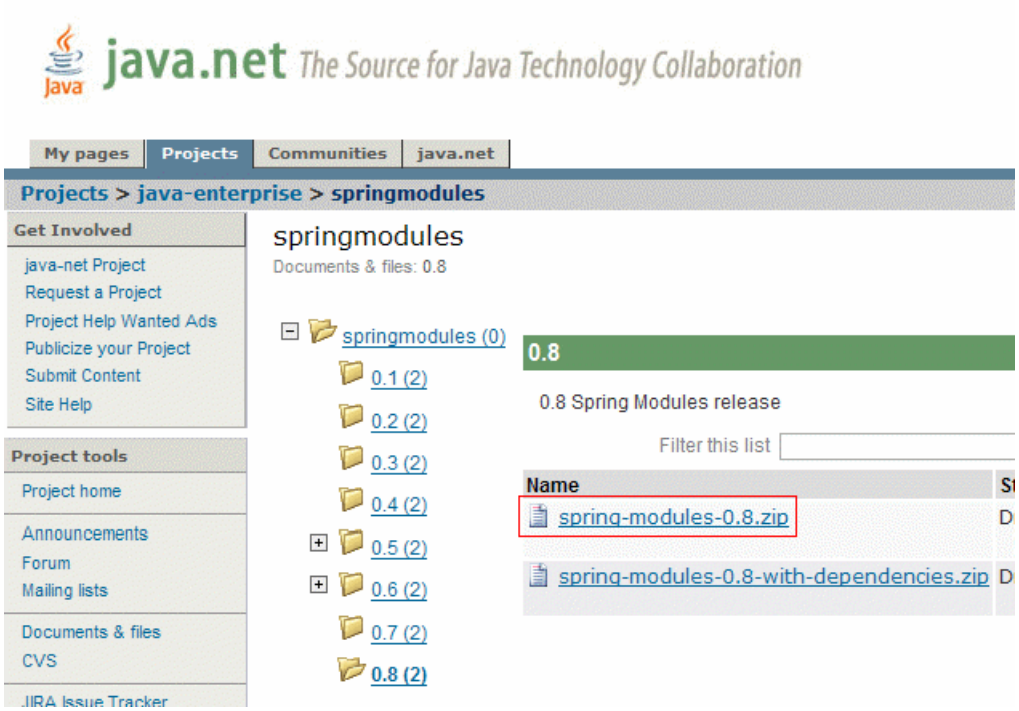
El objetivo de este tutorial es facilitar el primer contacto con esta tecnología partiendo del ejemplo realizado en el tutorial "Manual Básico de Spring WebFlow" analizar su aplicabilidad, recomendar algunos links y comentaros nuestras conclusiones.

Preparación del proyecto

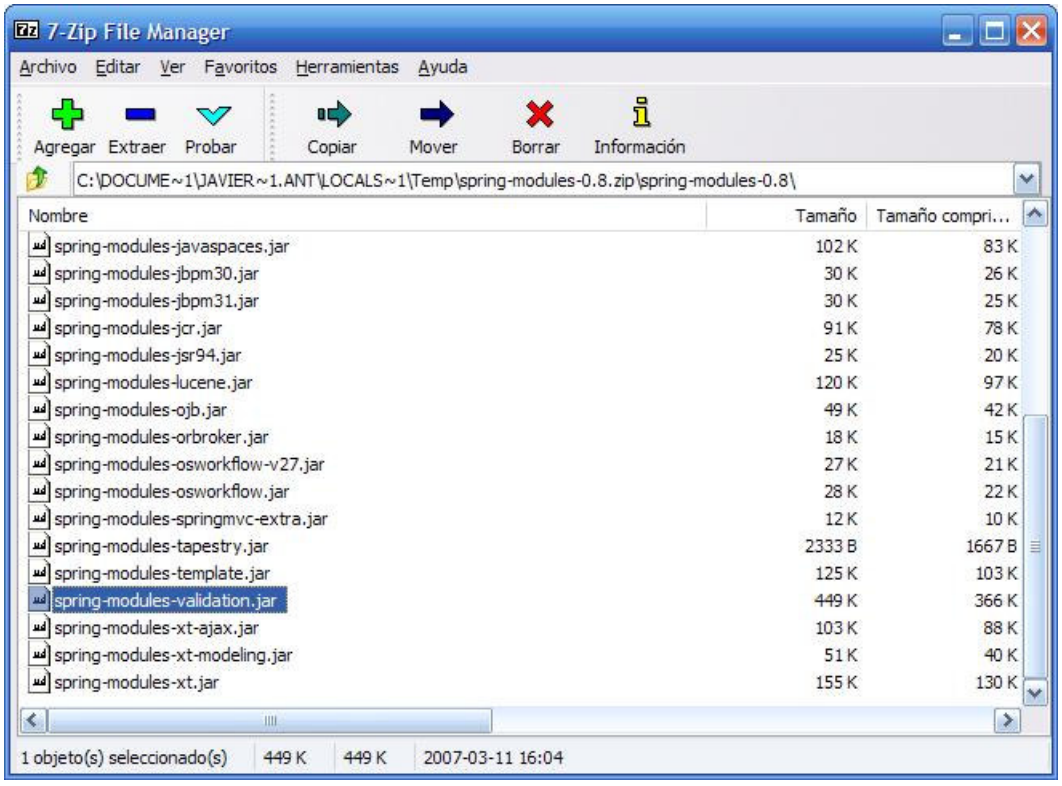
Comenzaremos a partir del proyecto que creamos en el ejemplo realizado en el tutorial "Manual Básico de Spring WebFlow".

Lo primero que haremos será descargarnos el spring modules https://springmodules.dev.java.net/#downloading seleccionando Releases y la última disponible:

Y la versión del fichero sin dependencias:



Entonces, abrimos el zip



y copiamos el spring-modules-validation.jar en nuestra carpeta /WEB-INF/lib del proyecto.

Configurando la aplicación

Abrimos el fichero pruebaWebFlowSimple-servlet-config.xml y añadimos las siguientes definiciones:

```
<bean id="messageSource"
```

```
class="org.springframework.context.support.ResourceBundleMessageSource">
<property name="alwaysUseMessageFormat" value="true" />
<property name="basenames">
<list>
<value>messages</value>
</list>
</property>
</bean>

<bean id="validatorFactory"
class="org.springmodules.validation.commons.DefaultValidatorFactory">
<property name="validationConfigLocations">
<list>
<value>/WEB-INF/validations/validator-rules.xml</value>
<value>/WEB-INF/validations/validation.xml</value>
</list>
</property>
</bean>
```

Primero definimos un messageSource con el que se resolverá la internacionalización de la aplicación. Luego se declara el factory del validador y configurándolo con un fichero de definición de reglas y otro de definición de la aplicación de dichas reglas a un formulario. También vamos a modificar la definición del formAction en /web/WEB-INF/flows/asistente-beans.xml para incluirle el validador (resaltado en negrita):

```
<beans xmlns="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-2.0.xsd">

<bean id="formAction"
class="es.autentia.tutoriales.manualBasicoSpringWebFlow.controller.action.AsistenteAction">
<property name="formObjectScope" value="FLOW" />
<property name="formObjectName" value="form" />
<property name="formObjectClass"
value="es.autentia.tutoriales.manualBasicoSpringWebFlow.controller.form.AsistenteForm" />
<property name="validator" ref="formActionValidator"/>
</bean>

<bean id="formActionValidator" class="org.springmodules.validation.commons.ConfigurablePageBeanValidator">
<property name="formName" value="formAction" />
<property name="validatorFactory" ref="validatorFactory" />
</bean>

</beans>
```

Vamos a crear los ficheros /WEB-INF/validator-rules.xml y /WEB-INF/validation.xml.Para hacerlo sencillo copiaremos los que vienen en el ejemplo spring-modules-validation-commons-samples-src.zip que viene en spring-modules-0.8.zip y que encontraremos en \spring-modules-0.8\samples\sources\spring-modules-validation-bean-samples-src.zip. Los encontraremos en la carpeta \webapp\WEB-INF\.

Analicemos un fragmento de validator-rules.xml:

```
<form-validation>

<global>

<validator name="required"

classname="org.springmodules.validation.commons.FieldChecks"

method="validateRequired"

methodParams="java.lang.Object,

org.apache.commons.validator.ValidatorAction,

org.apache.commons.validator.Field,

org.springframework.validation.Errors"

msg="errors.required">

<javascript><![CDATA[

...
```

En dicho fichero se define bastante intuitivamente cada regla de validación. Commons validation incluye una serie de reglas predefinidas pero podemos desarrollar fácilmente las nuestras (será parte de un futuro tutorial). Básicamente indicamos su nombre, clase, nombre y parámetro del método que implementa dicha validación y finalmente se puede incluir un javascript que realizará dicha valdación en cliente. Esto último requiere que incluyamos unas taglibs en las jsps del formulario.

Aprovecharemos el validation.xml del ejemplo para definir el que utilizaremos en el proyecto:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE form-validation PUBLIC

"-//Apache Software Foundation//DTD Commons Validator Rules Configuration 1.1//EN"

"http://jakarta.apache.org/commons/dtds/validator_1_1.dtd">

<form-validation>

<formset>

<form name="asistenteForm">

<field property="numero1" depends="required,integer,intRange">

<arg position="0" key="asistente.numero1.etiqueta" />

<arg position="1" name="intRange" key="${var:min}"

resource="false" />

<arg position="2" name="intRange" key="${var:max}"

resource="false" />

<var>

<var-name>min</var-name>

<var-value>1</var-value>
```

```
</var>

<var>

<var-name>max</var-name>

<var-value>20</var-value>

</var>

</field>

<field property="numero2" depends="required,integer,intRange">

<arg position="0" key="asistente.numero2.etiqueta" />

<arg position="1" name="intRange" key="${var:min}"

resource="false" />

<arg position="2" name="intRange" key="${var:max}"

resource="false" />

<var>

<var-name>min</var-name>

<var-value>1</var-value>

</var>

<var>

<var-name>max</var-name>

<var-value>20</var-value>

</var>

</field>

</form>

</formset>

</form-validation>
```

Ahora creamos el messages.properties en la carpeta src y copiamos las claves necesarias:

```
asistente.numero1.etiqueta=Primer número

asistente.numero2.etiqueta=Segundo número

# Struts Validator Error Messages

errors.required={0} es obligatorio.

errors.minlength={0} no debe ser menor a {1} caracteres.

errors.maxlength={0} no debe ser mayor a {1} caracteres.

errors.invalid={0} es invalido.

errors.byte={0} debe ser un byte.

errors.short={0} debe ser un short.

errors.integer={0} debe ser un integer.

errors.long={0} debe ser un long.
```

errors.float={0} debe ser un float.

errors.double={0} debe ser un double.

errors.date={0} no es una fecha válida.

errors.range={0} no está entre {1} y {2}.

errors.creditcard={0} no es un número de tarjeta de crédito válido.

errors.email={0} no es una dirección de e-mail válida.

Para poder apreciar mejor la validación, vamos a comentar las validaciones que se realizan en el AsistenteAction.java, dejando como resultado siempre success:

```
public Event validarPaso1YPrepararPaso2(RequestContext context)

throws Exception {

AsistenteForm form = (AsistenteForm) getFormObject(context);

Event resultado = success();

// if (form.getNumero1() < 0 || form.getNumero2() < 0) {

// form.setMensaje("Los números a sumar deben ser positivos");

// resultado = error();

// } else {

// form.setMensaje("El resultado de la suma es "

// + (form.getNumero1() + form.getNumero2()));

// resultado = success();

// }

return resultado;

}
```

Para que podamos visualizar los mensajes de error, vamos a incluir en /web/WEB-INF/jsp/paso1.jsp el siguiente código (resaltado en negrita):

```
<%@ taglib uri="http://www.springframework.org/tags" prefix="spring"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>

<title>Asistente sumador</title>

</head>

<body>

<h1>Asistente sumador: Paso 1</h1>

<spring:hasBindErrors name="asistenteForm">

<h2>Solucione los siguientes errores</h2>

<ul>

<c:forEach var="error" items="${errors.allErrors}">
```

```
<li>
<spring:message code="${error.code}" arguments="${error.arguments}"/>
</li>
</c:forEach>
</ul>
</spring:hasBindErrors>

<form>
<input type="hidden" name="_flowId" value="asistente-flow" />
<input type="hidden" name="_eventId" value="siguiente" />
<input type="hidden" name="_flowExecutionKey" value="${flowExecutionKey}" />
<p>${asistenteForm.mensaje}</p>
<p>Primer número: <input type="text" name="numero1" value="${asistenteForm.numero1}"/></p>
<p>Segundo número: <input type="text" name="numero2" value="${asistenteForm.numero2}"/></p>
<input type="submit" value="siguiente" />
</form>
</body>
</html>
```

También deberemos cambiar las páginas del paso 2 y 3 para que utilicen el formulario asistenteForm en lugar de form como se llamaba antes.

En el /web/WEB-INF/flows/asistente-beans.xml vamos a definir el validador a utilizar y vamos a cambiar el nombre del formulario:

```
<beans xmlns="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-2.0.xsd">

<bean id="formAction"
class="es.autentia.tutoriales.manualBasicoSpringWebFlow.controller.action.AsistenteAction">
<property name="formObjectScope" value="FLOW" />
<property name="formObjectName" value="asistenteForm" />
<property name="formObjectClass"
value="es.autentia.tutoriales.manualBasicoSpringWebFlow.controller.form.AsistenteForm" />
<property name="validator" ref="formActionValidator"/>
</bean>

<bean id="formActionValidator" class="org.springmodules.validation.commons.DefaultBeanValidator">
<property name="validatorFactory" ref="validatorFactory" />
</bean>

</beans>
```

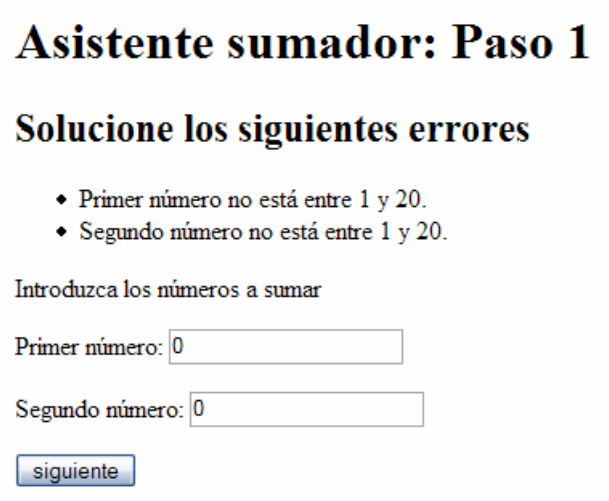
Finalmente, en el /web/WEB-INF/flows/asistente-flow.xml vamos a invocar al método bindAndValidate en la transición del paso 1 al 2 como se indica en negrita:

```
<?xml version="1.0" encoding="UTF-8"?>

<flow xmlns="http://www.springframework.org/schema/webflow"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/webflow

http://www.springframework.org/schema/webflow/spring-webflow-1.0.xsd">


<start-actions>

<action bean="formAction" method="setupForm" />

</start-actions>


<start-state idref="mostrarPaso1" />


<view-state id="mostrarPaso1" view="paso1">

<transition on="siguiente" to="avanzaAPaso2">

<action bean="formAction" method="bindAndValidate" />

</transition>

</view-state>


<action-state id="avanzaAPaso2">

<action bean="formAction" method="validarPaso1YPrepararPaso2" />

<transition on="success" to="mostrarPaso2" />

<transition on="error" to="mostrarPaso1" />

</action-state>


<view-state id="mostrarPaso2" view="paso2">

<transition on="siguiente" to="avanzaAPaso3">

<action bean="formAction" method="bind" />

</transition>

<transition on="anterior" to="mostrarPaso1">

<action bean="formAction" method="bind" />

</transition>

</view-state>


<action-state id="avanzaAPaso3">

<action bean="formAction" method="validarPaso2YPrepararPaso3" />

<transition on="success" to="mostrarPaso3" />

<transition on="error" to="mostrarPaso2" />

</action-state>

<end-state id="mostrarPaso3" view="paso3" />
```

```
<import resource="asistente-beans.xml" />

</flow>
```

Ejecutando la aplicación

Lanzamos el servidor con la URL http://localhost:8080/pruebaWebFlowSimple/asistente.flow y veremos en funcionamiento el ejemplo si ponemos 0 en los valores para el número 1 y 2:



Algunos links interesantes

- Sitio oficial de Spring Modules, https://springmodules.dev.java.net
- Documentación oficial de Referencia , https://springmodules.dev.java.net/docs/reference/0.8/html/validation.html
- Documentación oficial de Commons Validator, http://struts.apache.org/1.x/faqs/validator.html

Conclusiones

Tras utilizarlo en proyectos de envergadura concluimos que:

- El framework nos simplifica asombrosamente la implementación de las validaciones de interfaz de usuario y nos aporta ya resueltas las validaciones más frecuentes.
- Nos aporta la capacidad de poder centralizar y generar reportes actualizados de todas las validaciones de interfaz de usuario que se realizan en la aplicación. Esto es muy interesante de cara a presentar informes de revisión y auditoría a equipos de negocio.
- La configuración de las validaciones (validation.xml) se podría generar desde una herramienta o una excel operada por integrantes del equipo de negocio.
- El Apache Commons Validator se integra perfectamente con Spring WebFlow.
- Su internacionalización es muy potente y completa.
- No debemos confudir validaciones de interfaz de usuario con las validaciones de negocio. Las primeras son relativas a tipos de datos, rangos, etc. y las segundas tienen que ver con restricciones de negocio, cálculos, etc.
- Incluye características avanzadas como soporte a validaciones en interfaces tipo asistente (varios pasos), validación de múltiples campos, validaciones condicionales y hasta soporte a un lenguaje de validaciones en xml llamado valang.

Desde Autentia contamos con los conocimientos y experiencia para ayudarle a sacar la máxima ventaja de las tecnologías más innovadoras y mejorar la calidad de sus desarrollos software.

No dude en contactarse con nosotros mediante www.autentia.com .

Puedes opinar sobre este tutorial aquí

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e-mail	
	Enviar

Otros Tutoriales Recomendados (También ver todos)

Nombre Corto	Descripción
Comparativa entre EJB3 y Spring	En este tutorial os mostramos una comparativa entre EJB3 y Spring esperando que os ayude a decidir qué tecnología utilizar.
Spring WebFlow Tiles	En este tutorial aprenderemos el uso de tiles para usarlo en conjunción con Spring WebFlow.
URLs amigables con Spring MVC	En este tutorial se va a hacer un ejemplo práctico utilizando Spring MVC para la configuración de URLs amigables de nuestra aplicación
Manual básico de Spring WebFlow	En este tutorial Javier Antoniucci nos enseña cómo empezar a trabajar cpn el framework de desarrollo web Spring webflow.
Spring: definición dinámica de Beans	Este tutorial habla sobre la modificación dinámica de los beans del contexto para simplificar la configuración de Spring
Introducción a Spring Web Flow	Spring Web Flow es un módulo de extensión del framework Spring, que facilita la implementación del flujo de páginas de una aplicación web
Integración de JSF 1.2, Facelets e ICEFaces en Tomcat 6	Integración de JSF 1.2, Facelets e ICEFaces en Tomcat 6