

¿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

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

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

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

AdictosAlTrabajo

Temporada Completa
de Terrakas
terrakas.com



Entra en Adictos a través de



E-mail

Contraseña

Entrar

[Deseo registrarme](#)
[Olvidé mi contraseña](#)



[Inicio](#) [Quiénes somos](#) [Formación](#) [Comparador de salarios](#) [Nuestros libros](#) [Más](#)

» Estás en: [Inicio](#) [Tutoriales](#) [Mi primera vista en ZK como desarrollador JSF \(I\)](#).



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: 2014-01-09

Tutorial visitado 1 veces [Descargar en PDF](#)

Mi primera vista en ZK como desarrollador JSF (I).

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno.
- 3. Configuración.
- 4. Internacionalización.
- 5. Sistema de plantillas.
- 6. Generando la capa de control para mantener el estado y comportamiento de la vista.
- 7. Inyección de dependencias de beans de Spring en la capa de control -ViewModel- de ZK.
- 8. Referencias.
- 9. Conclusiones.
- 10. ¿Necesitas formación?.

1. Introducción

ZK es un framework orientado a la capa de presentación que permite generar aplicaciones RIA, aplicaciones ricas de internet, haciendo uso de una colección de componentes visuales que facilita enormemente la tarea de desarrollar una interfaz de usuario interactiva. La comunicación de eventos entre cliente y servidor se realiza por Ajax de una manera transparente sin necesidad de hacer uso de lenguajes de programación en cliente: HTML y/o Javascript. Son esos componentes visuales los que encapsulan la renderización en el navegador de la sintaxis adecuada de HTML y el comportamiento en Javascript para añadir la captura de eventos en el cliente y la invocación a métodos en la capa de control en el servidor.

El objetivo de este tutorial es mostrar los aspectos básicos del framework desde el punto de vista de un desarrollador con experiencia en otros frameworks de presentación y, en concreto, en JSF. Por definición, un framework de presentación debe proporcionar el soporte necesario para llevar a cabo la internacionalización de la interfaz, hacer uso de plantillas, validar la información de entrada, convertir tipos de datos complejos, mantener el estado de los objetos entre vistas, configurar la capa de control y reglas de navegación,... en este tutorial comenzaremos a trabajar en estos conceptos, dando por hecho que disponemos de conocimientos previos de JSF.

No se trata de realizar una comparativa entre ambos frameworks (ZK VS JSF), básicamente, porque no se pueden comparar, no ofrecen lo mismo. Para realizar una comparativa con JSF tendríamos que elegir una librería de componentes visuales para JSF, como Primefaces; todo se andará... para los que tengáis experiencia, sabréis que el primer párrafo de esta introducción podría describir, de la misma forma, a ambas librerías de componentes visuales, tanto ZK como Primefaces.

Sin llegar al nivel de componentes visuales RIA y basándonos en los conceptos que nos proporcionaría la especificación de JSF, vamos a ver cómo comenzar a trabajar con ZK; y lo haremos haciendo una clara separación de capas de modo tal que la lógica de negocio residirá en beans de Spring que inyectaremos en la capa de control con el soporte nativo que nos proporciona ZK.

Para más información en castellano sobre el framework ZK, podéis acudir a la [introducción](#) y [tutoriales anteriores de ZK](#) escritos por mi compi [Francisco Ferri](#), la persona que más sabe de ZK que conozco.

2. Entorno.

El tutorial, y el código que contiene, han sido escritos 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
- Eclipse Kepler + m2e plugin
- ZK 7.0.0 CE

3. Configuración.

Catálogo de servicios Autentia



Síguenos a través de:



Últimas Noticias

» [IX Autentia Cycling Day \(ACTUALIZADO\)](#)

» [Autentia en la carrera de las empresas](#)

» [Spring 4.0 ¿qué hay de nuevo amigo?](#)

» [Torneo de pádel solidario AMEB](#)

» [Próxima charla: Gradle como alternativa a Maven para la construcción de proyectos en Java](#)

[Histórico de noticias](#)

Últimos Tutoriales

» [Transiciones personalizadas en iOS7](#)

» [Spring BeanPostProcessor](#)

» [Cómo montar un raid1 en una máquina corriendo debian.](#)

» [Notificaciones locales en iOS.](#)

» [Configuración básica de seguridad en servidor linux con iptables y fail2ban](#)

En este punto vamos a ver la configuración básica en nuestro módulo web para soportar ZK y como de costumbre, haciendo uso de maven, lo primero es declarar nuestras dependencias en el fichero pom.xml:

```

1  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
2  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
3  <modelVersion>4.0.0</modelVersion>
4  <groupId>com.autentia.training</groupId>
5  <artifactId>management-training-zk</artifactId>
6  <packaging>war</packaging>
7  <version>0.0.1-SNAPSHOT</version>
8  <name>management-training-zk</name>
9
10 <organization>
11 <name>Autentia Real Business Solutions S.L.</name>
12 <url>http://www.autentia.com</url>
13 </organization>
14
15 <developers>
16 <developer>
17 <name>Jose Manuel Sánchez Suárez</name>
18 <email>jmsanchez@autentia.com</email>
19 </developer>
20 </developers>
21
22 <properties>
23 <java.version>1.7</java.version>
24 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
25 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
26 <spring.version>4.0.0.RELEASE</spring.version>
27 <zk.version>7.0.0</zk.version>
28 </properties>
29
30 <build>
31 <finalName>management-training-zk</finalName>
32 <plugins>
33 <plugin>
34 <groupId>org.apache.maven.plugins</groupId>
35 <artifactId>maven-compiler-plugin</artifactId>
36 <version>2.5.1</version>
37 <configuration>
38 <source>${java.version}</source>
39 <target>${java.version}</target>
40 </configuration>
41 </plugin>
42 </plugins>
43 </build>
44
45 <dependencies>
46
47 <!-- Spring IoC support -->
48 <dependency>
49 <groupId>org.springframework</groupId>
50 <artifactId>spring-context</artifactId>
51 <version>${spring.version}</version>
52 </dependency>
53 <dependency>
54 <groupId>org.springframework</groupId>
55 <artifactId>spring-web</artifactId>
56 <version>${spring.version}</version>
57 </dependency>
58
59 <dependency>
60 <groupId>org.apache.velocity</groupId>
61 <artifactId>velocity</artifactId>
62 <version>1.6</version>
63 </dependency>
64
65 <!-- ZK CE requirements -->
66 <dependency>
67 <groupId>org.zkoss.zk</groupId>
68 <artifactId>zk</artifactId>
69 <version>${zk.version}</version>
70 </dependency>
71 <dependency>
72 <groupId>org.zkoss.zk</groupId>
73 <artifactId>zhtml</artifactId>
74 <version>${zk.version}</version>
75 </dependency>
76 <dependency>
77 <groupId>org.zkoss.zk</groupId>
78 <artifactId>zml</artifactId>
79 <version>${zk.version}</version>
80 </dependency>
81 <dependency>
82 <groupId>org.zkoss.zk</groupId>
83 <artifactId>zul</artifactId>
84 <version>${zk.version}</version>
85 </dependency>
86 <dependency>
87 <groupId>org.zkoss.common</groupId>
88 <artifactId>zcommon</artifactId>
89 <version>${zk.version}</version>
90 </dependency>
91 <dependency>
92 <groupId>org.zkoss.common</groupId>
93 <artifactId>zweb</artifactId>
94 <version>${zk.version}</version>
95 </dependency>
96 <dependency>
97 <groupId>org.zkoss.common</groupId>
98 <artifactId>zvel</artifactId>
99 <version>${zk.version}</version>
100 </dependency>
101 <dependency>
102 <groupId>org.zkoss.zk</groupId>
103 <artifactId>zbind</artifactId>
104 <version>${zk.version}</version>
105 </dependency>
106
107 <!-- ZK PE & EE requirements -->
108 <dependency>

```

Últimos Tutoriales del Autor

- » ApacheDS: tests de integración contra un servidor LDAP embebido.
- » ¿Mockear métodos estáticos?, con el soporte de PowerMock.
- » Integración de la gestión de proyecto de Redmine en Eclipse con el soporte de Mylyn.
- » Omnifaces: una librería de utilidades para JSF2
- » JUnit test runners

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.

```

109     <groupId>org.zkoss.zk</groupId>
110     <artifactId>zkplus</artifactId>
111     <version>${zk.version}</version>
112 </dependency>
113
114 <!-- javax activation -->
115 <dependency>
116     <groupId>javax.activation</groupId>
117     <artifactId>activation</artifactId>
118     <version>1.1-rev-1</version>
119 </dependency>
120
121 <!-- ZK Themes -->
122 <dependency>
123     <groupId>org.zkoss.theme</groupId>
124     <artifactId>sapphire</artifactId>
125     <version>${zk.version}</version>
126 </dependency>
127 <dependency>
128     <groupId>org.zkoss.theme</groupId>
129     <artifactId>silvertail</artifactId>
130     <version>${zk.version}</version>
131 </dependency>
132
133 <!-- Java servlet -->
134 <dependency>
135     <groupId>javax.servlet</groupId>
136     <artifactId>javax.servlet-api</artifactId>
137     <version>3.0.1</version>
138     <scope>provided</scope>
139 </dependency>
140
141 <!-- commons -->
142 <dependency>
143     <groupId>commons-io</groupId>
144     <artifactId>commons-io</artifactId>
145     <version>1.3.1</version>
146 </dependency>
147 <dependency>
148     <groupId>commons-logging</groupId>
149     <artifactId>commons-logging</artifactId>
150     <version>1.1.1</version>
151 </dependency>
152 <dependency>
153     <groupId>commons-digester</groupId>
154     <artifactId>commons-digester</artifactId>
155     <version>2.0</version>
156 </dependency>
157 <dependency>
158     <groupId>commons-collections</groupId>
159     <artifactId>commons-collections</artifactId>
160     <version>3.2.1</version>
161 </dependency>
162
163 <!-- Log4j -->
164 <dependency>
165     <groupId>log4j</groupId>
166     <artifactId>log4j</artifactId>
167     <version>1.2.16</version>
168 </dependency>
169
170 <dependency>
171     <groupId>org.slf4j</groupId>
172     <artifactId>slf4j-simple</artifactId>
173     <version>1.6.4</version>
174 </dependency>
175
176 <dependency>
177     <groupId>org.apache.geronimo.ext.tomcat</groupId>
178     <artifactId>juli</artifactId>
179     <version>7.0.23.1</version>
180     <scope>test</scope>
181 </dependency>
182
183 <!-- Rome -->
184 <dependency>
185     <groupId>rome</groupId>
186     <artifactId>rome</artifactId>
187     <version>1.0</version>
188 </dependency>
189
190 </dependencies>
191
192 <repositories>
193 <repository>
194     <id>zk repository</id>
195     <url>http://mavensync.zkoss.org/maven2</url>
196 </repository>
197 <repository>
198     <id>ZK EE Evaluation</id>
199     <url>http://mavensync.zkoss.org/zk/ee-eval</url>
200 </repository>
201 <repository>
202     <id>com.asual.maven.public</id>
203     <name>Asual Public Repository</name>
204     <url>http://www.asual.com/maven/content/groups/public</url>
205 </repository>
206 </repositories>
207
208 </project>

```

No son pocas y solo vamos a usar las de la versión community.

Una vez declaradas y teniendo el soporte de la especificación de servlets 3.0 no necesitaríamos más configuración que incluir el fichero de configuración de zk, zk.xml dentro del directorio WEB-INF, aunque aún vacío.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <zk>
3     <!-- 7.0.0 -->

```

```
4 | </zk>
```

En la dependencia zk-7.0.0.jar ya se incluye el correspondiente web-fragment.xml que configura todo lo necesario para el framework y mapea las siguientes extensiones contra el servlet de entrada de todas las peticiones del framework, siguiendo el patrón front controller:

```
1 | <servlet-mapping>
2 |   <servlet-name>DHtmlLayoutServlet</servlet-name>
3 |   <url-pattern>*.zul</url-pattern>
4 | </servlet-mapping>
5 | <servlet-mapping>
6 |   <servlet-name>DHtmlLayoutServlet</servlet-name>
7 |   <url-pattern>*.zhtml</url-pattern>
8 | </servlet-mapping>
```

Este tipo de configuración automática también la tenemos con JSF 2.0 y cualquier otro framework servlet 3.0 compliant.

4. Internacionalización.

ZK proporciona un soporte nativo para internacionalizar los literales de nuestras interfaces de usuario basado, por defecto, en ficheros de propiedades.

Como decía, por defecto, trata de encontrar un fichero zk-label_lang.properties en el directorio WEB-INF, donde lang es el locale del idioma del cliente. El idioma del cliente se obtiene de forma automática de la configuración del navegador, que llega a través de las cabeceras de la petición http.

Si queremos personalizar el nombre de los ficheros de propiedades o añadir más de uno podemos incluir la siguiente configuración en el fichero zk.xml.

```
1 | <system-config>
2 |   <label-location>/WEB-INF/classes/messages.properties</label-location>
3 | </system-config>
```

Para hacer uso de los mensajes, dentro de las vistas zul podemos hacer referencia, a través del Expression Language de ZK, al objeto labels.

```
1 | <column hflex="2" label="{labels.name}" align="center" />
```

El objeto labels es un mapa, como en JSF, con lo que podemos acceder a las claves vía `{labels['actions.save']}` donde action.save es una clave dentro del fichero.

Si dentro del fichero tuvieramos más de una misma clave con el mismo prefijo separado por puntos, se crearía un nuevo mapa dentro del mapa de labels que permitiría acceder a las claves vía `{labels.actions.save}`.

También podemos recuperar los mensajes desde código java haciendo uso del método estático `getLabel` de la clase `Labels`.

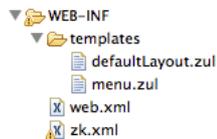
```
1 | Labels.getLabel("name");
```

Si tuviésemos la necesidad de usar una estrategia distinta, como [recuperar los mensajes de internacionalización de una tabla de base de datos](#), al igual que vimos para JSF2, con ZK también podríamos hacerlo implementando un `LabelLocator` propio.

5. Sistema de plantillas.

ZK proporciona un sistema de plantillas propio, similar al de `facelets` para JSF2 que permite definir un layout común para todas nuestras vistas de modo tal que las páginas incrustan contenido en las secciones preparadas para ello de la plantilla. Así las distintas vistas se ocupan de su contenido concreto y todas las secciones comunes se configuran en la plantilla.

Podemos definir una plantilla común incluyendo una página zul en un directorio de templates dentro del de WEB-INF



con un contenido similar al siguiente:

```
1 | <?xml version="1.0" encoding="UTF-8"?>
2 | <zk xmlns:n="native">
3 |
4 |   <n:div id="wrapper">
5 |     <n:div id="header">
6 |       <n:h1>${labels.app.title}</n:h1>
7 |       <n:span id="logo">
8 |         <image src="/resources/img/logo.png" />
9 |       </n:span>
10 |     </n:div>
11 |
12 |     <n:div id="nav">
13 |       <include src="/WEB-INF/templates/menu.zul" />
14 |     </n:div>
15 |
16 |     <n:div id="content">
17 |       <n:h2><div self="@{insert(title)}" /></n:h2>
18 |
19 |       <n:br />
20 |
21 |       <div self="@{insert(content)}">
22 |     </div>
23 |   </n:div>
24 |
25 |   <n:div id="footer">
26 |     <n:small>${labels.app_technology}</n:small>
27 |   </n:div>
28 | </n:div>
29 | </zk>
```

Con las marcas `@{insert(title)}` definimos las regiones de la plantilla que las páginas pueden personalizar, donde `title` o `content` es el nombre de la region. En realidad lo que hacemos es redefinir el contenido de los componentes de zk mediante el atributo `self`, de este modo las páginas definen un contenido que se incrustará en esos componentes, no existe un componente de inserción propiamente dicho.

Las páginas que necesitan hacer uso de la plantilla pueden hacerlo de la siguiente forma:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <?page title="{labels.actions.customers.list}" ?>
3 <?init class="org.zkoss.zk.ui.util.Composition" arg0="/WEB-INF/templates/defaultLayout.zul" ?>
4 <zk>
5   <label self="@define(title)" value="{labels.actions.customers.list}" />
6
7   <div self="@define(content)" apply="org.zkoss.bind.BindComposer"
8     viewModel="@id('vm')" @init('com.autentia.training.masters.views.CustomersView')">
9     <grid id="grid"
10      model="@load(vm.customers)"
11      mold="paging" pageSize="10">
12       <columns>
13         <column hflex="2" label="{labels.name}" align="center" />
14         ...
15       </columns>
16       <template name="model">
17         <row>
18           <label value="@load(each.name)" />
19           ...
20         </row>
21       </template>
22     </grid>
23   </div>
24 </zk>

```

Para hacer uso de la plantilla con la directiva `<?init` y declarando la clase `org.zkoss.zk.ui.util.Composition` podemos pasar como argumento la ubicación de la plantilla. En realidad es la ubicación o ubicaciones, puesto que acepta más de una plantilla.

Con las directivas `@define(title)` incluidas también en el atributo `self` de los componentes definimos el contenido de las secciones. Podemos definir más de un contenido de modo tal que se irán incrustando en los componentes de la plantilla de modo secuencial.

La plantilla no contiene la cabecera de html, puesto que la genera zk, con lo que la primera duda que se nos plantea es cómo redefinir el título de la página. En la plantilla la directiva `<?page title...` no se procesa puesto que no se genera la cabecera de html con lo que tenemos la opción de declararla como una página completa o definir dicha directiva en las páginas que hacen uso de la plantilla. En el ejemplo, hemos optado por la segunda de las opciones.

Por último, en la plantilla:

- estamos combinando componentes de zk con el espacio de nombres por defecto y componentes html nativos con el espacio de nombres "n". ZK tiene un componente `div` que no es más que un wrapper del `div` de html, nos puede interesar hacer uso del componente nativo de html para preparar una maquetación por capas haciendo uso del `id`, puesto que con el componente `div` de ZK ese `id` es autogenerado.
- podemos realizar inclusiones de otras páginas, como la del menú, por claridad en el código y para reutilizarlo.

La página de menú, empezaría a tener un contenido como el que sigue:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <zk>
4   <menubar id="menubar" autodrop="true">
5     <menuItem label="{labels.menu.customers}" href="/pages/masters/customers.zul"/>
6     ...
7   </menubar>
8 </zk>

```

Mi primera vista en ZK comienza a tener el siguiente aspecto, donde el rectángulo rojo es en la región `content` y la región `title` es la azul.

Gestión de la formación



Cientes

Gestión de clientes

Nombre
Quality Objects
Tecnocom
Everis
ICA, Informática y Comunicaciones Avanzadas S.L.
Joelle Web Developers

[1 - 5 / 6]

JPA 2.0, Spring 4.0.0, ZK 7.0.0 CE

Ya estamos usando un componente de tabla y paginación, aunque no es el objetivo de este tutorial; por cierto que el listado de clientes lo he extraído de las empresas que a día de hoy tienen publicada una oferta de trabajo en [infojobs](#) pidiendo experiencia con ZK.

6. Generando la capa de control para mantener el estado y comportamiento de la vista.

Con ZK podemos trabajar de 4 formas distintas:

- generando la vista en ZUL (XML) y la capa de control en java, con dos aproximaciones

- MVC: declarando nuestros controladores como Composers y vinculando los componentes visuales de la vista con componentes java en la capa de control; si quisieramos encontrar algún paralelismo en JSF sería como declarar un binding entre un componente de la vista y un componente de la capa de control, manteniendo en los managed beans una instancia de los componentes visuales haciendo uso de las clases java que los implementan,
- MVVM: sin necesidad de declarar nuestros controladores con ningún tipo de herencia, son POJOs que exponen el estado y el comportamiento; es como nosotros solemos trabajar con los Managed Beans de JSF.
- generando tanto la vista como la capa de control en Java; como cuando en JSF generamos el contenido del árbol de componentes de forma dinámica creando las instancias de los componentes visuales en java y añadiéndolos al árbol,
- generando tanto la vista como la capa de control en ZUL con ZKscript, como si usasemos JSP con scriptlets o JSF 1.2 con JSP y scriptlets; ZK desaconseja esta aproximación por falta de rendimiento.

En nuestra primera vista usaremos el modo MVVM por ser el más similar a cómo venimos trabajando con JSF.

Para declarar un controlador no se usa ninguna anotación ni hay que extender de ninguna clase porque es desde la vista desde la que se declara que quieres vincular un componente, y los componentes hijos de este, al estado y comportamiento de un controlador.

```
1 <div apply="org.zkoss.bind.BindComposer" ?
2   viewModel="@id('vm') @init('com.autentia.training.masters.views.CustomersView')">
```

Con la anotación @id se da un identificador a la instancia del controlador dentro de la sección de componentes de la página zul, y con la anotación @init se hace referencia al controlador, paquete y nombre de la clase (cuidado con las refactorizaciones de paquetes).

Dentro de los componentes hijos podremos hacer referencia a las instancias de los objetos declarados en el controlador con el prefijo de su id declarado en la anotación @id y las siguientes anotaciones, entre otras:

- @load: para mostrar información,
- @save: para almacenar información,
- @bind: para mostrar y almacenar información,

```
1 <grid id="grid" ?
2   model="@load(vm.customers)" >
3
```

- @command: para declarar un evento de acción con un id declarado como un comando en un método del controlador,

```
1 <textbox value="@bind(vm.nameToSearch)"/> ?
2 <button label="Buscar" onClick="@command('search')"/>
3
```

En el controlador, para declarar un método de acción debemos usar la anotación @Command que, por defecto, declara un comando de acción con el nombre del método anotado:

```
1 @NotifyChange("customers") ?
2 @Command
3 public void search() {
4   ...
```

Y con la anotación @NotifyChange({"customers"}) junto al evento declarado en el Controlador con @Command es similar a la declaración de rerender o update que podemos usar con los componentes Ajax de JSF2, de tal manera que indicamos en ese punto qué componentes tienen que repintarse cuando se produce ese evento de acción. La anotación admite un asterisco como parámetro que significaría que se repintarían todos los componentes visuales asociados a ese controlador, como el @form de JSF2.

7. Inyección de dependencias de beans de Spring en la capa de control -ViewModel- de ZK.

ZK proporciona un soporte nativo para enganchar tanto con la inyección de dependencias de Spring como de CDI.

En el caso de Spring, podemos anotar la clase del controlador con @VariableResolver como sigue

```
1 @VariableResolver(org.zkoss.zkplus.spring.DelegatingVariableResolver.class) ?
2 public class CustomersView implements Serializable {
3
4   @WireVariable
5   private CustomerRepository customerRepository;
6
7   ...
8
```

y, como veis, las interfaces de los servicios de Spring con @WireVariable.

También podríamos obtener una instancia de un bean haciendo uso de un método estático de una clase de utilidades de ZK llama SpringUtil.

Con ello, ya tenemos cableados nuestros servicios con la lógica de negocio en la capa de control para poder ser consumidos en las vistas exponiendo y recogiendo información del cliente.

8. Referencias.

- <https://www.google.es/#q=www.adictosaltrabajo.con+zk>
- http://books.zkoss.org/wiki/ZK_Developer's_Reference/Internationalization/Labels
- http://books.zkoss.org/wiki/ZK_Developer%27s_Reference/UI_Patterns/Templating/Composition
- <http://books.zkoss.org/wiki/ZK%20Developer%27s%20Reference/MVVM/Data%20Binding>
- <http://books.zkoss.org/wiki/ZK%20Spring%20Essentials/Working%20with%20ZK%20Spring/Working%20with%20ZK%20Spring%20Core/Using%20Spring%20Va>

9. Conclusiones.

En ocasiones puedes elegir framework y en otras te viene dado, de tal manera que debes intentar adaptar tu manera hacer las cosas al marco tecnológico en el que te encuentres en cada momento, pero siempre desde el punto de vista de las buenas prácticas y patrones de diseño que venías usando.

En este primer tutorial hemos visto como comenzar a usar ZK, desde el punto de vista de los conceptos de JSF pero sin

perder la perspectiva de que estamos trabajando con un framework distinto.

Aún nos queda por cubrir muchos conceptos, stay tuned!

Un saludo.

Jose

jmsanchez@autentia.com

10. ¿Necesitas formación?.

En los cursos de formación que Autentia imparte sobre JSF y donde vemos librerías de componentes (Primefaces, Richfaces, ICEfaces, Apache Suite, ADF,...) aprendemos a diferenciar claramente entre especificación e implementación y las distintas opciones opensource que tenemos en el mercado para JSF, te invito a que le eches un vistazo al [catálogo de cursos de Autentia](#), que siempre estamos dispuestos a ampliar y personalizar con las necesidades de nuestros clientes.

A continuación puedes evaluarlo:

[Regístrate para evaluarlo](#)

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

[Share](#) |

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

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



Esta obra está licenciada bajo licencia [Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

PUSH THIS [Page Pushers](#) [Community](#) [Help?](#)

---- 0 people brought clicks to this page

no clicks + + + + + + + +

powered by [karmacracy](#)

Copyright 2003-2014 © All Rights Reserved | [Texto legal y condiciones de uso](#) | [Banners](#) | [Powered by Autentia](#) | [Contacto](#)

