Aplicaciones RIA con ZK

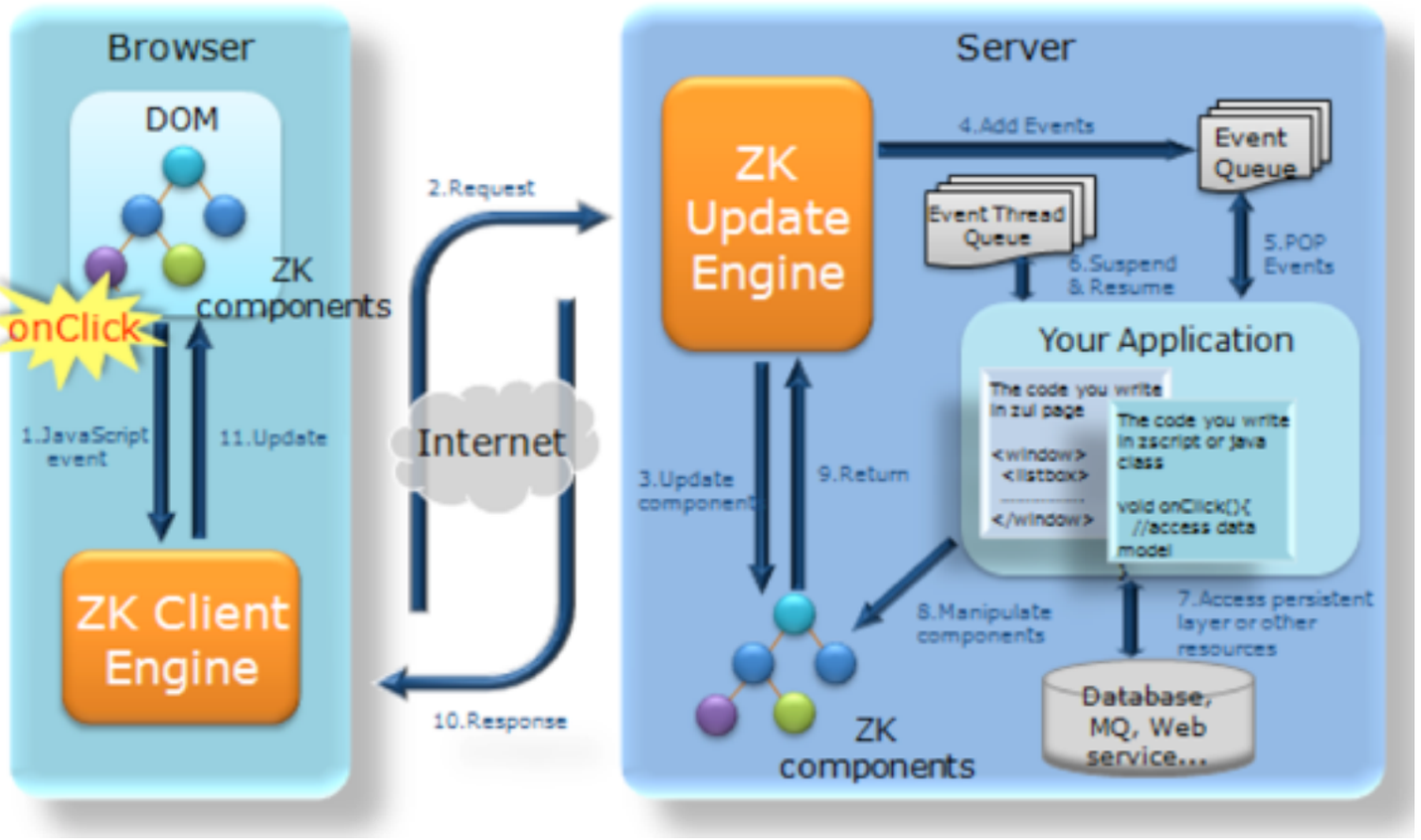
- Sesión 6: Eventos y data binding

Índice

- Eventos
- Data binding
- Aplicación ejemplo

Ciclo de vida de los eventos

- Fase de procesamiento de la petición
- Fase de procesamiento del evento
- Fase de renderizado



Manejando eventos en zscript

- Tres formas de hacer lo mismo:

```
<window>
   <button onClick='alert("here is a zcript")'/>
</window>
```

```
<window>
   <button>
      <attribute name="onClick">
         alert("here is a zscript");
      </attribute>
   </button>
</window>
```

```
<window>
   <zscript>
      public void showAlert(){
         alert("here is a zcript too");
      }
   </zscript>
   <button onClick="showAlert()"/>
</window>
```

Manejadores de eventos en Java (1)

- Utilizando el atributo ZK **use** para especializar un componente

Código Java:

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

public class MyWindow extends Window {
   public void showAlert(){
      try {
         Messagebox.show("handle event in java");
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

Código ZUML:

```
<window id="win_1" use="MyWindow">
   <button onClick="win_1.showAlert()"/>
</window>
```

Manejadores de eventos en Java (2)

- Usando el atributo ZK **apply** para definir un manejador de eventos

Código Java:

```
import org.zkoss.zk.ui.event.Event;
import org.zkoss.zk.ui.util.GenericComposer;
import org.zkoss.zul.Messagebox;

public class MyComposer extends GenericComposer {
    public void onShowAlert(Event evt) {
        try {
            Messagebox.show("handle event in java");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Código ZUML:

```
<window apply="MyComposer">
    <button forward="onShowAlert()"/>
</window>
```

El atributo forward

- Permite redirigir un evento a un componente padre
- Sintaxis:

```
forward="original_event=target_event_expr"
forward="target_event_expr"
```

- Ejemplo:

```
<button forward="onOK"/>
<button forward="onClick=onOK"/>
```

- Es posible reenviar más de un evento:

```
<textbox forward="onChanging=onUpdating, onChange=some.onUpdate"/>
```

Variables en escuchadores de eventos (1)

- En los eventos no es posible utilizar la variable each, porque sólo tiene para construir la página
- Supongamos que queremos colocar un botón en cada uno de los elementos de una colección
- Incorrecto:

```
<zscript><![CDATA[
   String[] countries = {
      "China", "France", "Germany", "United Kindom", "United States"};
]]></zscript>
<hbox>
   <button label="${each}" forEach="${countries}"
      onClick="alert(each)"/> <!-- incorrect!! -->
</hbox>
```

Variables en escuchadores de eventos (2)

- Versión correcta: utilizamos la instrucción ZK **custom-attributes** que nos permite definir un atributo en un componente (cada botón) y lo inicializamos con el valor de ${each} en la construcción de la página

```
<hbox>
   <button label="${each}" forEach="${countries}"
      onClick='alert(self.getAttribute("country"))'>
         <custom-attributes country="${each}"/>
   </button>
</hbox>
```

Data binding

- Muy útil para mantener sincronizadas la vista y el modelo
- Similar al enfoque de JSF
- En la inicialización de la página:

```
<?init
class="org.zkoss.zkplus.databind.AnnotateDataBinderInit" ?>
```

- Sintaxis:

```
 <textbox value="@{person.firstName}"/>
```

Especificación de eventos

- Es necesario definir cuando queremos cargar y salvar los datos:

```
<component-name attribute-name=
    "@{bean-name.attribute-name, save-when='component-id.event-name'}"/>
```

```
<component-name attribute-name=
   "@{bean-name.attribute-name, load-when='component-id.event-name'}"/>
```

Binding de una colección

- Es posible utilizar un data binding con listas con el atributo model:

```
<listbox rows="4" model="@{persons}">
      <listhead>
         <listheader label="First Name" width="100px" />
         <listheader label="Last Name" width="100px" />
         <listheader label="Full Name" width="100px" />
      </listhead>
      <!-- define variable person here-->
      <listitem self="@{each='person'}">
         <listcell>
            <textbox value="@{person.firstName}" />
         </listcell>
         <listcell>
            <textbox value="@{person.lastName}" />
         </listcell>
         <listcell label="@{person.fullName}" />
      </listitem>
   </listbox>
```

Una pantalla completa