

¿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)



» Estás en: [Inicio](#) [Tutoriales](#) [Empezar a programar con ZK](#)



Francisco Ferri Pérez

Consultor tecnológico de desarrollo de proyectos informáticos.

Desarrollador de proyectos informáticos, Microsoft Certified IT Professional - Enterprise Administrator

[Ver todos los tutoriales del autor](#)

Catálogo de servicios Autentia



Fecha de publicación del tutorial: 2012-10-05

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

Framework ZK



Empezar a programar (2-5)

Contenido

1. Introducción
2. Declarando una Clase de Dominio
3. Creando el Interfaz de Usuario
4. Referencias

Introducción

Este tutorial es la segunda parte de una serie de cinco:

1. [Configurar el entorno](#)
2. [Empezar a programar](#) (Estás en este)
3. ¿Qué es MVC y MVVM? (no publicado todavía)
4. MVC en ZK (no publicado todavía)
5. MVVM en ZK (no publicado todavía)

Está dirigido a desarrolladores con experiencia en la creación de aplicaciones web en Java. A continuación aprenderás a crear una clase de dominio y tu primera interfaz de usuario con ZK.

A lo largo de este documento hacemos referencias a fuentes externas, las encontrarás todas en la sección de Referencias

Declarando una Clase de Dominio

Lo siguiente es la clase de dominio que representa a un coche.

Extraído de Car.java

```
public class Car {
    private Integer id;
    private String name;
    private String company;
    private String preview;
    private String description;
    private Integer price;
    //omit getter and setter for brevity
}
```

- Para ver el código completo de la clase puedes ir a continuación, a la sección de referencias [3]

Ahora definimos una interfaz para implementarla en una clase de servicio que contendrá la lógica de negocio necesaria en el ejemplo, como buscar los coches.

Extraído de CarService.java

```
public interface CarService {

    /**
     * Retrieve all cars in the catalog.
     * @return all cars
     */
    public List findAll();
}
```

Síguenos a través de:



Últimas Noticias

» ¡¡¡Terrakas 1x04 recién salido del horno!!!

» 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

[Histórico de noticias](#)

Últimos Tutoriales

» Eventos en MySQL

» Plantillas para los métodos equals y hashCode en Eclipse, usando la librería de Apache Commons Lang.

» Como convertir ficheros Flash (.swf) a HTML5

» Tu primer proyecto web con ZK

» Selección manual de idioma en la interfaz de usuario con JSF2.

Últimos Tutoriales del Autor

```

/**
 * search cars according to keyword in name and company.
 * @param keyword for search
 * @return list of car that match the keyword
 */
public List search(String keyword);
}

```

En este ejemplo, hemos definido la clase **CarServeImpl** que implementa la interfaz anterior. Para simplificar, usamos una lista estática de objetos como modelo de datos. Puedes reescribirla para que se conecte a una base de datos en una aplicación real. Los detalles de la implementación no están cubiertos en el ámbito de este artículo, puedes echar una ojeada a continuación en la sección de referencias [4]

Creando el Interfaz de Usuario

Diseñar la interfaz de usuario es un buen comienzo para crear una aplicación, además de que te ayuda a definir el alcance de tu aplicación.

ZK nos provee de cientos de componentes, listos para ser usados en el interfaz del usuario, por lo tanto un desarrollador puede crear la interfaz de usuario que desee combinando y mezclando esos componentes sin tener que crearlos desde 0.

En ZK puedes usar el ZK Markup Language para la creación del Interfaz del usuario (**ZUML**) [5], que es un lenguaje estructurado como un XML, que te permite definir la Interfaz del Usuario.

La convención que se utiliza en ZK es que para los ficheros en formato ZUML utilicemos la extensión **.zul** como sufijo. En los ficheros zul, podemos representar un componente como un elemento del XML (tag) y configurarlo (estilo, comportamiento, funcionalidad) mediante los atributos del elemento XML. [6]

En el case de esta aplicación de ejemplo, primero, queremos diseñar una ventana con un título específico y bordes normales, como si fuera el borde exterior (marco) de la aplicación.

Extraído de search.zul

```

<window title="Search" width="600px" border="normal">
  <!-- put child components inside a tag's body -->
</window>

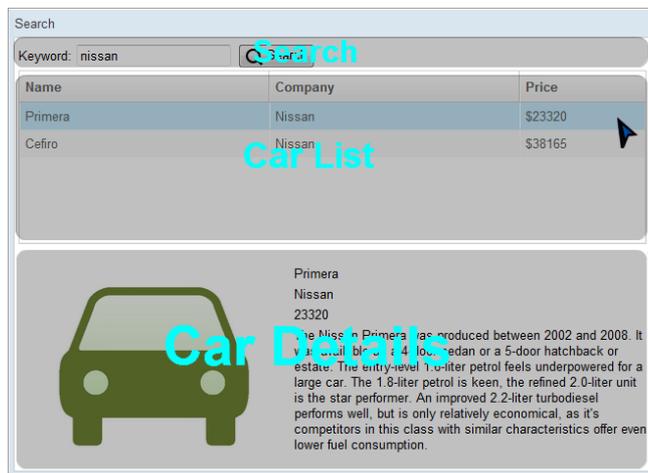
```

Como **"window"** es el componente que contiene al resto, lo llamamos componente raíz o "root". El componente "window" (ventana) normalmente se usa como contenedor de otros componentes, porque en esencia, simplemente muestra una ventana vacía como en una aplicación de escritorio tradicional, y en esta podemos añadir los componentes que queramos.

Los componentes dentro de la ventana los llamaremos hijos o **"child"**, y deben estar dentro del cuerpo del elemento "window".

Por ejemplo, hacemos que aparezca el título de la ventana estableciendo texto en el atributo **"title"** del elemento "window", y hacemos visible el borde de la ventana estableciendo el atributo **"border"**. En el caso del ancho, usaremos el atributo **"width"**, pero en este caso estableceremos el valor de la propiedad CSS, es decir "800px" o "60%".

Básicamente, el interfaz de usuario de nuestra aplicación de ejemplo se divide en 3 áreas dentro de la ventana ("window"), que son (de arriba a abajo): **Área del buscador**, **Área de listado de los coches** y **Área de detalles del coche**.



Área del buscador

Trabajar con los componentes de ZK es como trabajar construir bloques de código, puedes combinar y mezclar componentes que existan (incluso crear los tuyos propios) para crear la interfaz de usuario que desees.

Para permitir a los usuarios buscar, necesitamos un componente que les permita escribir el texto, es decir un "input", y un botón para lanzar la búsqueda.

A continuación vemos un ejemplo de cómo podemos usar algunos componentes simples para cubrir estos requisitos.

Extraído de search.zul

```

<hbox align="center">
  Keyword:
  <textbox id="keywordBox" />
  <button id="searchButton" label="Search" image="/img/search.png" />
</hbox>

```

hbox es un componente contenedor, que ordena horizontalmente los componentes que contenga. Seguro que has adivinado, la "h" de "hbox" significa horizontal. Como los componentes hijos o "child", tienen diferentes tamaños, establecemos el atributo **"align"** con valor **"center"** para que se alineen, entre ellos sobre su línea central.

» Tu primer proyecto web con ZK

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

» Tutorial básico de bases de datos en Java mediante JDBC

» Introducción a bases de datos SQL en Java.

» Introducción a bases de datos NoSQL (Not Only SQL)

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.

En algunos componentes del ejemplo también especificamos un **atributo "id"**, esto nos permitirá referirnos a ellos y por lo tanto poder controlarlos usando el "id". Si quieres convertir el botón en un botón con imagen solo tienes que especificar el path de la imagen en el **atributo "image"** del mismo.

Área de listado de los coches

ZK dispone de muchos componentes para mostrar un conjunto de datos, como por ejemplo los componentes de lista **"listbox"**, malla **"grid"** y árbol **"tree"**. En este ejemplo, hemos elegido usar una lista **"listbox"** para mostrar un listado de coches, con tres columnas: Nombre, Compañía y Precio.

Establecemos el **atributo "height"**, de forma que se mostrarán tantas filas como quepan; puedes navegar con la barra de scroll para ver el resto de filas.

El atributo **"emptyMessage"** se usa para mostrar un mensaje cuando la lista no contiene elementos.

Puesto que el componente de lista **"listbox"** también es un componente contenedor puedes añadirle un componente **"listhead"** para mostrar y definir los nombres de las columnas. También puedes añadirle un componente **"listitem"** para mostrar los datos, y dentro de este componentes **"listcell"**, tantos como columnas hayas definido (en el **"listhead"** por ejemplo).

En el siguiente código de ejemplo usamos **"listcell"** con contenido estático (3 listcells) para mostrar la estructura de un componente **"listitem"**, pero a continuación, te mostramos cómo crear un listitem dinámicamente, de acuerdo a los datos que recibe.

Extraído de search.zul

```
<listbox id="carListbox" height="160px" emptyMessage="No car found in the result">
  <listhead>
    <listheader label="Name" />
    <listheader label="Company" />
    <listheader label="Price" width="20%" />
  </listhead>
  <listitem>
    <listcell label="product name"></listcell>
    <listcell label="company"></listcell>
    <listcell>${label value="price" /}></listcell>
  </listitem>
</listbox>
```

Área de detalles del coche

Al igual que el **"hbox"**, **"vbox"** es un componente que distribuye los componentes hijos "child", pero en este caso en vertical. Combinando estos 2 componentes contenedores, podemos mostrar más información en la pantalla. El **atributo "style"** permite personalizar el estilo del componente escribiendo en él CSS directamente.

Extraído de search.zul

```
<hbox style="margin-top:20px">
  <image id="previewImage" width="250px" />
  <vbox>
    <label id="nameLabel" />
    <label id="companyLabel" />
    <label id="priceLabel" />
    <label id="descriptionLabel" />
  </vbox>
</hbox>
```

Puedes ver el fichero **.zul** completo a continuación en la sección de referencias en el siguiente link [x]

Referencias

- [1] [Página web oficial de ZK](#)
- [2] [Guía de instalación manual de ZK y ejemplos de web.xml \(Servlet 3.0, 2.4 y 2.3\)](#)
- [3] [Car.java](#)
- [4] [CarService.java CarServiceImpl.java](#)
- [5] [ZUML](#)
- [6] [ZK Component Reference](#)
- [7] [Fichero search.zul completo](#)

Este documento es un extracto de la documentación oficial del Framework ZK, traducido y ampliado por Francisco Ferri. Colaborador de Potix (creadores del Framework ZK). Si quieres contactar con él puedes hacerlo en franferri@gmail.com, en twitter [@franciscoferri](#) o en LinkedIn

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**:

» [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

IMPULSA

Impulsores

Comunidad

¿Ayuda?

sin clicks

0 personas han traído clicks a esta página

+ + + + + + + +

powered by [karmacray](#)

Copyright 2003-2012 © All Rights Reserved | [Textos](#) | [Contacto](#) | [Condiciones de uso](#) | [Banners](#) | Powered by Autentia

