



## **Administrador de Contenidos**

**Zwei Admin v 0.9**

**Zweicom S.A**

**París 720, Piso 4. Santiago de Chile**

**Tel: +562 6328415 Fax: +562 6641935**

**Email: [info@zweicom.com](mailto:info@zweicom.com) Home: [www.zweicom.com](http://www.zweicom.com)**

Santiago, 14 de Marzo 2012

## Contenido

Contenido .....	2
1    Control de cambios.....	4
2    Descripción.....	5
2.1    Arquitectura.....	6
3    Plataformas .....	8
3.1    Configuraciones usadas:.....	9
3.1.1    PHP .....	9
3.1.2    Java.....	9
4    Estructura .....	10
4.1    Arbol de directorios .....	10
5    Debug .....	12
6    Componentes XML.....	13
6.1    Introducción .....	13
6.2    Archivo DTD de definición de datos XML.....	15
6.3    Tareas comunes .....	18
6.3.1    Validación en XML con expresiones regulares.....	18
6.3.2    Validación adicional con PHP - Javascript .....	19
6.3.3    Buscadores.....	20

6.3.3.1	Buscador de texto simple .....	20
6.3.3.2	Buscador a partir de tablas.....	21
6.3.3	Selects.....	22
6.3.4	A partir de una lista .....	23
6.3.5	A partir de una tabla.....	23
6.4	Lightbox .....	24
6.5	Gráfico lineal.....	25
7	Ejecutar funciones auxiliares .....	27
8	Settings .....	29

## **1 Control de cambios**

<b>Fecha</b>	<b>Autor</b>	<b>Observaciones</b>
06/09/11	Rodrigo Riquelme	Primera versión del documento

## 2 Descripción

Zwei Admin es un marco de trabajo para el desarrollo de administradores de Back Office, está enfocado a bajar los costos de desarrollo, reduciendo las horas de programación y de debug de las aplicaciones web usando un layout html estándar y componentes personalizados.

Por otra parte se busca estandarizar los artefactos que componen los módulos de manera de separar la lógica del lenguaje de programación en sí, de forma de mantener la misma lógica en cada módulo incluso migrando de lenguaje de programación usando los mismos artefactos o componentes.

Para lograr esto, es que se crea una capa de trabajo basada en **componentes XML** que describan la lógica principal de cada módulo del sistema en forma independiente a los lenguajes de programación utilizados, el único requisito en este caso es trabajar sobre un framework que soporte POO y MVC.

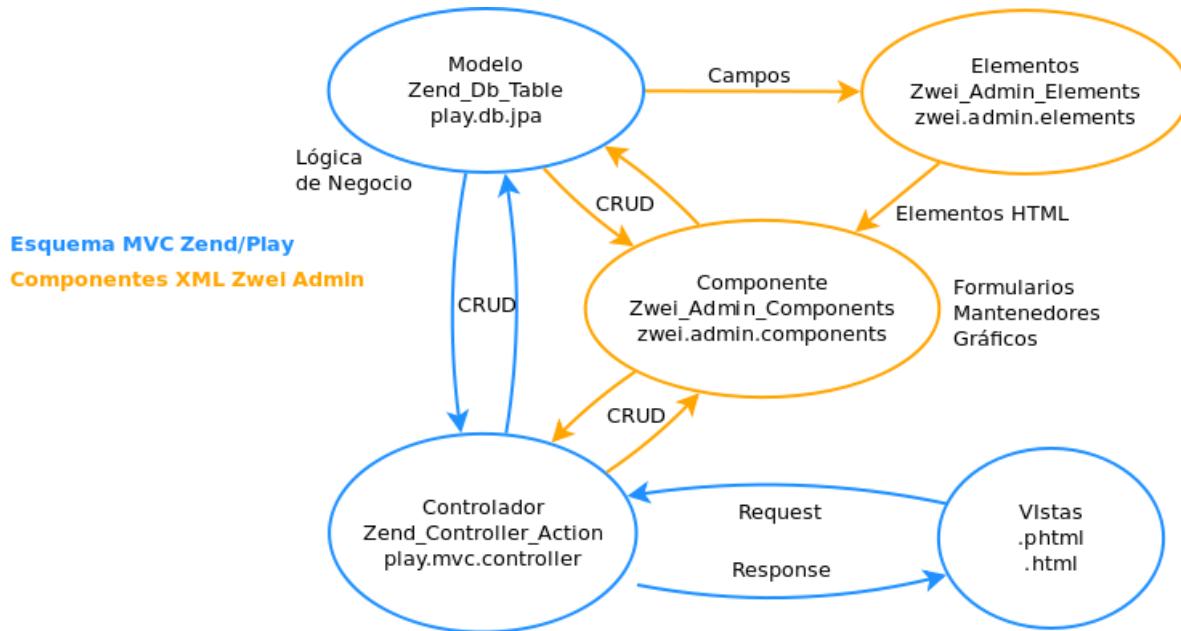
Para modificar estos componentes no se requieren grandes conocimientos para programar en lenguaje cliente (Javascript-Dojo) o servidor (PHP-Java), excepto para crear, extender, modificar o corregir los componentes base que son instanciados en el archivo XML que define al módulo.

También es posible extender la funcionalidad de interfaz web mediante javascript o modificar la lógica de negocio si es necesario personalizando las clases modelo, de forma que el uso de componentes XML no se transforme en rigidez al momento de solucionar requerimientos menos genéricos y más personalizados.

## 2.1 Arquitectura

Este framework está basado en una arquitectura Modelo Vista Controlador que tiene a grandes rasgos la siguiente jerarquía.

1. *Controladores genéricos* en primer lugar y plantillas HTML como Vistas, esto corresponde al esquema clásico MVC de **PHP Zend Framework** y **Java Play Framework** sobre los cuales se comenzado a implementar este desarrollo, estos frameworks están inspirados en la arquitectura MVC de Ruby on Rails .
2. Sobre estos controladores genéricos son parseados los *Componentes XML*, estos hacen de controladores secundarios que unen los modelos con componentes de la librería Zwei Admin, en su mayoría son inputs (por ejemplo textfields, selects, checkboxes) para la configuración de los módulos para las operaciones de crear, listar, modificar y eliminar (CRUD).



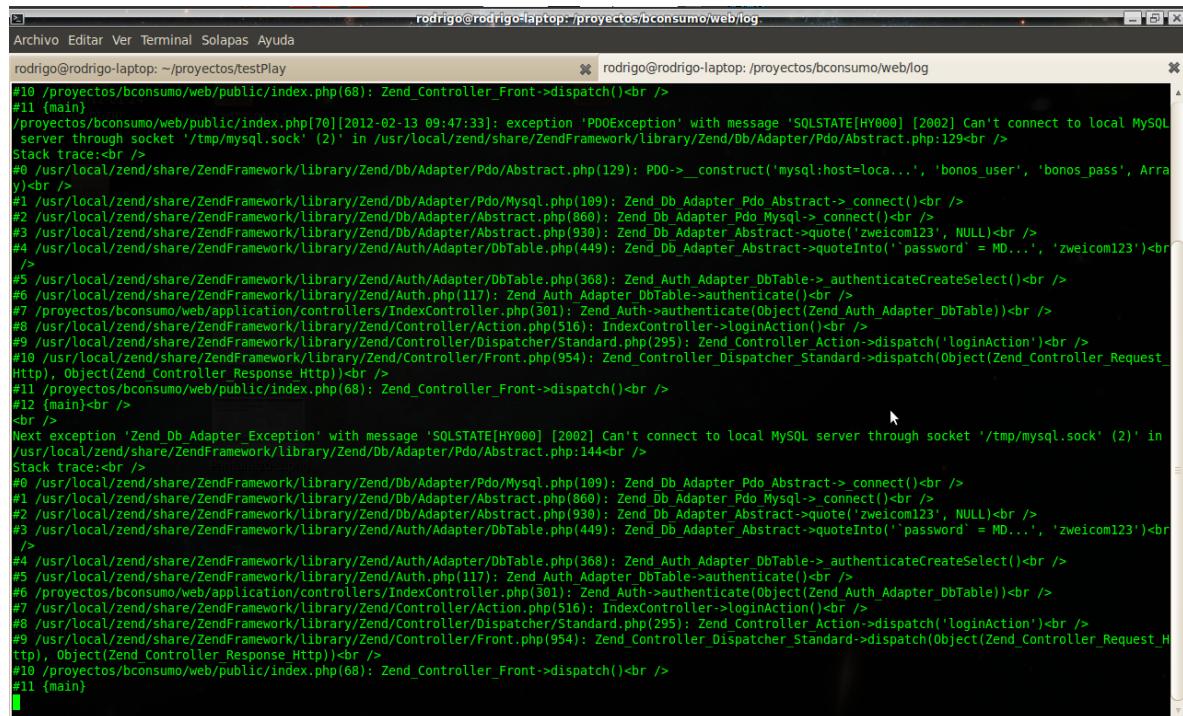
Es importante recalcar que si bien la parte más recurrente en la configuración de estos componentes XML es resolver operaciones CRUD como en este ejemplo, el uso de componentes XML no está limitado a resolver ese tipo de problemas y de hecho la relación de los Componentes con Modelos y Elementos es opcional.

Siempre se pueden crear nuevos tipos de componentes en la librería Zwei Admin Components en caso de necesitar una nueva solución que no corresponda a algo hecho antes, al igual que los elementos Zwei Admin Elements.

La idea de usar components XML es transformar una solución específica en una solución genérica, esto generalmente implica un mayor costo de desarrollo para la primera implementación de un nuevo componente, pero los costos bajan dramáticamente para la reutilización. Es decir, este enfoque busca **reducir la deuda técnica** al mínimo.

Otro punto importante en la filosofía de desarrollo es facilitar la **depuración** y un seguimiento adecuado de errores y del comportamiento de la aplicación tanto en ambientes de pruebas como en producción, y poder hacer este seguimiento en tiempo de ejecución sin interferir en la experiencia del usuario.

### 3



A screenshot of a terminal window titled "rodrigo@rodrigo-laptop: ~/proyectos/bconsumo/web/log". The window shows a stack trace of a PDOException. The error message is "exception 'PDOException' with message 'SQLSTATE[HY000] [2002] Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)'". The stack trace starts at index.php(68) and goes up to main(). The terminal window has a dark background with light-colored text.

```
rodrigo@rodrigo-laptop: ~/proyectos/bconsumo/web/public/index.php(68): Zend_Controller_Front->dispatch()  
#11 {main}  
/proyectos/bconsumo/web/public/index.php[70][2012-02-13 09:47:33]: exception 'PDOException' with message 'SQLSTATE[HY000] [2002] Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)' in /usr/local/zend/share/ZendFramework/library/Zend/Db/Adapter/Pdo/Abstract.php:129  
Stack trace:  
#0 /usr/local/zend/share/ZendFramework/library/Zend/Db/Adapter/Pdo/Abstract.php(129): Pdo->__construct('mysql:host=loca...', 'bonos_user', 'bonos_pass', Array)  
y)->br />  
#1 /usr/local/zend/share/ZendFramework/library/Zend/Db/Adapter/Pdo/Mysql.php(109): Zend_Db_Adapter_Pdo_Abstract->connect()  
#2 /usr/local/zend/share/ZendFramework/library/Zend/Db/Adapter/Abstract.php(860): Zend_Db_Adapter_Pdo_Mysql->connect()  
#3 /usr/local/zend/share/ZendFramework/library/Zend/Db/Adapter/Abstract.php(930): Zend_Db_Adapter_Abstract->quote('zweicom123', NULL)  
#4 /usr/local/zend/share/ZendFramework/library/Zend/Auth/Adapter/DbTable.php(449): Zend_Db_Adapter_Abstract->quoteInto(['password' = MD..., 'zweicom123'])  
/>>  
#5 /usr/local/zend/share/ZendFramework/library/Zend/Auth/Adapter/DbTable.php(368): Zend_Auth_Adapter_DbTable->authenticateCreateSelect()  
#6 /usr/local/zend/share/ZendFramework/library/Zend/Auth.php(117): Zend_Auth_Adapter_DbTable->authenticate()  
#7 /proyectos/bconsumo/web/application/controllers/IndexController.php(301): Zend_Auth->authenticate(Object(Zend_Auth_Adapter_DbTable))  
#8 /usr/local/zend/share/ZendFramework/library/Zend/Controller/Action.php(516): IndexController->loginAction()  
#9 /usr/local/zend/share/ZendFramework/library/Zend/Controller/Dispatcher/Standard.php(295): Zend_Controller_Action->dispatch('loginAction')  
#10 /usr/local/zend/share/ZendFramework/library/Zend/Controller/Front.php(954): Zend_Controller_Dispatcher_Standard->dispatch(Object(Zend_Controller_Request_Http), Object(Zend_Controller_Response_Http))  
#11 /proyectos/bconsumo/web/public/index.php(68): Zend_Controller_Front->dispatch()  
#12 {main}  
<br />  
Next exception 'Zend_Db_Adapter_Exception' with message 'SQLSTATE[HY000] [2002] Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)' in /usr/local/zend/share/ZendFramework/library/Zend/Db/Adapter/Pdo/Abstract.php:144  
Stack trace:  
#0 /usr/local/zend/share/ZendFramework/library/Zend/Db/Adapter/Pdo/Mysql.php(109): Zend_Db_Adapter_Pdo_Abstract->connect()  
#1 /usr/local/zend/share/ZendFramework/library/Zend/Db/Adapter/Abstract.php(860): Zend_Db_Adapter_Pdo_Mysql->connect()  
#2 /usr/local/zend/share/ZendFramework/library/Zend/Db/Adapter/Abstract.php(930): Zend_Db_Adapter_Abstract->quote('zweicom123', NULL)  
#3 /usr/local/zend/share/ZendFramework/library/Zend/Auth/Adapter/DbTable.php(449): Zend_Db_Adapter_Abstract->quoteInto(['password' = MD..., 'zweicom123'])  
/>>  
#4 /usr/local/zend/share/ZendFramework/library/Zend/Auth/Adapter/DbTable.php(368): Zend_Auth_Adapter_DbTable->authenticateCreateSelect()  
#5 /usr/local/zend/share/ZendFramework/library/Zend/Auth.php(117): Zend_Auth_Adapter_DbTable->authenticate()  
#6 /proyectos/bconsumo/web/application/controllers/IndexController.php(301): Zend_Auth->authenticate(Object(Zend_Auth_Adapter_DbTable))  
#7 /usr/local/zend/share/ZendFramework/library/Zend/Controller/Action.php(516): IndexController->loginAction()  
#8 /usr/local/zend/share/ZendFramework/library/Zend/Controller/Dispatcher/Standard.php(295): Zend_Controller_Action->dispatch('loginAction')  
#9 /usr/local/zend/share/ZendFramework/library/Zend/Controller/Front.php(954): Zend_Controller_Dispatcher_Standard->dispatch(Object(Zend_Controller_Request_Http), Object(Zend_Controller_Response_Http))  
#10 /proyectos/bconsumo/web/public/index.php(68): Zend_Controller_Front->dispatch()  
#11 {main}
```

## Plataformas

Al momento de escribir este capítulo la única versión estable es la basada en **Zend Framework PHP**, paralelamente es esta desarrollando una versión Java – Play Framework para aprovechar algunas ventajas de Java como la portabilidad de la capa de datos mediante JPA .

Esta documentación se hizo basada en la última versión de Zwei Xml Admin, la cual a la fecha está implementada en Bonos de Consumo, por lo cual es aconsejable tomarlo como referencia de lo que se explica en este documento

Los modelos se manejan en PHP con **Zend\_Db\_Table** y en Java con **JPA**.

La capa de **presentación** en Play se maneja con **Groovy**, en Zend con archivos **.phtml** (php mezclado con html).

### **3.1 Configuraciones usadas:**

#### **3.1.1 PHP**

**PHP versión:** = 5.2, 5.3 (idealmente usar la versión superior).

**Zend Framework versión:** 1.11

**Dojo Toolkit versión:** 1.6.

**Apache versión:** 2 con mod\_rewrite habilitado.

**MySQL versión:** >= 5.1 (En teoría podría funcionar con otro RDBMS pero no ha sido probado).

#### **3.1.2 Java**

**Java versión:** >= 5.

**Play Framework versión:** 1.0.

**Application Servers:** Getty (integrado con Play Framework), Tomcat, Glassfish.

**MySQL versión:** >= 5.1 (En teoría podría funcionar con otro RDBMS pero no ha sido probado) .

## 4 Estructura

### 4.1 Arbol de directorios

Usamos una implementación de ZF basada en 5 carpetas principales, estas pueden ser creadas a mano o mediante la herramienta **zftool** para generar una estructura de proyectos ZF por consola mediante el comando “zf create project {Nombre del proyecto}”.

Esta misma estructura es la ocuparemos en Java-Play Framework, con la diferencia de que la carpeta application será llamada app.

**application:** Contiene las clases que componen el MVC de la aplicación y la configuración general del sitio.

**application/components:** Contiene los componentes XML para los listados y formularios de back office, cada archivo XML corresponde a un módulo, como ayuda para autocompletar usamos un archivo de definición de datos llamado components.dtd el cual sirve como referencia para usar con IDEs o editores de texto para ejecutar las funciones de autocompletar y facilitar su configuración.

**application/configs:** El archivo a considerar acá se llama application.ini, este contiene los strings de conexión y rutas a usar por la aplicación.

**application/models:** Contienen las clases modelo de la aplicación, para que las interprete correctamente Zend Autoloader deben comenzar con mayúscula y terminar con el sufijo “Model”.

**application/controllers:** Contienen las clases controladores de la aplicación, para que las interprete correctamente Zend Autoloader deben comenzar con mayúscula y terminar con el sufijo “Controller”.

**application/views:** Contienen los layouts y templates de la aplicación.

**docs:** Acá se deja la documentación técnica del proyecto como configuración sugerida del apache, scripts SQL y toda la información del proyecto que se considere relevante.

**library:** Almacenan las bibliotecas que usa la aplicación.

**library/Zwei:** Acá se almacena el core de las librerías que no pertenecen a funcionalidades del MVC Zend, aunque puede hacer uso de estas.

**library/Zwei/Admin:** Contiene las clases que componen el Core de este proyecto y fundamentalmente lo que hace funcionar a los components XML.

**library/Zwei/Db:** Contiene las clases que extienden Zend\_Db para a su vez extender la funcionalidad de todas las clases Modelo usadas en la aplicación.

**public:** Acá está el controlador principal de la aplicación (index.php) que procesa todas las solicitudes, y donde se configuran funcionalidades como autoload de ZF, no debiera variar mucho entre distintos proyectos por lo que se puede copiar y pegar el index.php de un proyecto anterior y modificarlo si es necesario.

Acá tambien debe ir el .htaccess, favicons, css, js, imágenes y todo lo que se necesite ser entregado directamente al navegador cliente.

## 5 Debug

Los lógica de Debug del framework está en la clase `Zwei_Utils_Debug`, por defecto debe crearse una carpeta llamada `/log` en la raíz del sitio y crear los archivos de texto ‘debug’ y ‘transactions’ y opcionalmente un archivo error (para el log de Apache) todos esos archivos deben tener permisos de escritura por Apache.

Para hacer un seguimiento de eventos de debug en tiempo real, debemos movernos a la carpeta `/log` en consola y ejecutar:

```
tail -f debug
```

Se desplegarán las advertencia capturadas por el framework además de todos los mensajes capturados con el comando PHP

```
Zwei_Utils_Debug::write($mensaje);
```

Cuando este se encuentre en la ejecución de la aplicación

Tambien se puede realizar un Debug opcional configurable en el admin, asociado a el módulo Settings (ver capítulo Settings) con este script PHP.

```
Zwei_Utils_Debug::writeBySettings($mensaje, $idSettings, $settingsValue);
```

`$idSettings`: campo “id” de tabla settings, string alfanumérico ,

`$settingsValue`: opcional, campo “value” esperado de table settings para escribir en log, en caso de no existir este parámetro buscará el string por defecto “SI”.

## 6 Componentes XML

### 6.1 Introducción

La mayor parte de la lógica de los módulos está contenida en archivos XML que actúan básicamente como componentes que n diferentes clases que juntas forman un módulo, por ejemplo Módulo Usuarios o Módulo Promociones.

Estos componentes XML están compuestos por un componente principal el cual está indicado en el nodo padre, nodo *component*, y componentes secundarios en los nodos hijos, nodos *element*. Tambien puede tener nodos secundarios con el atributo *tab* para separar los *element* en pestañas.

El componente principal está en el package Zwei\_Admin\_Components

Los componentes secundarios están en el package Zwei\_Admin\_Elements

Ejemplo componente XML (franjas-horarias.xml)

```
<?xml version="1.0"?>
<!DOCTYPE section PUBLIC "-//COMPONENTS// "components.dtd">
<component name="Franja Horaria" type="table_doj0" js="franjashorarias.js"
search_table="evento" edit_additional_validation="true" search_table_field="descripcion"
search_table_target="evento_id" target="franja_horaria" list="false" edit="true" add="true"
delete="true">
    <element name="ID" target="id" type="id_box" visible="false" edit="false" add="false"/>
    <element name="" target="evento_id" type="hidden" visible="false" edit="true" add="true"/>
    <element name="D&iacute;as" target="dias_aplicacion" formatter="formatDias"
type="dias_multi_select" edit_custom_display="true" required="true" trim="true" visible="true"
edit="true" add="true">
        <element name="Hora Inicio" target="hora_inicio" type="dojo_validation_textbox"
reg_exp="(([0-1][0-9])|([2][0-3]):)([0-5][0-9]):([0-5][0-9])" prompt_message="Hora en formato
HH:MM:SS, desde 00:00:00 hasta 23:59:59" invalid_message="Hora en formato HH:MM:SS, desde
00:00:00 hasta 23:59:59" required="true" trim="true" visible="true" edit="true" add="true"/>
        <element name="Hora Fin" target="hora_fin" type="dojo_validation_textbox" reg_exp="(([0-
1][0-9])|([2][0-3]):)([0-5][0-9]):([0-5][0-9])" prompt_message="Hora en formato HH:MM:SS, desde
00:00:00 hasta 23:59:59" invalid_message="Hora en formato HH:MM:SS, desde 00:00:00 hasta
23:59:59" required="true" trim="true" visible="true" edit="true" add="true"/>
    </component>
```

Los nombres de las clases están en el atributo “type” de cada nodo.

En este ejemplo el “type” del componente principal (nodo *component*) es igual a “table\_doj0” por lo tanto su lógica genérica está en la clase *Zwei\_Admin\_Components\_TableDojo*.

*Los “type” de los elementos auxiliares (nodos element) son una referencia a los tipos de entrada de estos, en este ejemplo los types iguales a “id\_box”, “dojo\_validation\_textbox”, “dojo\_filtering\_select”, “dojo\_yes\_no”, hacen referencias a las clases Zwei\_Admin\_Elements\_IdBox, Zwei\_Admin\_Elements\_DojоФilteringSelect y Zwei\_Admin\_Elements\_DojoYesNo respectivamente.*

La idea que hay detrás de empaquetar la lógica de estos diferentes elementos en XML es minimizar las tareas repetitivas de copiar y pegar código, reduciéndola a reutilizar archivos XML estos XML a su vez generan la mayor parte de código necesario para funcionar.

## 6.2 Archivo DTD de definición de datos XML

La estructura de los archivos XML está resumida en este archivo de definición de datos, el cual sirve como referencia global y para la tarea de autocompletar con la ayuda de un IDE o editor XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT component (element*,tab*)>
<!ATTLIST component
    name CDATA #IMPLIED
    type (table | table_dojo | settings | settings_dojo | php_info | curl_dojo | google_chart)
#REQUIRED
    functions CDATA #IMPLIED
    excel (true | false) #IMPLIED
    change_password (true | false) #IMPLIED
    functions_permissions (list | add | edit | delete) #IMPLIED
    table_dojo_type (dojox.grid EnhancedGrid) #IMPLIED
    plugins CDATA #IMPLIED
    id CDATA #IMPLIED
    target CDATA #IMPLIED
    search CDATA #IMPLIED
    search_type CDATA #IMPLIED
    search_dojo_type CDATA #IMPLIED
    search_invalid_message CDATA #IMPLIED
    search_prompt_message CDATA #IMPLIED
    search_constraints CDATA #IMPLIED
    search_table CDATA #IMPLIED
    search_table_field CDATA #IMPLIED
    search_table_target CDATA #IMPLIED
    search_display CDATA #IMPLIED
    search_additional_validation (true) #IMPLIED
    edit_additional_validation (true) #IMPLIED
    inherits_data (true) #IMPLIED
    cols CDATA #IMPLIED
    titles CDATA #IMPLIED
    items CDATA #IMPLIED
        links CDATA #IMPLIED
        popups CDATA #IMPLIED
        popups_iframe CDATA #IMPLIED
        popups_title CDATA #IMPLIED
        js CDATA #IMPLIED
        list (true | false) #IMPLIED
        edit (true | false) #IMPLIED
        add (true | false) #IMPLIED
        delete (true | false) #IMPLIED
    >
<!ELEMENT element (#PCDATA)>
<!ATTLIST element
    name CDATA #IMPLIED
    type (null | checkbox | dojo_combo_box | dojo_calendar | dojo_filtering_select |
    dojo_validation_textbox | dojo_checkbox | id_box | password | select | textarea | textfield | xml_list |
```

```

yes_no | dojo_yes_no | element | permissions_modules | dojo_textarea | dojo_validation_textarea |
ajax_box | list_files | hidden) #REQUIRED
    target CDATA #IMPLIED
    trim (true | false) #IMPLIED
    reg_exp CDATA #IMPLIED
    required (true | false) "true"
    visible (true | false) "true"
    edit (true | false) "false"
    add (true | false) "false"
    table CDATA #IMPLIED
    table_pk CDATA #IMPLIED
    table_method CDATA #IMPLIED
    field CDATA #IMPLIED
    image CDATA #IMPLIED
    link CDATA #IMPLIED
    list CDATA #IMPLIED
    default_value CDATA #IMPLIED
    default_text CDATA #IMPLIED
    constraints CDATA #IMPLIED
    search_format (equals, greaterorequals, lesserorequals, date_format)
    search_required CDATA #IMPLIED
    search_constraints CDATA #IMPLIED
    search_reg_exp CDATA #IMPLIED
    invalid_message CDATA #IMPLIED
    prompt_message CDATA #IMPLIED
    search_onchange CDATA #IMPLIED
    onchange CDATA #IMPLIED
    onblur CDATA #IMPLIED
    disabled CDATA #IMPLIED
    offset CDATA #IMPLIED
    checked CDATA #IMPLIED
    formatter CDATA #IMPLIED
    maxlength CDATA #IMPLIED
    join CDATA #IMPLIED
    width CDATA #IMPLIED
    clone (true | false) #IMPLIED
    functions CDATA #IMPLIED
    path CDATA #IMPLIED
    functions_permissions (list | add | edit | delete) #IMPLIED

>
<!ELEMENT tab (field*)>
<!ATTLIST tab
    name CDATA #IMPLIED
    target CDATA #IMPLIED
    id CDATA #IMPLIED
    add (true | false) #IMPLIED
    edit (true | false) #IMPLIED
    clone (true | false) #IMPLIED
>

```

Para algunos proyectos con requerimientos muy específicos puede ser necesario crear *components* o *elements* personalizados los cuales no tendría sentido añadir al archivo

DTD genérico, en ese caso el editor XML puede señalar que existen atributos incorrectos en el archivo XML, simplemente hay que ignorar esas advertencias.

## 6.3 Tareas comunes

### 6.3.1 Validación en XML con expresiones regulares

Por lo general sólo es necesario hacer las validaciones con expresiones regulares con el atributo *reg\_exp* de los elementos XML.

Este ejemplo contiene una expression regular para validar un e-mail.

```
<element name="E-Mail" target="email" reg_exp="[\w-\.]{3,}@[\\w-]{2,}\\.\\{2,4\\}[\w-]{2,4}" invalid_message="mail no valido" type="dojo_validation_textbox" visible="true" edit="true" add="true" />
```

De ser necesario se puede hacer validación adicional con PHP - Javascript.

### 6.3.2 Validación adicional con PHP - Javascript

De ser necesario también se puede usar javascript para una validación mas avanzada y una interacción mas personalizada escribiendo el JS adicional en la function `getEditValidation()` de los **modelos** que heredan `Zwei_Db_Table` retornando un string javascript que sera embebido dentro del js de la página HTML en el evento submit del formulario, puede discriminarse la edición del la inserción mediante la variable javascript '`global_opc`' que puede ser igual a '`edit`' o '`add`'.

Hay que tomar en cuenta que los ids de los elementos Dojo no dependen atributos XML por lo que hay que revisar el código HTML generado para poder usar los punteros `dijit.byId({id})` o `document.getElementById({id})`.

Ejemplo (Se debe sobreescribir método dentro de clase Modelo específica, el Modelo es el atributo target XML nodo principal).

```
public function getEditValidation(){
    return "
        var pfx = global_opc == 'add' ? '_add' : '';
        if (dijit.byId('edit0_'+pfx+'4').get('value') >= dijit.byId('edit0_'+pfx+'5').get('value')){
            alert('Hora Fin debe ser mayor a Hora Inicio');
            return false;
        }
    ";
}
```

## 6.3.3 Buscadores

### 6.3.3.1 Buscador de texto simple

Se busca por un solo campo a la vez mediante string con la operacion LIKE '%texto%', para esto se debe usar el atributo search en el nodo Component y separar cada campo por punto y coma.

Las etiquetas son obtenidas de los atributos name de los nodos Element cuyos atributos target sean iguales a los atributos search, separados por punto y coma.

Ejemplo (etiquetas clave en colores)

```
<?xml version="1.0"?>
<!DOCTYPE section PUBLIC "-//COMPONENTS//components.dtd">
<component name="Usuarios" search="user_name;email" type="table_dojo" target="acl_users" list="true" edit="true" add="true" delete="true">
    <element name="ID" target="id" type="id_box" visible="false" edit="false" add="false"/>
    <element name="Usuario" target="user_name" type="dojo_validation_textbox" visible="true" edit="false" add="true"/>
    <element name="Nombres" target="first_names" type="dojo_validation_textbox" visible="true" edit="true" add="true"/>
    <element name="Apellidos" target="last_names" type="dojo_validation_textbox" visible="true" edit="true" add="true"/>
    <element name="E-Mail" target="email" reg_exp="[\w-\.]{3,}@[\\w-]{2,}\\.\\{[\w-]{2,}\\}\\.\\{[\w-]{2,4}\\}" invalid_message="mail no valido" type="dojo_validation_textbox" visible="true" edit="true" add="true"/>
    <element name="Perfil" target="acl_roles_id" default_value="" type="dojo_filtering_select" table="acl_roles" field="role_name" visible="true" edit="true" add="true"/>
    <element name="Activo" target="approved" type="dojo_yes_no" visible="true" edit="true" add="true"/>
</component>
```

Usuario	Nombres	Apellidos	E-Mail	Perfil	Activo
zweicom	Soporte	Zweicom	rodrigo.riguele@zweicom.	Desarrollador	No
admin	Administrador	Zweicom	crodriguez@telefonicamov	Administrador	Si
sac	sac	sac		Consultas	Si

### 6.3.3.2 Buscador a partir de tablas

Acá se consideran los atributos

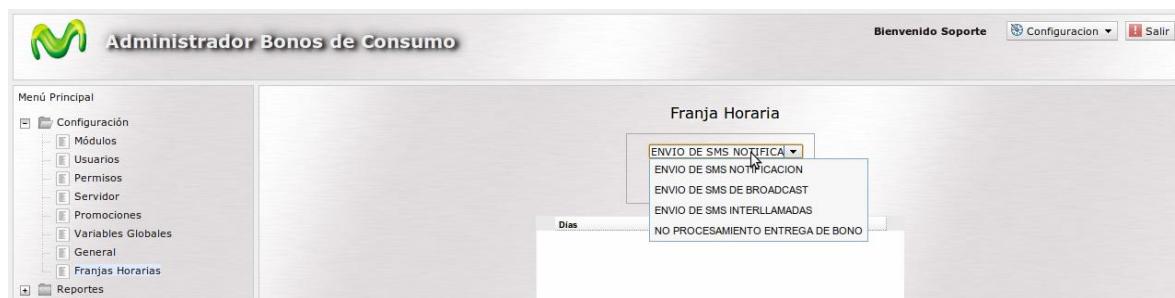
search\_table: Modelo del cual se poblará el combo.

search\_table\_field: Campo a desplegar en el combo

search\_table\_target: Key con la cual se hará JOIN entre modelo principal y la PK del modelo de search\_table.

search\_table\_pk: PK (opcional) si no se especifica se asume como "id".

```
<?xml version="1.0"?>
<!DOCTYPE section PUBLIC "-//COMPONENTS//components.dtd">
<component name="Franja Horaria" type="table_doj o" js="franjashorarias.js"
search_table="evento" edit_additional_validation="true" search_table_field="descripcion"
search_table_target="evento_id" target="franja_horaria" list="false" edit="true" add="true"
delete="true">
    <element name="ID" target="id" type="id_box" visible="false" edit="false" add="false"/>
    <element name="" target="evento_id" type="hidden" visible="false" edit="true"
add="true">
        <element name="D&iacute;as" target="dias_aplicacion" formatter="formatDias"
type="dias_multi_select" edit_custom_display="true" required="true" trim="true" visible="true"
edit="true" add="true"/>
        <element name="Hora Inicio" target="hora_inicio" type="dojo_validation_textbox"
reg_exp="(([0-1][0-9])|([2][0-3]):([0-5][0-9]):([0-5][0-9])" prompt_message="Hora en formato
HH:MM:SS, desde 00:00:00 hasta 23:59:59" invalid_message="Hora en formato HH:MM:SS, desde
00:00:00 hasta 23:59:59" required="true" trim="true" visible="true" edit="true" add="true"/>
        <element name="Hora Fin" target="hora_fin" type="dojo_validation_textbox" reg_exp="(([0-
1][0-9])|([2][0-3]):([0-5][0-9]):([0-5][0-9])" prompt_message="Hora en formato HH:MM:SS, desde
00:00:00 hasta 23:59:59" invalid_message="Hora en formato HH:MM:SS, desde 00:00:00 hasta
23:59:59" required="true" trim="true" visible="true" edit="true" add="true"/>
    </component>
```



## 7 Buscador Múltiple

Los atributos:

component.search\_type: debe ser “multiple”.

component.search: campos por los cuales realizar la búsqueda

component.search\_dojo\_type: es el atributo “dojoType” del input, separar por punto y coma por cada input, omitir con “null”

element.search\_format (opcional) puede ser (equals, greaterorequals, lesserorequals, date\_format), si no es especificado se usara el operador like en la búsqueda.

search\_required: (opcional) true

```
<?xml version="1.0"?>
<!DOCTYPE section PUBLIC "-//COMPONENTS//components.dtd">
<component name="Detalle Tráfico" excel="true" target="detalle_trafico"
type="table_dojo" list="false" edit="false" add="false" search="origen;fecha_trafico"
search_type="multiple" search_table="promociones_tipos" search_table_field="tipo_nombre"
search_table_target="tipo_trafico" search_dojo_type="null;dijit.form.DateTextBox">
    <element name="Número de Cliente" width="70" target="origen"
type="dojo_validation_textbox" visible="false" edit="false" add="false" search_required="true"
search_format="equals" trim="true" />
    <element name="Fecha Tráfico" width="70" target="fecha_trafico"
type="dojo_validation_textbox" visible="false" edit="false" add="false" search_required="true"
search_format="date_format" trim="true" search_constraints="{datePattern:'yyyy-MM-dd'}/>
    {...}
</component>
```



## 7.1 Selects

### 7.1.1 A partir de una lista

Deben configurarse los atributos:

**type**: “dojo\_filtering\_select” (recomendado) o “select” (no se acostumbra a usar)

**list**: lista de elementos separados por coma.

```
<element name="M&acute;x_Env&iacute;os" target="inter_llamadas_max"
  type="dojo_filtering_select" list="1,2,3,4" visible="false" edit="true" add="true"
  clone="true"/>
```

### 7.1.2 A partir de una tabla

Deben considerarse los atributos

**table**: modelo del cual se obtendrán los datos del select

**table\_pk**: (opcional) ingresar nombre de la primary key, si este no es el por defecto (“id”).

**table\_method**: (opcional) ingresar método de clase table si es que no se usará el método “select”, éste debe existir y retornar un objeto Zend\_Db\_Table\_Select válido.

**field**: nombre del campo a desplegarse en select, este campo también debe ser considerado en el método select del modelo principal (atributo target del nodo padre en XML), de lo contrario debe usarse además **table\_field**.

**table\_field**: (opcional) nombre del campo a desplegarse en el select, usar en de no existir este campo en modelo principal.

**default\_value**: (opcional) valor por defecto

```
<element name="Perfil" target="acl_roles_id" default_value=""
  type="dojo_filtering_select" table="acl_roles" field="role_name" visible="true"
  edit="true" add="true"/>
```

## 7.2 Lightbox

Se pueden incluir enlaces a lightbox tomando en cuenta los atributos

**popups**: enlace a abrir en nuevo lightbox, si son varios separarlos por punto y coma.

**popups\_title**: texto a desplegar en los botones respectivos, si son varios separarlos por punto y coma.

**popups\_iframe**: (opcional) si es true el lightbox cargara dentro de un iframe, esto es útil para prevenir problemas javascript o para llamadas XHR cross-domain.

```
<component name="Solicitudes" type="table_dojo"
    table_dojo_type="dojox.grid.EnhancedGrid" functions="assign_request" excel="true"
    popups="index/components?p=requests-details" popups_title="Detalles"
    popups_iframe="true" functions_permissions="edit" target="solicitud_th" list="false"
    search="msisdn,fijo" search_required="true" edit="false" add="false" delete="false">
    {elementos}
</component>
```

The screenshot shows a web application interface for managing requests. On the left, a sidebar menu includes 'Configuración', 'Reportes', and 'Consultas'. The main area is titled 'Solicitudes' and shows a table with two rows of data. A lightbox window titled 'Detalles' is open over the table, displaying the same data in a more detailed format with columns for ID, Móvil, Fijo, Fecha, Intento, and Resultado. The lightbox also includes a 'Select/Unselect All' checkbox and a 'Páginas: 1' link. To the right of the lightbox, there is a 'Resultado' column showing 'ERROR INTERNO' and a 'Texto SMS' column containing a promotional message about a new affiliation offer. At the bottom of the page, there are buttons for 'Exportar a Excel', 'Reprocesar Solicitud', and 'Detalles'.

## 7.3 Gráfico lineal

Se puede graficar un listado en un lightbox mediante un link a "index/components" en el atributo popups y luego crear el xml según el parámetro "p" del link al cual se hace referencia, en este caso ganadores-grafico.xml

```
<component name="Ganadores" target="ganadores" type="table_doj o" search_type="multiple"
search="stamp" search_table="promociones_tipos" search_display="between;table"
search_table_field="tipo_nombre" search_table_target="promociones.tipo"
search_doj o_type="dijit.form.DateTextBox" search_constraints="{datePattern:'yyyy-MM-dd'}"
popups="index/components?p=ganadores-grafico" popups_iframe="true"
popups_title="Ver Gr&amp;aacute;fico" list="false" edit="false" add="false" delete="false"
excel="true">
{elementos omitidos}
</component>
```

Los atributos:

**type:** debe ser igual a google chart,

**inherits\_data:** debe ser true (por ahora sólo hereda los datos del window.opener)

**options:** es opcional y corresponde al atributo options según la [API de Google](#),

**chart\_x\_type:** es opcional y puede especificar si el eje X es de tipo datetime (único valor válido), en este caso dibujará una línea continua (sin segmentar) en lugar de discreta.

**titles:** especifica los elementos del eje x

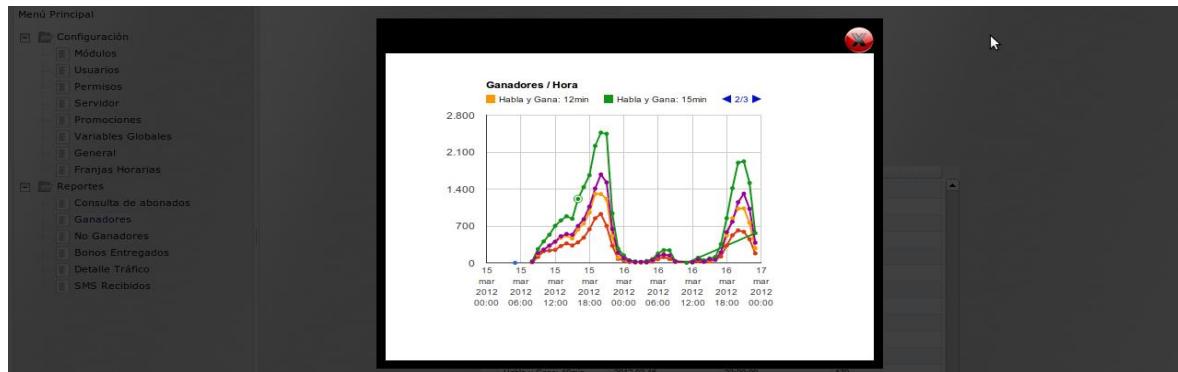
**items:** especifica los elementos del eje y

**cols:** especifica como se agrupan los elementos titles/items (x/y), se dibujará una linea diferente por cada cols, si es que tienen datos.

```

<component type="google_chart" options="{'width: 480, height: 290, title: 'Ganadores / Hora', strictFirstColumnType: true, fill: 0}" chart_x_type="datetime" inherits_data="true" cols="nombre" titles="fecha_hora" items="cantidad"/>

```



## 8 Ejecutar funciones auxiliares

Para ejecutar funciones auxiliares pasando como parámetros el modelo y el identificador se deben agregar los siguientes parámetros en el nodo principal del XML.

El parámetro functions\_permissions indica el nivel de permisos necesarios para ejecutar la función.

```
<?xml version="1.0"?>

<!DOCTYPE section PUBLIC "-//COMPONENTS/" "components.dtd">

<component name="Solicitudes" type="table_dojo"
    table_dojo_type="dojox.grid.EnhancedGrid" functions="assign_request"
    functions_permissions="edit" excel="true" popups="index/components?p=requests-
    details" popups_title="Detalles" target="solicitud_th" list="false" search="msisdn;fijo"
    search_required="true" edit="false" add="false" delete="false">

    {elements}

</component>
```

Móvil	Fijo	Fecha Recepción	Fecha Ejecución	Fecha Alta	Resultado	Texto SMS
234234234	42342343	2011-08-23 12:56:29	2011-08-21 01:00:00	2011-08-28	ERROR INTERNO	Estimado cliente, este numero fijo ya se encuentra afiliado a la promocion, su afiliacion adicional sera valida a partir del 1ero de septiembre. Gracias

El método debe ser escrito dentro de la clase Zwei\_Admin\_CustomFunctions, siempre recibe los parámetros: “id”, identificador de la fila seleccionada (si la hubiera) y “object”, nombre del componente.

Se debe asociar en el array \$\_names la descripción que aparecerá en el botón.

```
private $_names=array(
```

```
'clonarPromocion'=>'Clonar Promoción',
assignRequest=>'Reprocesar Solicitud'
);

public function clonarPromocion(){
echo "
<script type=\"text/javascript\">
window.parent.cargarTabsPanelCentral('promociones','clone');
</script>
";
}

}
```

## 9 Settings

Es recomendable que cada sistema web use por lo menos un modulo de configuración general, cuyo archivo XML tiene una estructura de este estilo.

```
<?xml version="1.0"?>  
<!DOCTYPE component PUBLIC "-//COMPONENTS/" "components.dtd">  
<component name="Configuraci&acute;n" type="settings_dojo" target="settings" />
```

Estructura de la tabla:

```
CREATE TABLE IF NOT EXISTS 'web_settings' (  
    'id' varchar(255) NOT NULL DEFAULT "",  
    'enum' varchar(255) NOT NULL,  
    'value' varchar(255) NOT NULL DEFAULT '0',  
    'type' varchar(255) NOT NULL DEFAULT "",  
    'description' text NOT NULL,  
    'ord' int(11) NOT NULL DEFAULT '0',  
    'group' varchar(255) NOT NULL,  
    'function' varchar(255) NOT NULL,  
    'approved' enum('0','1') NOT NULL,  
    PRIMARY KEY ('id')  
)
```

**id:** identificador alfanumerico, se debe ingresar a manualmente, debe tener un nombre descriptivo, sin usar caracteres especiales ni espacios, ejemplos: max\_upload\_size, query\_log

**enum:** Opciones posibles, separadas por coma.

**value:** Valor seleccionado, en caso de NO estar vacio **enum**, este valor debe corresponder a uno de sus valores.

**type:** El tipo de input que se usara para mantener la variable via admin, se recomienda usar “dojo\_filtering\_select” o “dojo\_validation\_textbox”

**description:** Tip de ayuda para explicar la funcionalidad de la variable.

**ord:** Los elementos debieran ordenarse me menor a mayor según el valor de este número (en pruebas).

**group:** String identificador de la pestaña. Por ejemplo todos los registros que tengan ‘group’='X' se desplegarán en la pestaña ‘X’

**function:** Función para ejecutar al presionar el botón asociado (En desarrollo),

**approved:** solo se mostrarán los registros con approved=1.

Observación: enum es palabra reservada en Java y da conflicto con los Setters y Getters por defecto en JPA por lo que deben ser modificados a mano. Para evitar esto a futuro sería útil cambiar enum por otra palabra en una actualización de este framework.

La clase modelo debe incluir el nombre de la tabla y la function loadGroups() para cargar diferentes tabs.

```
<?php

class SettingsModel extends Zwei_Db_Table

{
    protected $_name = "web_settings";

    public function loadGroups()

    {
        $query=$this->select()
            ->distinct()
            ->from(array('s' => $this->_name), 'group');

        $data=$this->fetchAll($query);

        return $data;
    }
}
```

