autentia
Autentia Real Business Solution, S.L.

# ¿Qué ofrece Autentia?

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

- Desarrollo de componentes y proyectos a medida.
- Auditoría de código y recomendaciones de mejora.
- Arranque de proyectos basados en nuevas tecnologías.
- Recomendaciones arquitectónicas, coaching tecnológico, implantación de Framework, Java,...
- HeadHunting tecnológico.
- Cursos de Formación (impartidos por desarrolladores en activo):
  - Dirección de Proyectos Informáticos.
  - Gestión eficaz del Tiempo.
  - Arquitecturas de desarrollo Web: Web, J2EE, SOA, WebServices, BPM, etc.
  - Java/ J2EE a todos los niveles: JSPs, Servlets, EJBs, JMS, JNI, etc.
  - Análisis y diseño orientado a objetos.
  - UML y patrones de diseño.
  - Buenas prácticas en el desarrollo de aplicaciones.
  - Técnicas avanzadas: Lucene, Hibernate, Spring, JSF, Struts, etc.

Nuestra mejor referencia son los conocimientos que compartimos en nuestra web:

**www.adictosaltrabajo.com**

## Decenas de entidades cuentan ya con nosotros

Para más información visítenos en:
**www.autentia.com**

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**

## XIV Charla Autentia

# Introducción a ZK


con **Francisco Ferri**

**@franciscoferri**

# Bienvenido a la XIV Charla de Autentia

El vídeo, la presentación y el ponente de esta y otras charlas están disponibles en la web de:

**www.AdictosAlTrabajo.com**

**twitter**
**@adictosaltrabaj**
**@franciscoferri**

# ¿Qué es ZK?

ZK es un proyecto libre que nació con el objetivo de:

- Simplificar radicalmente el desarrollo de aplicaciones web

# ¿Quién lo usa?


# ¿Cómo lo encuentro?

**ZK** está disponible para ser descargado en **www.ZKoss.org**, pero además tiene una empresa que lo desarrolla, mantiene y coordina a la comunidad, **POTIX** en su web www.potix.com

# ¿Qué es ZK?

· ZK es AJAX sin escribir JavaScript.

· ZK es un framework de componentes dirigido a través de eventos (Event-Driven). Con él podemos desarrollar interfaces de usuarios de un modo profesional y extremadamente fácil.

· Open Source, pero además detrás tiene el respaldo de una compañía POTIX. Más adelante veremos los diferentes tipos de paquetes que podemos descargarnos.

· Está basado en tecnologías abiertas, con una curva de aprendizaje casi plana:

XHTML (HTML escrito con la sintaxis de XML)

XUL (http://www.mozilla.org/projects/xul/)

· Funciona también con JSP, JSF, Portlet, tecnologías Java EE y se integra con los IDE's más comunes. En el caso de Eclipse por ejemplo con ZK Studio.

· Diseñado para ser Direct RIA (Direct Rich Internet Applications).

# Características

· Diseñado para ser extremadamente ligero:

- Sin dependencias
- No requiere plugins de ningún tipo.

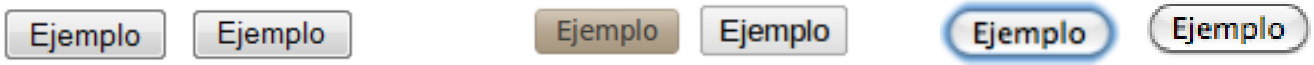
· Compatible con la mayoría de navegadores existentes, incluso legacy (y sin sorpresas):

· ZK también soporta los navegadores de dispositivos móviles, de hecho existe ZK Mobile, que es ZK aplicado al desarrollo de aplicaciones Móviles, accesibles por el navegador de los mismos.

· Se comporta de igual modo en todos los navegadores

· Se renderiza lo mismo para el usuario, es independiente del decorador que utilice el navegador según el sistema operativo.

- Por ejemplo pintando un botón en Mac, Windows o Linux.

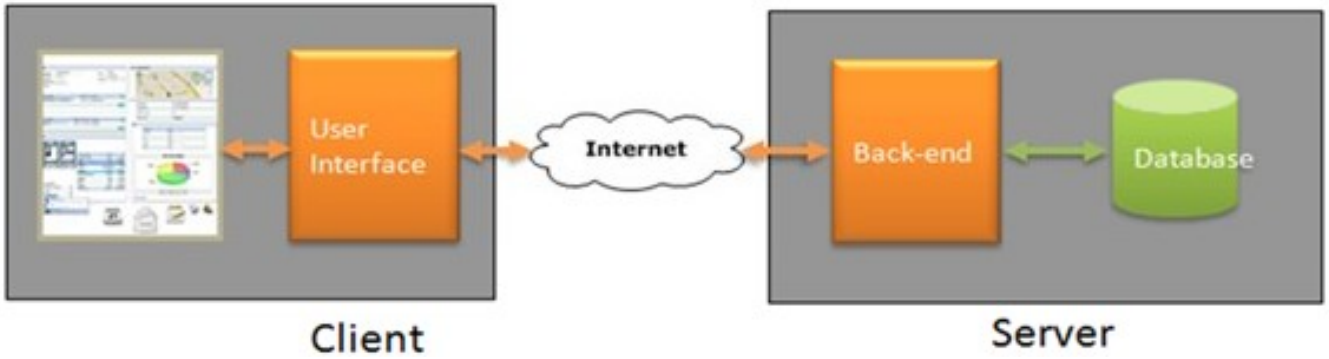



- La decoración de los componentes no depende del sistema operativo. Y es completamente personalizable.

· Los componentes que forman ZK son una representación POJO (Plain Old Java Objects) de todos los componentes XHTML y una batería adicional de todos los componentes del propio ZK. En total unos 200.

# ¿Porqué surgió ZK?

· Antiguamente el desarrollador tenía que controlar incompatibilidades de browsers, la comunicación entre el cliente y servidor, y el rendimiento de la aplicación. Las aplicaciones web limitaban la ubicación del interfaz de usuario al cliente, y la lógica de negocio al servidor.




· Ajax eliminó muchos de los problemas del modelo antiguo y abrió la puerta de las interfaces de usuario ricas al mundo web. Pero manejar Ajax requiere un gran nivel y esfuerzo. Y las implementaciones de Ajax existentes no abstraen al programador del mismo.

¿Porqué no hacer que desarrollar una página web sea como hacer una aplicación de escritorio? Tanto para el usuario como para el desarrollador.

En ZK puede desarrollarse de un modo **Server-Centric, Client-Centric e hibrido**. Pero ZK es Server Centric. Para Client Centric ver: http://books.zkoss.org/wiki/Small_Talks/2009/July/ZK_5.0_and_Client-centric_Approach

# Server-Centric & Client-Centric

La diferencia entre ServerCentric o ClientCentric radica en dónde se está ejecutando la aplicación.

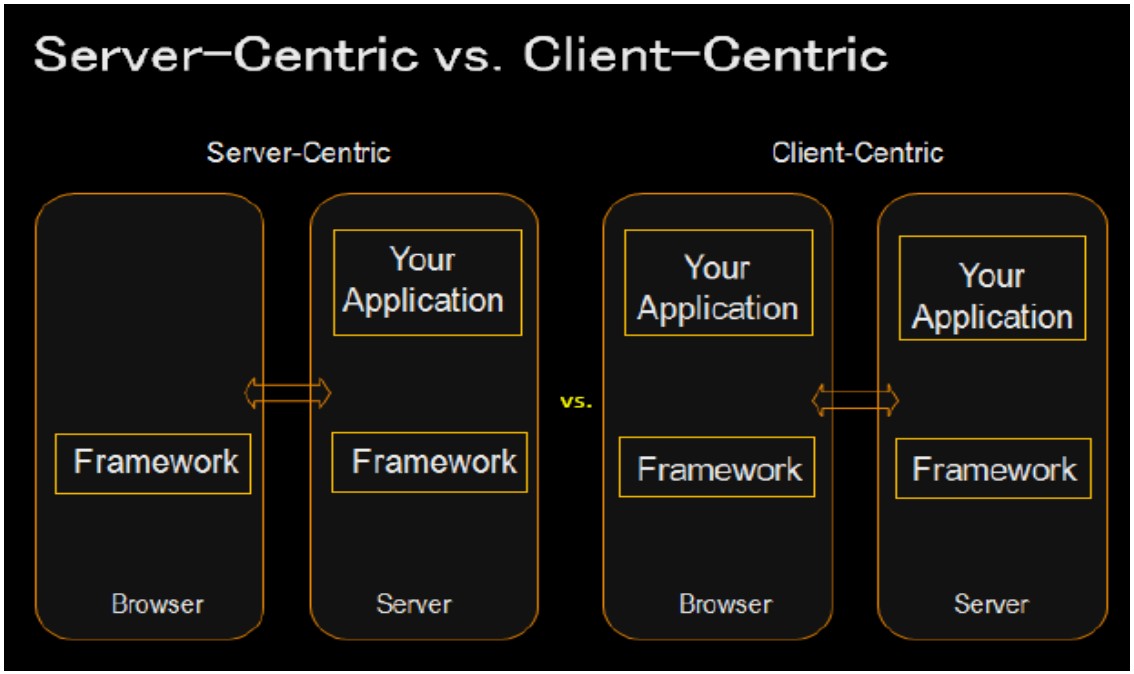
· Server Centric: la aplicación se procesa completamente en el servidor, el cliente sólo se utiliza para mostrar información.

· Client Centric: la aplicación se ejecuta mayormente en el cliente, como por ejemplo hace GWT (Google Web Toolkit)

Tanto ZK como GWT ofrecen una solución al rompecabezasque supone la programación en JavaScript (ajax).

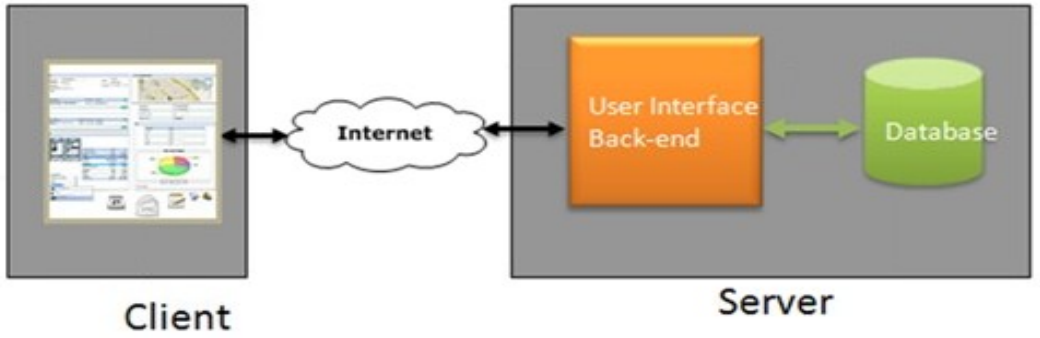
· GWT, convierte Java en JavaScript permitiendo a la aplicación ejecutarse en el browser cliente en vez de en el servidor. Sólo interactua con el servidor cuando necesita recibir datos.

·ZK ejecuta la aplicación en el servidor y ZK es el que se encarga de la capa de presentación.



# Server-Centric

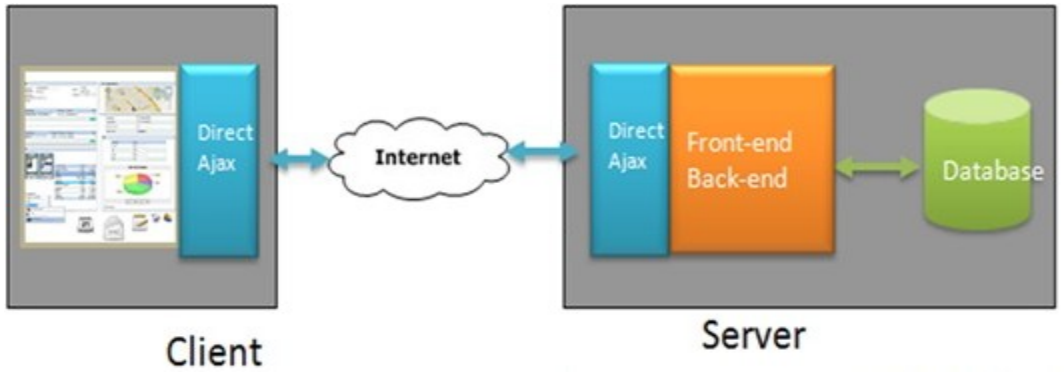
Esta sería la implementación Ajax ideal que permitiera a los programadores centrarse en la lógica de negocio.



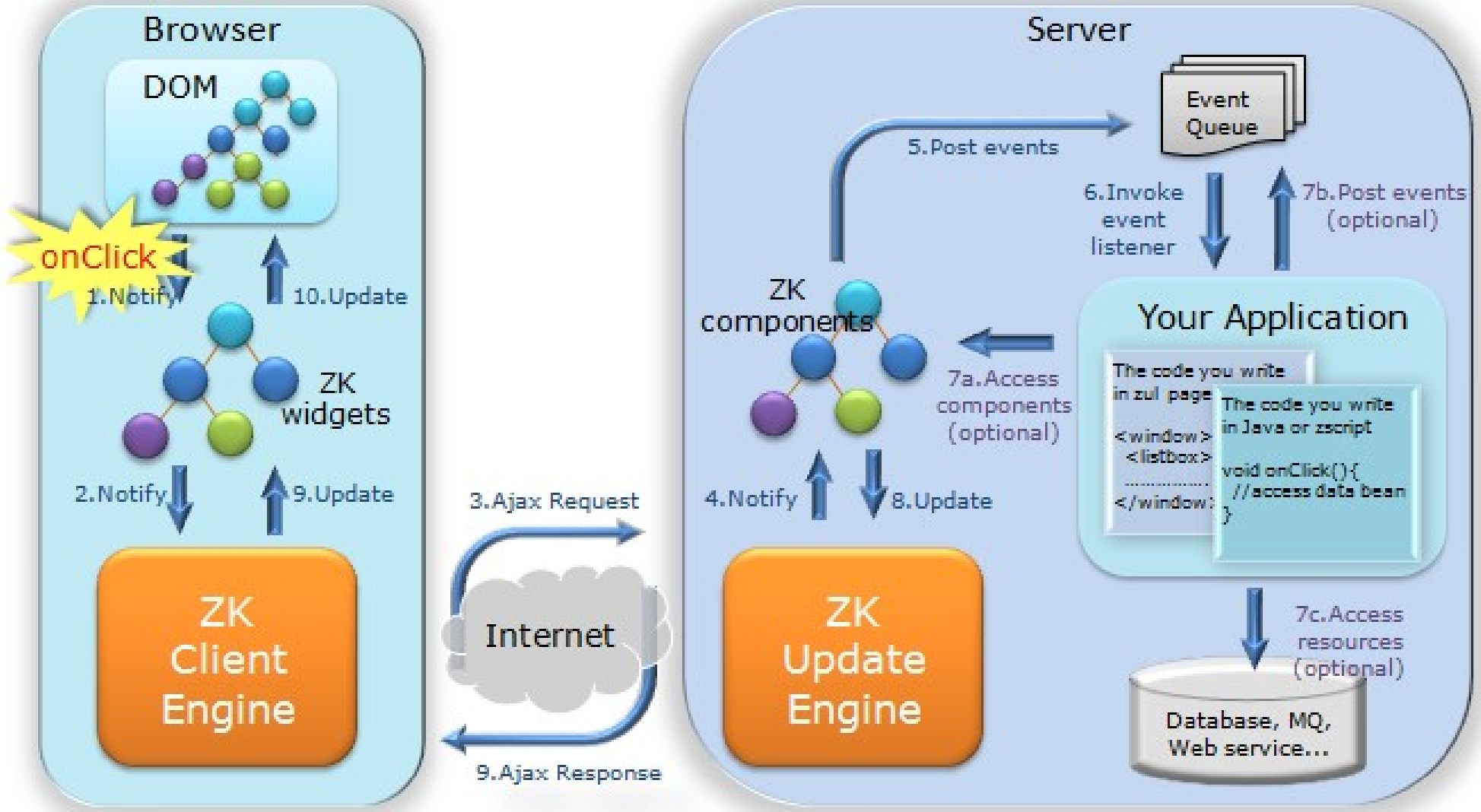

Mapeando todos los componentes y sus eventos de la presentación como POJOs, por ejemplo, como un ORM abstrae de trabajar directamente con tablas de una base de datos.

**Direct R.I.A.**

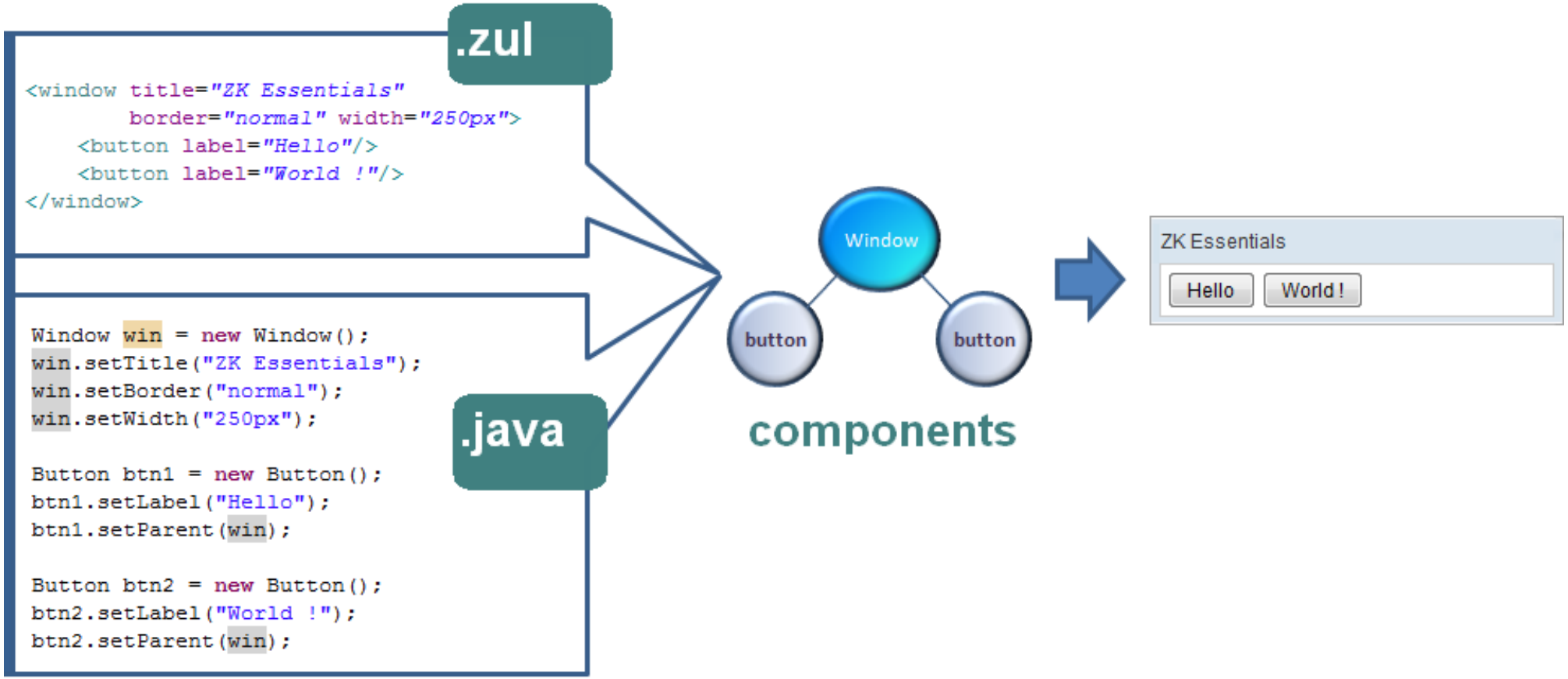
Para conseguirlo ZK inyecta una capa en el cliente de JavaScript que es atendida en el servidor de forma que todos los eventos y procesos son gestionados por ella.



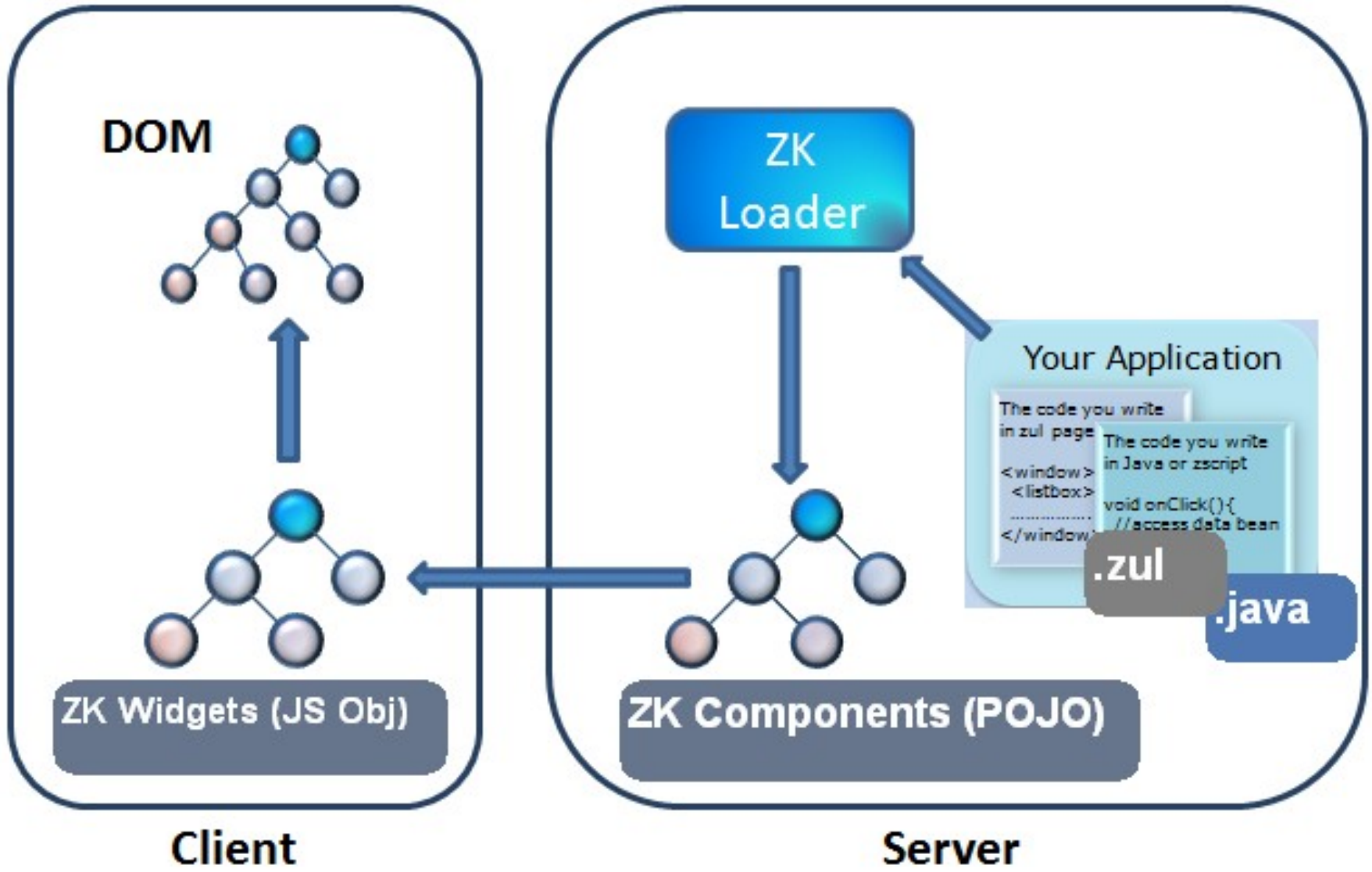
# Arquitectura



# Arquitectura de forma simple (A)



# Arquitectura de forma simple (B)



# ZUML – ZK User Interface Markup Language

ZUML es utilizar la tecnología XUL aplicada al mundo web con algunas extensiones como la posibilidad de incrustar código.

· Com ZUML conseguimos:

Para el usuario la aplicación es como una aplicación de escritorio.

Para el desarrollador es como programar una aplicación standalone (o independiente).

· ZK utiliza el paradigma Event-Driven, para ello encapsula el tráfico request-response y gestiona el JavaScript de forma transparente. Así que el desarrollador no tiene que hacer nada al respecto.

· La sincronización entre cliente y servidor se realiza de forma transparente, automática y bajo demanda según interactua el usuario.

· ZUML contiene UI Components, EL Expressions y ZScript.

· Como ZUML está basado en **XML** podemos extenderlo mediante namespaces.

· Los ficheros ZUML no requieren compilarse como sucedería con una JSP. Más adelante veremos los detalles del rendimiento. Así pues los cambios son imediatos.

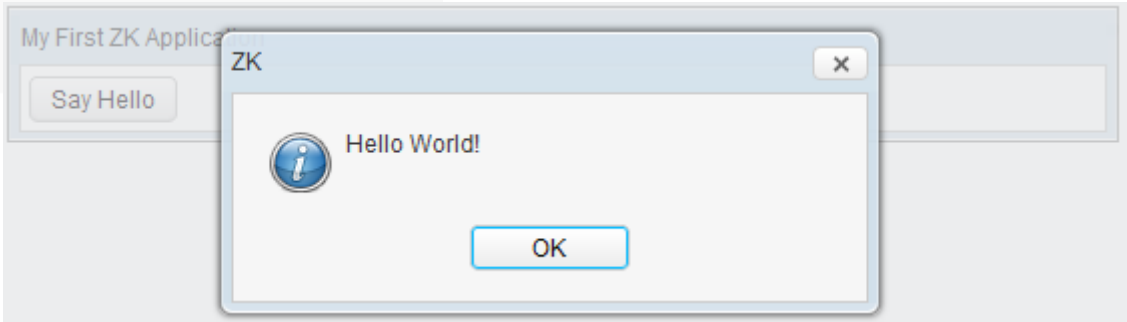
# ZUML – Ejemplo

Ejemplo 1

```
<window title="My First ZK Application" border="normal">
    <label value="Hello World!"/>
</window>
```


Ejemplo 2

```
<window title="My First ZK Application" border="normal">
   <button label="Say Hello" onClick='alert("Hello World!")'/>
   <zscript>
       void alert(String message){ //declare a function
           Messagebox.show(message);
       }
   </zscript>
</window>
```



# ZUML – Ejemplo

Ejemplo 3: conjugando otras tecnologías

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:x="[http://www.potix.com/2005/zul http://www.zkoss.org/2005/zul]"
      xmlns:zk="http://www.zkoss.org/2005/zk">
    <head>
        <title>ZHTML Demo</title>
    </head>
    <body>
        <script type="text/javascript">
        function woo() { //running at the browser
        }
        </script>
        <zk:zscript>
        void addItem() { //running at the server
        }
        </zk:zscript>
        <x:window title="HTML App">
            <input type="button" value="Add Item"
            onClick="woo()" zk:onClick="addItem()"/>
        </x:window>
    </body>
</html>
```

Xmlns

Xmlns:xx

```
<window xmlns:h="http://www.w3.org/1999/xhtml">
    <h:div>
        <button/>
    </h:div>
</window>
```

```
<html xmlns="native" xmlns:u="zul" xmlns:zk="zk">
    <head>
        <title>ZHTML Demo</title>
    </head>
    <body>
        <script type="text/javascript">
        function woo() { //running at the browser
        }
        </script>
        <zk:zscript>
        void addItem() { //running at the server
        }
        </zk:zscript>
        <u:window title="HTML App">
            <input type="button" value="Add Item"
            onClick="woo()" zk:onClick="addItem()"/>
        </u:window>
    </body>
</html>
```

# ZScript

- Sirve para inicializar y declarar variables (incluso globales) y métodos.
- Puede programarse en **Java**, **JavaScript**, **Ruby** (JRuby), **Groovy** (Grails), **Phyton** (Jhyton), etc...
- El código puede estar también separado en ficheros, para aplicar correctamente MVC.

Acceder a un componente sabiendo su ID

Desde ZScript

```
<window id="win_1">
    <zscript><![CDATA[
        win_1.title="given by zscript";
    ]]>
    </zscript>
</window>
```

Desde Java

```
Window win = (Window)Path.getComponent("/win_1");
win.setTitle("given by java");
```

Definiendo una variable y accediendo a ella mediante EL

Desde ZScript

```
<window>
    <zscript><![CDATA[
        var="abc";
    ]]>
    </zscript>
    1:${var}
</window>
```

Desde Java

```
<window id="win_1" use="MyWindow">
    1:${win_1.var}
</window>
```

```
import org.zkoss.zul.Window;

public class MyWindow extends Window {
    String var = "abc";
    public String getVar(){
        return var;
    }
}
```

**Groovy/Grails**: http://blog.zkoss.org/index.php/2009/10/29/securing-your-application-using-zk-grails/
**Phyton(Jhyton)**: http://books.zkoss.org/wiki/Small_Talks/2010/February/Python_With_ZK

# ZScript avanzado

- ZScript por defecto es Java, pero puede cambiarse su idioma mediante el atributo **language**

- Si indicamos en el atributo language javascript, este será inyectado. En el caso de Java, Groovy y Phyton, serán procesados en el servidor

- Podemos agrupar con el atributo **src** todo el código en ficheros.

```
<zk>

<zscript src="PythonForwardComposer.py" language="python"></zscript>
<zscript src="BobController.py" language="python"></zscript>

    <window title="Test Hello Window" border="normal" height="200px"
        apply="${BobController}" width="200px" closable="true"
        sizable="true">
        Hello World Message:
        <label id="lMessage"/>
        <textbox id="txMessage"/>
        <button id="okButton" label="OK"/>
    </window>
</zk>
```

- En una misma página pueden haber varios zscripts en idiomas distintos, podemos combinar Groovy con Ruby, Phyton y Java.

# Características avanzadas 1


En esta presentación no vamos a ver en detalle los siguientes temas, así que únicamente comentarlos:

· **100% Basado en componentes**:

Para el programador, todos los componentes de la interfaz de usuario son POJOS, y son completamente operables desde el API de Java.

Los componentes tienen atributos, 0 o n.

Los componentes tienen 0 o n eventos, que son ejecutados según el usuario interactua.

· **Seguridad**:

- No se expone la lógica de negocio al cliente, o información a internet.

· **Avanzado**:

- ZK selecciona permite configurar el Server Push (basade en Comet) de forma transparente, a su vez escoge la estrategia e implementación del mismo automáticamente.

- ZK permite mediante **CSA** (Client side actions) ejecutar eventos en el cliente. Puesto que no tiene sentido ejecutar un rollover o animación con Ajax. Aún así, CSA permite escuchar en el servidor los eventos (onfocus, onblur, onmouseover...) para trabajar con ellos.

# Características avanzadas 2

· **Extensibilidad**:

- ZK permite crear componentes desde 0 o extenderlos, incluso conjuntos de ellos de varias formas. Directamente en un fichero ZUL, dentro de el mismo, o desde Java.
- ZK Mobile aporta desarrollo para aplicaciones online via Browser.
- **ZK Spring, integra ZK con Spring MVC, Spring Web Flow y Spring Security.**
- **ZK JSP Tags** y **ZK JSF** Components hace posible enriquecer aplicaciones legacy con ZK.
- ZK abstrae de los problemas de compatibilidad entre navegadores, incluso de IE 6.
- ZK Richlets para crear mini-aplicaciones integrables en webs hechas en cualquier tenología.
- ZK con Liferay, con Jboss Seam, JasperReports... etc.
- ZK JSR 299 CDI **:)** http://blog.zkoss.org/index.php/2010/01/07/integrate-zk-and-jsr-299weld/

La documentación es abundante y muy actualizada para las diferentes versiones de los productos de terceros.

· **Accesibilidad**:

- ZK Accesibility
  http://www.zkoss.org/zk508/
- Niveles de conformidad
  http://www.zkoss.org/zk508/levelsOfConformance.htm
- Artículos "How to Make Your AJAX Applications Accessible"
  http://www.zkoss.org/zk508/additionalArticles.htm

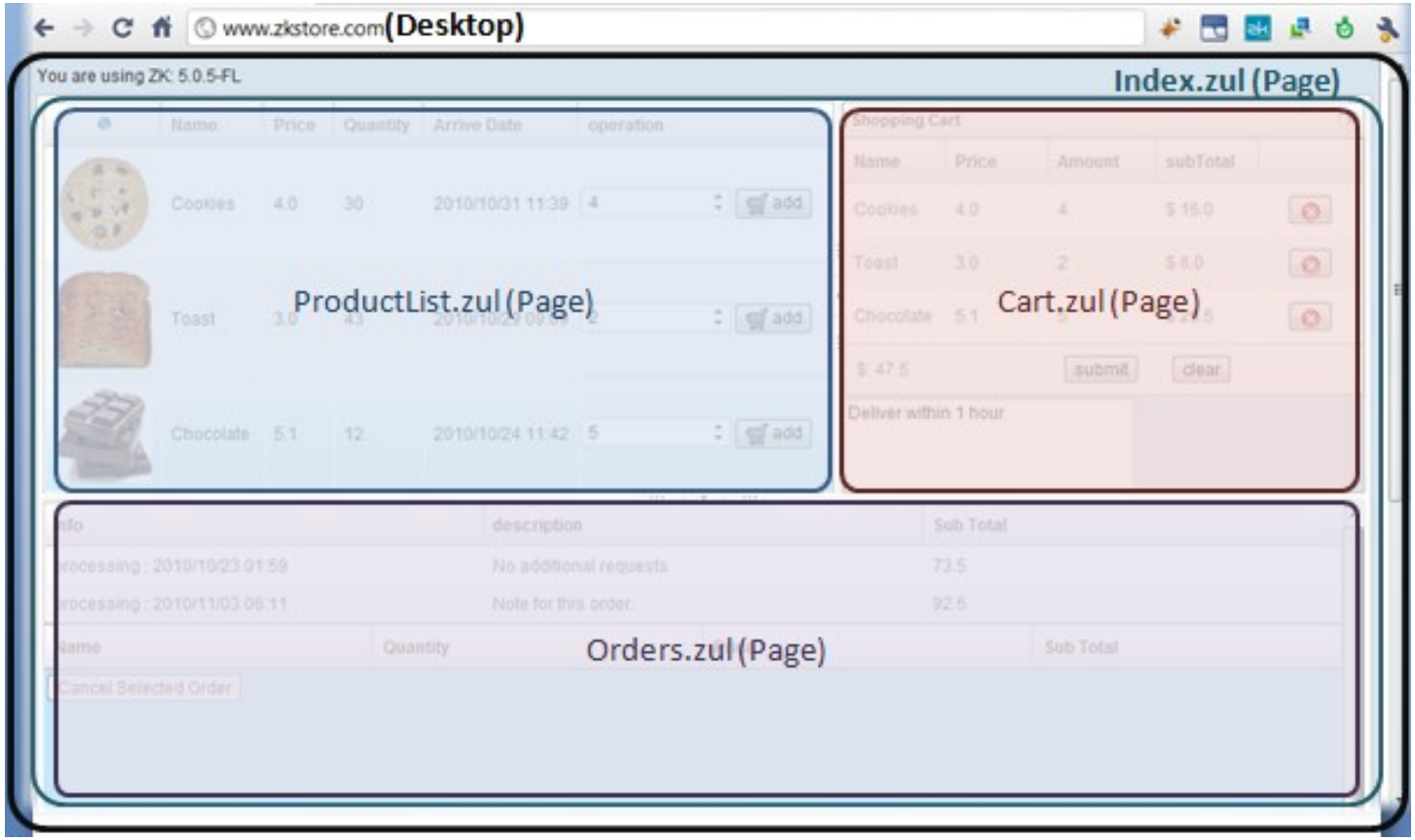
**Todo esto y alguna cosa más quizás para la siguiente charla**

# ¿A quién pertenecen los componentes? (A)

**Los Componentes**

**Las Páginas (Pages)**

**El Escritorio (Desktop)**



# ¿A quién pertenecen los componentes? (B)

**Los componentes**

· Son objetos de la vista que tienen parte en el cliente y parte en el servidor.

· Son las etiquetas, árboles, botones.

· Cuando un componente se crea, su parte visual se añade al browser, a su vez, si se elimina, se elimina también del browser.

· Implementan la interfaz org.zkoss.zk.ui.Component

**Las Páginas (Pages)**

· Son colecciones de componentes.

· Se crean cuando ZK Loader interpreta un fichero ZUML.

· Como los componentes, son completamente dinámicas. Pueden añadirse, quitarse e interactuar con otras páginas.

· Implementan la interfaz org.zkoss.zk.ui.Page

**El Escritorio (Desktop)**

· Es una colección de páginas que sirven a una petición de URL, es decir un escritorio se genera a partir de páginas, cuando pedimos una URL determinada.

· Las páginas pueden añadirse o eliminarse de forma dinámica.

· Implementa org.zkoss.zk.ui.Desktop

# Sesiones, soluciones

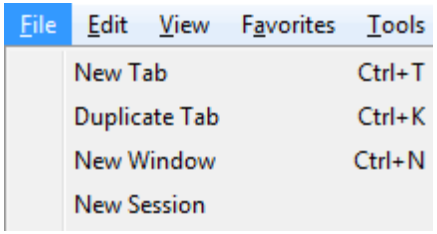
El problema:

· Antiguamente, cada instancia del navegador IE6 replicaba la sesión en curso. Con Mozilla en el mismo caso teníamos sesiones completamente diferentes.

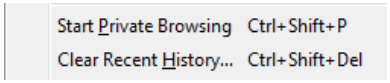
· Hoy en día, los navegadores modernos soportan Tabs, que pueden o no compartir la sesión entre ellas, a su vez, si abrimos otra instancia de un mismo navegador, podemos tener también sesiones compartidas o sesiones aisladas, como si de otro pc se tratara.

· Incluso navegaciones privadas o de incógnito, con un contexto completamente aislado del resto.

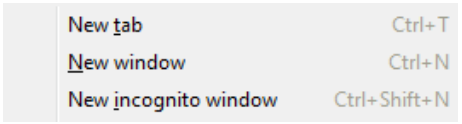
IE 8



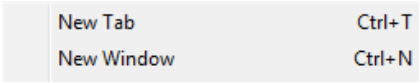

Firefox




Chrome




Safari




Algunos como Chrome, incluso dejan arrastrar tabs entre instancias en ejecución o crear nuevas instancias a partir de un tab ya existente.

Esto supone un problema en el desarrollo del UI y el manejo de sesiones, puesto que no sabemos qué está haciendo el usuario o mejor dicho qué comportamiento espera de la aplicación.

# Sesiones, ámbitos

- Esta distribución nos permite tener varios ámbitos en los que manejar la información:

  · Scope a nivel de aplicación
  · Scope a nivel de escritorio
  · Scope a nivel de página
  · Scope a nivel de componente

- En un diseño inicial de una aplicación sin ninguna característica especial, genérica, encontraríamos algo así:

  · A nivel de **Aplicación**: **- G**uardaríamos cosas como el usuario cuando se loguea.

  · A nivel de **Desktop** en ejecución: - Guardaríamos las cosas que requiera esa pantalla. Y será diferente aunque un nuevo navegador/tab comparta la sesión y pidamos la misma página

  · A nivel de **Page**: - Podemos guardar información local a esa página, y aunque dinámicamente cargemos la página mas veces, estas no se interfieren.

  A nivel de **Componente**: - Podemos guardar información que pueda requerir este, de igual modo que lo anterior. Si por ejemplo queremos declarar un atributo temporal para el mismo.

- Con un simple getAttribute o setAttribute dónde y según nos convenga, puesto que ya viene implementado en las interfaces de ZK

# Características avanzadas 3

Una vez conocemos la arquitectura de ZK comentar que:

· ZK es capaz de comunicarse entre páginas.
Estén o no en el mismo desktop (url)

http://books.zkoss.org/wiki/ZK%20Developer's%20Reference/UI%20Patterns/Communication/Inter-Page%20Communication

· ZK es capaz de comunicarse entre Desktops.
Estén o no en la misma aplicación.

http://books.zkoss.org/wiki/ZK%20Developer's%20Reference/UI%20Patterns/Communication/Inter-Desktop%20Communication

· ZK es capaz de comunicarse entre aplicaciones.
Dentro de un Ear pueden haber varios War o aplicaciones diferentes.

http://books.zkoss.org/wiki/ZK%20Developer's%20Reference/UI%20Patterns/Communication/Inter-Application%20Communication

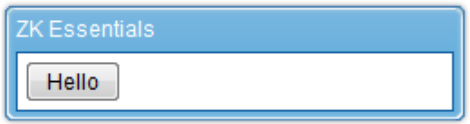
**http://books.zkoss.org/wiki/ZK%20Developer's%20Reference**

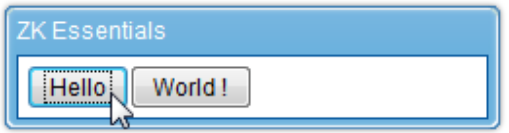
# Programación dirigida a eventos

El detalle del proceso en: http://books.zkoss.org/wiki/ZK%20Essentials/Introduction%20to%20ZK/Event%20Driven%20Programming

## Definiendo eventos en el fichero ZUL

```
<window id="win" title="ZK Essentials" border="normal" width="250px">
    <button label="Hello">
        <attribute name="onClick">
        <![CDATA[
             Button btn = new Button();
             btn.setLabel("World !");
             btn.setParent(win);
        ]]>
        </attribute>
    </button>
</window>
```




## Definiendo eventos en el controlador

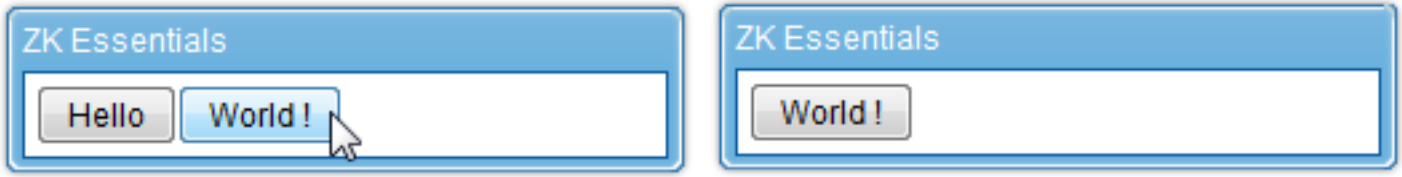
```
<window id="win" title="ZK Essentials" border="normal" width="250px" apply="demo.zkoss.SampleCtrl">
    <button id="helloBtn" label="Hello"/>
</window>
```

```
package demo.zkoss;

import org.zkoss.zk.ui.util.GenericForwardComposer;
import org.zkoss.zul.Button;
import org.zkoss.zul.Window;

public class SampleCtrl extends GenericForwardComposer {

    Window win;

    public void onClick$helloBtn(){
        Button btn = new Button();
        btn.setLabel("World !");
        btn.setParent(win);
    }
}
```

# Ejemplo: Manejando de eventos

## De componentes creados dinámicamente

```
1  public class SampleCtrl extends GenericForwardComposer {
2
3      Window win;
4
5      public void onClick$helloBtn(){
6          Button btn = new Button();
7          btn.setLabel("World !");
8          btn.setParent(win);
9
10         btn.addEventListener("onClick", new EventListener(){
11             public void onEvent(Event event) throws Exception {
12                 win.getFellow("helloBtn").detach();
13             }
14         });
15     }
16 }
```



# Rendimiento

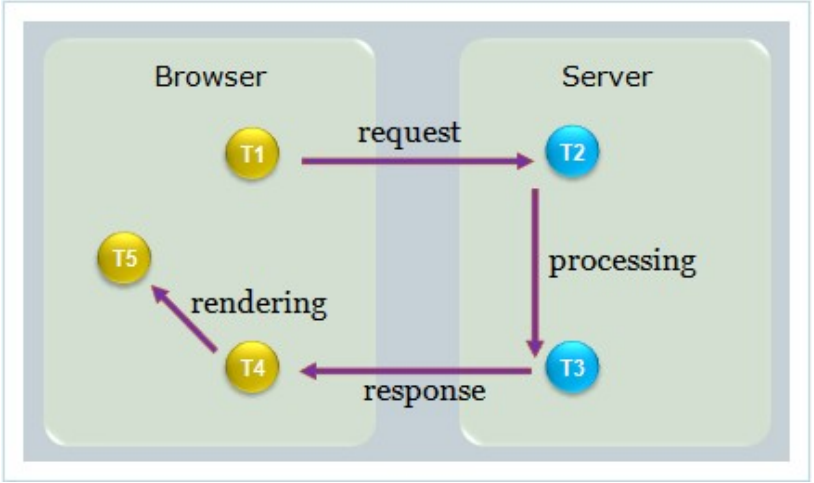
· Como hemos visto, no es necesario compilar ZK como pasa con las JSP's, al igual que no es necesario compilar el HTML, o en PHP. Está basado en tecnología XML.

Qué pasa con el rendimiento:

1. ZK detecta y recarga las páginas modificadas.
2. Convierte estas a un formato intermedio que es procesado muy rápido.
3. Cachea este código intermedio garantizando el rendimiento

· A diferencia de la gran mayoría de frameworks para la vista, ZK permite monitorizar el tiempo de ejecución en cada una de las partes. Y de un modo muy simple y efectivo implementando el interfaz PerfomanceMeter.

T1: PerformanceMeter.requestStartAtClient (String, Execution, long)
T2: PerformanceMeter.requestStartAtServer(String, Execution, long)
T3: PerformanceMeter.requestCompleteAtServer(String, Execution, long)
T4: PerformanceMeter.requestReceiveAtClient(String, Execution, long)
T5: PerformanceMeter.requestCompleteAtClient(String, Execution, long)




**Consejos de rendimiento**: http://books.zkoss.org/wiki/ZK%20Developer's%20Reference/Performance%20Tips
**Monitorización del rendimiento**: http://books.zkoss.org/wiki/ZK%20Developer's%20Reference/Performance%20Monitoring

# ¿Para qué y porqué utilizar ZK?


· Es una plataforma perfecta para montar prototipos y probar código.

· Es completamente factible utilizarlo en entornos altamente explotados por los usuarios.

· Podemos crear simples Richlets web, que son componentes con todo lo necesario para funcionar dentro de otras páginas hechas en cualquier tecnología, respondiendo a una simple url.

· Es una tecnología completamente madura, que existe como tal desde el año 2005 y ha tenido una comunidad que no ha parado de crecer de una forma increíble.

# Descargas y Licencias de ZK

· ZK 3

| | |
|---|---|
| CE/SE - Community Edition/Standard Edition | GPL o Commercial |
| PE - Professional Edition | Commercial |
| EE - Enterprise Edition | Commercial |

· ZK 5

| | |
|---|---|
| **CE/SE - Community Edition/Standard Edition** | **LGPL** |
| PE - Professional Edition | ZOL o Commercial |
| EE - Enterprise Edition | ZOL o Commercial |

GPL – Libre siempre y cuando se utilice en aplicaciones sin ánimo de lucro

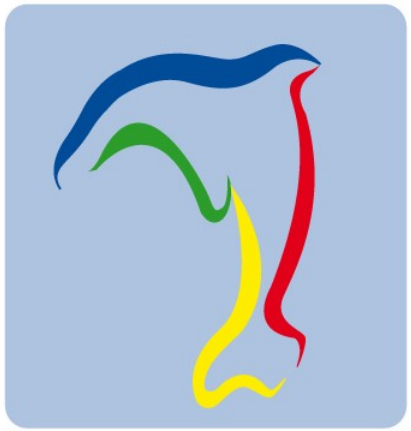
LGPL – Libre aunque se utilice con aplicaciones con ánimo de lucro

ZOL – Completamente comprometidos con OpenSource, si el proyecto es sin ánimo de lucro o compatible con la licencia GPL puede accederse a la licencia completa de ZK

# Tiempo para charlar...


Imagen extraída de la página web oficial de zk: www.zkoss.org

# Gracias



autentia

Autentia Real Business Solutions, S.L.

**Para saber más acerca de nosotros y nuestras charlas:**

| | |
|---|---|
| **Autentia** | **www.autentia.com** |
| **AdictosAlTrabajo** | **www.adictosaltrabajo.com** |

twitter
@adictosaltrabaj
@franciscoferri