lazos

La unión de innovación y tecnología

Documentación para desarrolladores

"workflow"

Junio - 2009

http://www.lazos.cl

Tabla de Contenidos

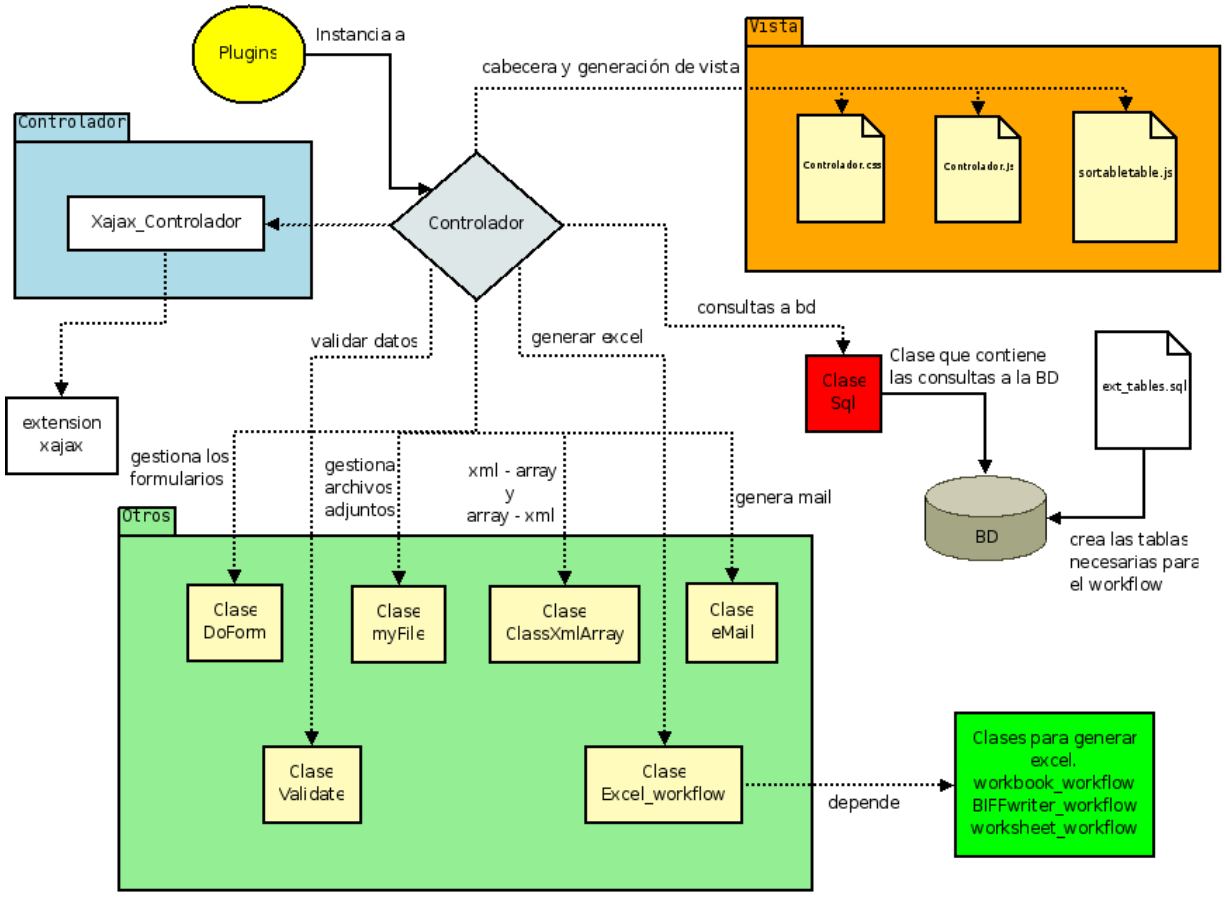
1. INTRODUCCION

Esta documentación esta orientada para servir de apoyo a la instalación y configuración del flujo de trabajo utilizado en el sistema de solicitudes. En este documento se describen los distintos componentes que interactuan para que el sistema de solicitudes quede operativo.

2. CONCEPTOS GENERALES SOBRE LA ARQUITECTURA DEL SISTEMA WORKFLOW

El sistema workflow esta compuesto por una serie de clases que son utilizadas para generar el flujo. En este capitulo se revisara la estructura que tiene el sistema workflow utilizado en lazos.

Como punto de partida se presenta a continuación un esquema integrando los componentes fundamentales que participan de la arquitectura de la extension workflow .



La extension motor de workflow la podemos dividir en cinco componentes fundamentales, cada uno de estos cumple con una labor especifica dentro de la extensión. A continuación describiremos cada uno de estos componentes:

Controlador

El controlador es el componente primario de la extension, es el componente encargado de gestionar el flujo de trabajo.

Vista

Este componente contiene las configuraciones necesarias para desplegar los formularios, este se compone de dos archivos fundamentales; una hoja de estilo css y un script validador javascript.

Consultas SQL

Se encarga de conectar la clase controlador con la base de datos utilizando consultas sql. La clase que guarda las consultas del workflow a la base de datos se llama *"Sql"* . Las tablas de la base de datos se crean utilizando un archivo reservado de typo3 que se encuentra en el directorio raiz de la extension y se llama ext_tables.sql

Plugins

Para realizar un flujo se necesita de varios plugins internos de la extensión. El workflow posee 6 plugins:

tx_workflow_pi1: A la fecha de este documento este plugins esta en desarrollo, es una portada.

tx_workflow_pi2: Plugin para ingresar procesos

tx_workflow_pi3: Plugin ver las operaciones enviadas

tx_workflow_pi4: Plugin ver las operaciones recibidas

tx_workflow_pi5: Plugin para ver todos los procesos

tx_workflow_pi6: Plugin utilizado para generar excel

Paquete de clases Otros: Componente integrado por clases necesarias para procesar algunas funcionalidades deseadas para el workflow. Podriamos decri que este es el paquete de utilizades.

Clase MyFile: esta encargada de gestionar los archivos que se adjuntan en una solicitud

Calse validate: Valida algunos tipos de datos especiales

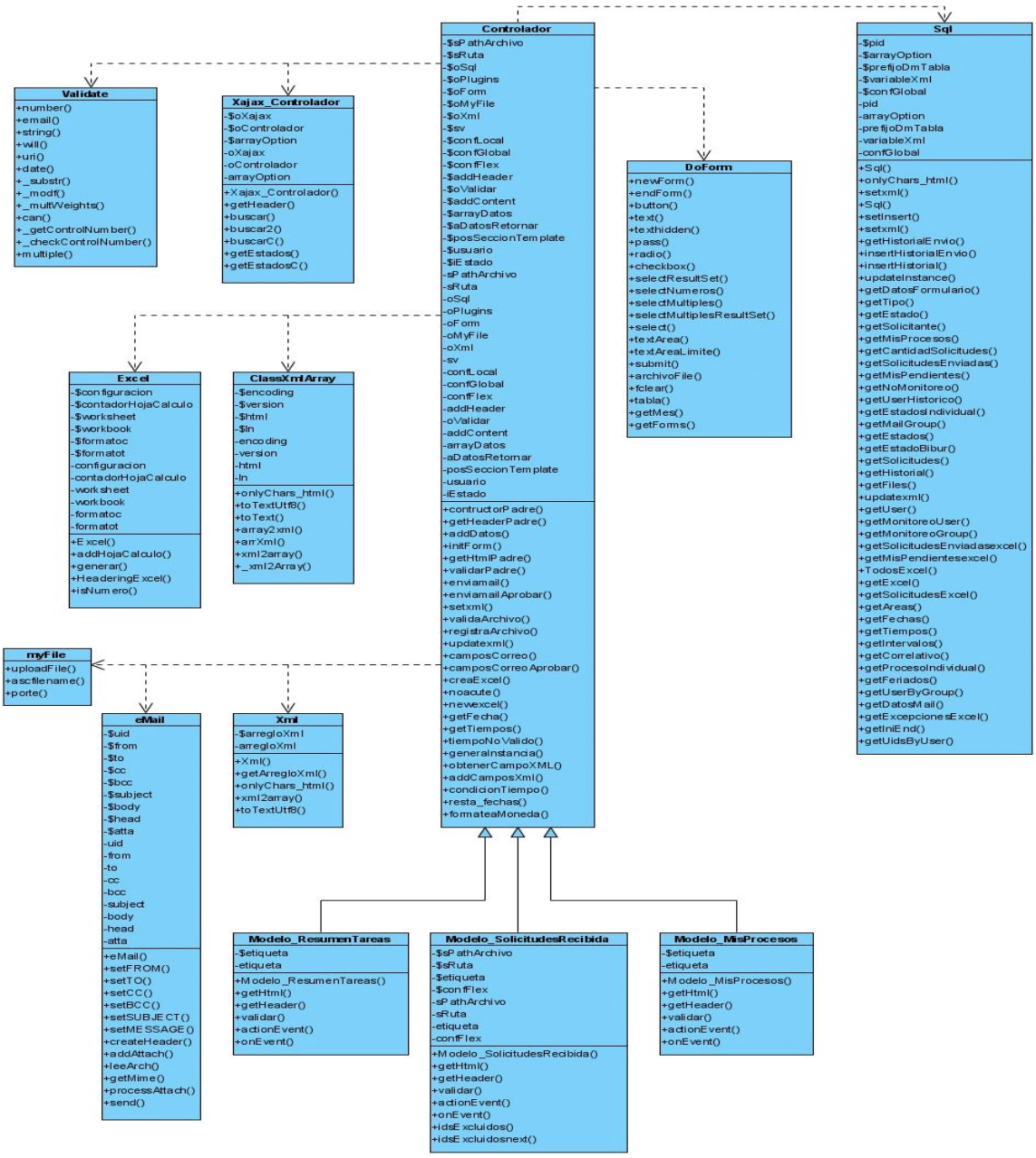
Clase Excel_workflow: Clase que produce un excel detallado de procesos seleccionados.

Clase DoForm: Clase encargada de gestionar los formularios, cuando se crean formularios que hacen uso de la extension workflow, estos heredan caracteristicas de esta clase.

Clase XmlArray: Se encarga de pasar de xml a array y viceversa.

2.1 Diagrama de clases Workflow

Para obtener una visión global del sistema workflow se muestra el diagrama de clases del workflow



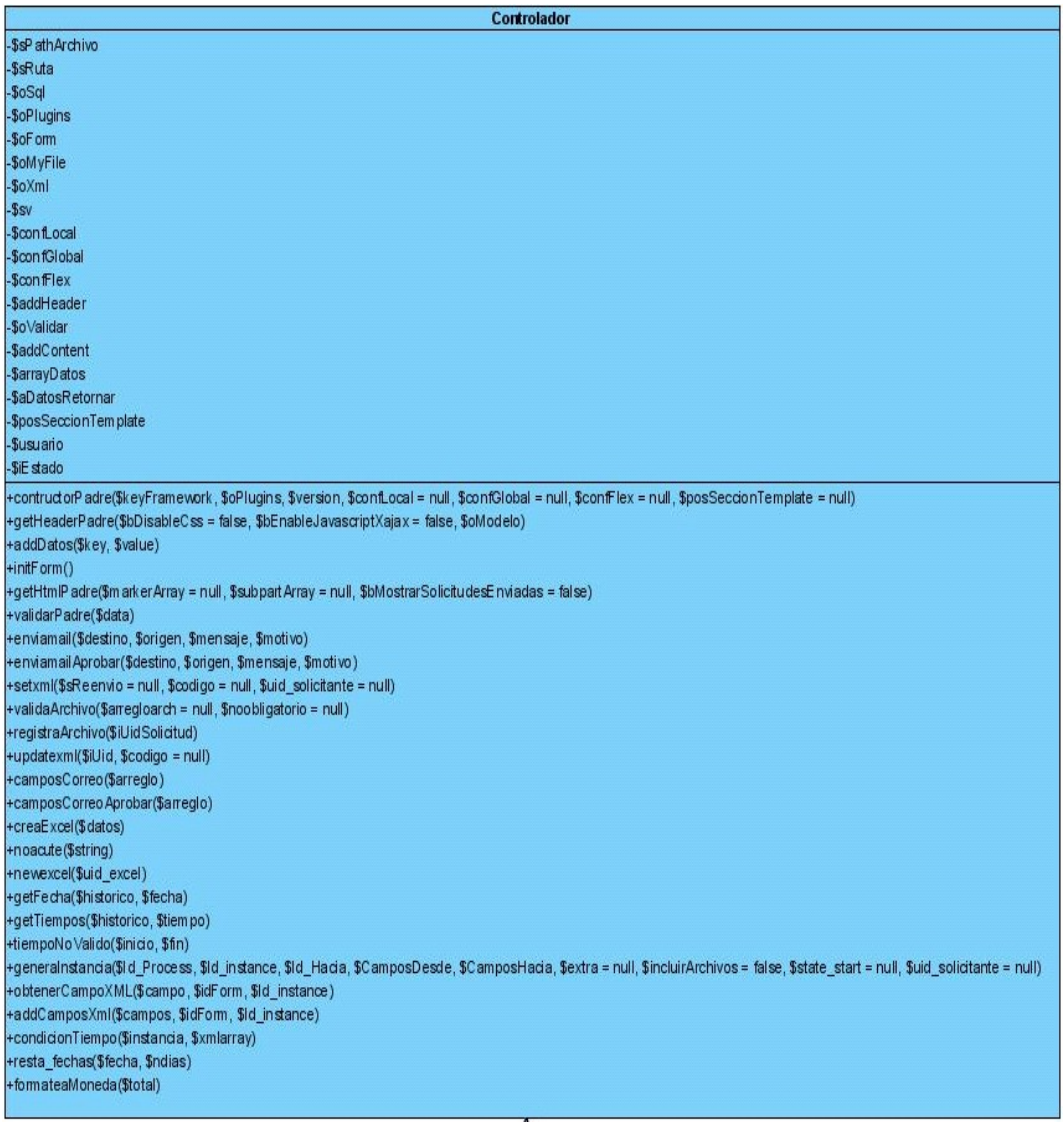
2.2 Descripción de las clases fundamentales del workflow

En esta sección las clases fundamentales que componen el workflow. Estas clases son: Controlador, Xajax_Controlador, DoForm y Sql.

2.2.1 Clase Controlador

Esta clase es la que procesa el flujo en general

Diagrama



Descripción de la clase Controlador

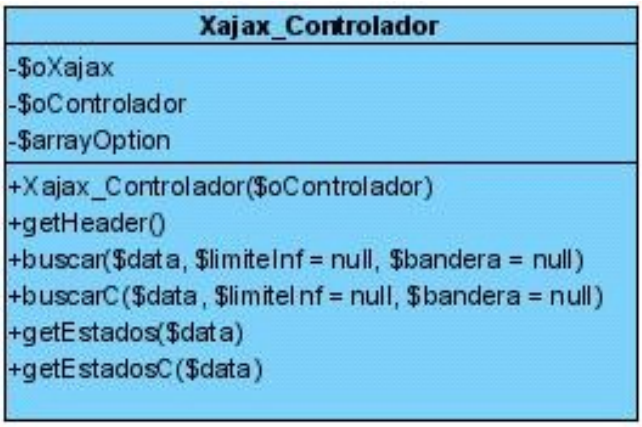
Variables Globales de la clase	
sPathArchivo	string Contiene la ruta relativa a la una carpeta de procesos temporales. Ej: fileadmin/processes/
sRuta	string Contiene la ruta relativa del directorio sv1
oSql	Contiene una instancia a la clase que almacena las consultas sql para hacer a la base de datos
oPlugins	Contiene un instancia al plugins para heredar las propiedades
oForm	Contiene un instancia del objeto que maneja los formularios
oMyFile	Contiene un instancia del objeto que maneja los archivos adjuntos.
oXml	Contiene un instancia al plugins para heredar las propiedades
sv	Prefijo del directorio del servicio
confLocal	Contiene la configuración Local
confGlobal	Contiene la configuración Global
confFlex	Contiene la configuración Flex
addHeader	Contiene elemento adicionales a la cabecera HEADER
oValidar	Contiene una instancia del objeto que valida los formularios
addContent	Contiene contenido extra
arrayDatos	Contiene la datos para mostrar en un arreglo
aDatosRetornar	Contiene la datos para retornar al formularios
posSeccionTemplate	Contiene la sección por defecto del template
usuario	Usuario Logeado o de la solicitud
iEstado	Estado Activo

Funciones		
Nombre de la función	Descripción	Referencias a otros archivos requiere_once(), href= y src=
constructorPadre	Función constructora de la clase controlador workflow	DAO/Sql.class.php Otros/Forms.class.php Otros/Validate.class.php Otros/myFile.class.php Otros/Xml.class.php
getHeaderPadre	Obtiene la cabecera,esta incluye los script Javascript, las hojas de estilo y el controlador Xajax return: Código Html con contenido de la cabecera	sv1/Controlador/Xajax_Controlador.class.php Vista/Controlador.css Vista/Controlador.js Vista/sortabletable.js
addDatos		
initForm	Método que carga los valores iniciales del formulario, no implementado en la versión revisada	

getHtmlPadre	Obtiene el html de la extensión	
validarPadre	Permite validar los campos obligatorios * Los campos que son enteros * los campos que son string * los campos que son rut * los campos que son teléfonos * Los campos que son mail * los campos que son fecha	
enviamail	Métodos enviamail utilizados por el proceso convenio	Otros/eMail.class.php
enviamailAprobar	Métodos enviamailAprobar utilizados por el proceso convenio aprobar	Otros/eMail.class.php
setxml	Guarda la instancia del proceso	Otros/class.xmlarray.php
validaArchivo	Función que valida los archivos adjuntos en una solicitud.	
registraArchivo	Función que registra los archivos en la base de datos	
updatexml	Actualiza la instancia del proceso	Otros/class.xmlarray.php
camposCorreo	Métodos utilizados por el proceso convenio	
camposCorreoAprobar	Métodos utilizados por el proceso convenio	
creaExcel	Metodo que crea Excel para el workflow	
noacute	Elimina los acute y los reemplaza por acentos, se usa para los excel que no permiten acute	
newexcel	Función que crea un nuevo excel del workflow	Otros/Excel.class.php
getFecha	Se obtienen las fechas configurada para cada solicitud	
getTiempos	Función que se obtienen los tiempos en días	
tiempoNoValido	Identifica que tiempo es no valido días no hábiles	
generaInstancia	Función que procesa una instancia de solicitud.	Otros/class.xmlarray.php
obtenerCampoXML	Obtiene un campo dentro del XML de la instancia	Otros/class.xmlarray.php
addCamposXml	Agrega un campo dentro del XML de la instancia	Otros/class.xmlarray.php
condicionTiempo	Retorna booleano que identifica si alguna instancia esta vencida o no	
resta_fechas	Resta dos fechas	
formateaMoneda	formatea los campos que son números	

2.2.2 Clase Xajax_Controlador

Diagrama



Descripción de la clases Xajax_Controlador

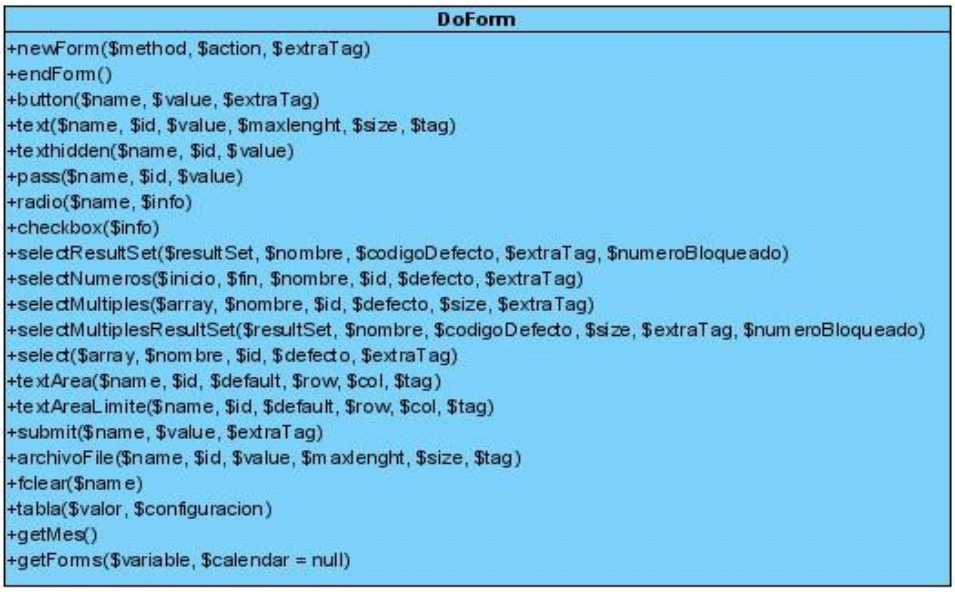
Variables Globales de la clase	
oXajax	Instanciación a la clase XAJAX
oControlador	Instancia a la clase del plugins
arrayOption	No se le encontró ninguna utilidad en la clase

Funciones		
Nombre de la función	Descripción	Referencias a otros archivos requiere_once(), href= y src=
Xajax_Controlador	constructor de la clase	class.tx_xajax.php , clase de la extension xajax
getHeader	Obtiene la cabecera javascript return: html Cabecera javascript de los métodos parseado por xajax	
buscar	Método que busca las solicitudes, pestaña 1	
buscarC	Método que busca solicitudes, pestaña 2	
getEstados	Método que rescata un estado de la base de datos	
getEstadosC	Método que rescata un estado de la base de datos	

2.2.3 Clase DoForm

Clase que permite Insertar Campos de Formularios

Diagrama



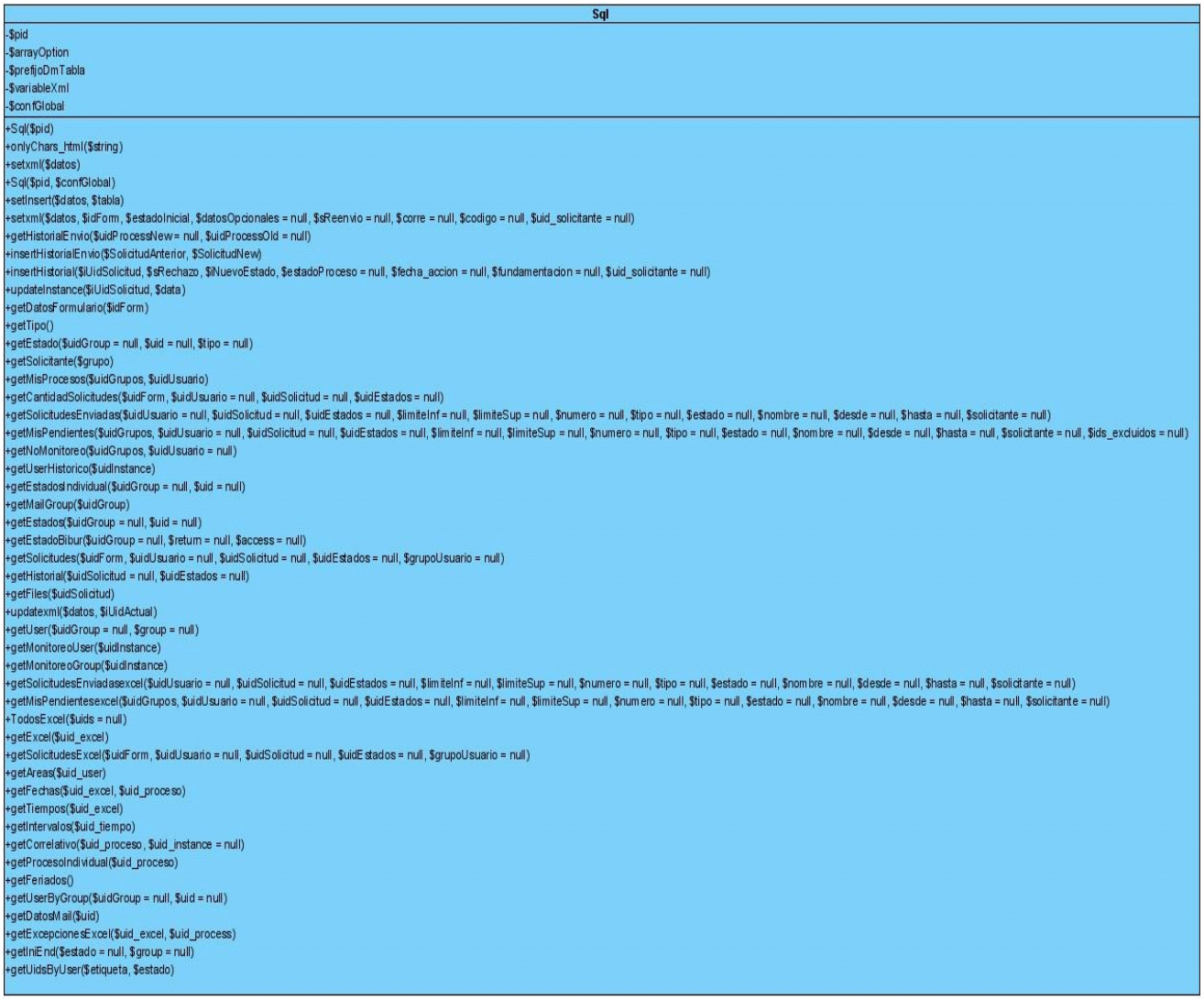
Funciones	
Nombre de la función	Descripción
newForm	Inicia un nuevo formulario Html return: string Código Html generado al iniciar un formulario
endForm	Finaliza un formulario return:string Código Html generado al finalizar un formulario
button	Crea un botón para suministrar el formulario return: string Código Html generado para visualizar el boton suministrar
text	Inserta un campo de texto return: string Código Html generado para visualizar el campo de texto en html
texthidden	Inserta un campo de texto Oculto return: string Código Html generado para visualizar el campo de texto Oculto en html
pass	Inserta un campo de Password return: string Código Html generado para visualizar el campo de Password en html
radio	Inserta un campo de radio return: string Código Html generado del objeto html radio

checkbox	Inserta un conjunto de checkbox string Código Html generado del objeto html Checkbox
selectResultSet	Genera un select con una consulta de sql debe tener esta consulta dos campos de valores: El primero es el valor y el segundo es el nombre del select. Igual permite crear select con una consulta con un solo campo ese campo sera el valor y la descripción del select. Return: string Código html del select originado
selectNumeros	Genera un combo-box
selectMultiples	Genera un combo-box de selección múltiple
selectMultiplesResultSet	Genera un combo-box de selección múltiple con otros parámetros
select	Genera un select
textArea	Genera un área de texto
textAreaLimite	Genera un área de texto con limite de área y caracteres, específicos
submit	Función enviar formulario
archivoFile	Genera un campo file
fclear	Resetea un campo
tabla	Genera una tabla
getMes	Genera arreglo con los meses
getForms	Genera un campo de formulario según parámetro que se le entregue

2.2.3 Clase Sql

Esta clase se encarga de procesar las consultas sql en la base de datos

Diagrama



Descripción

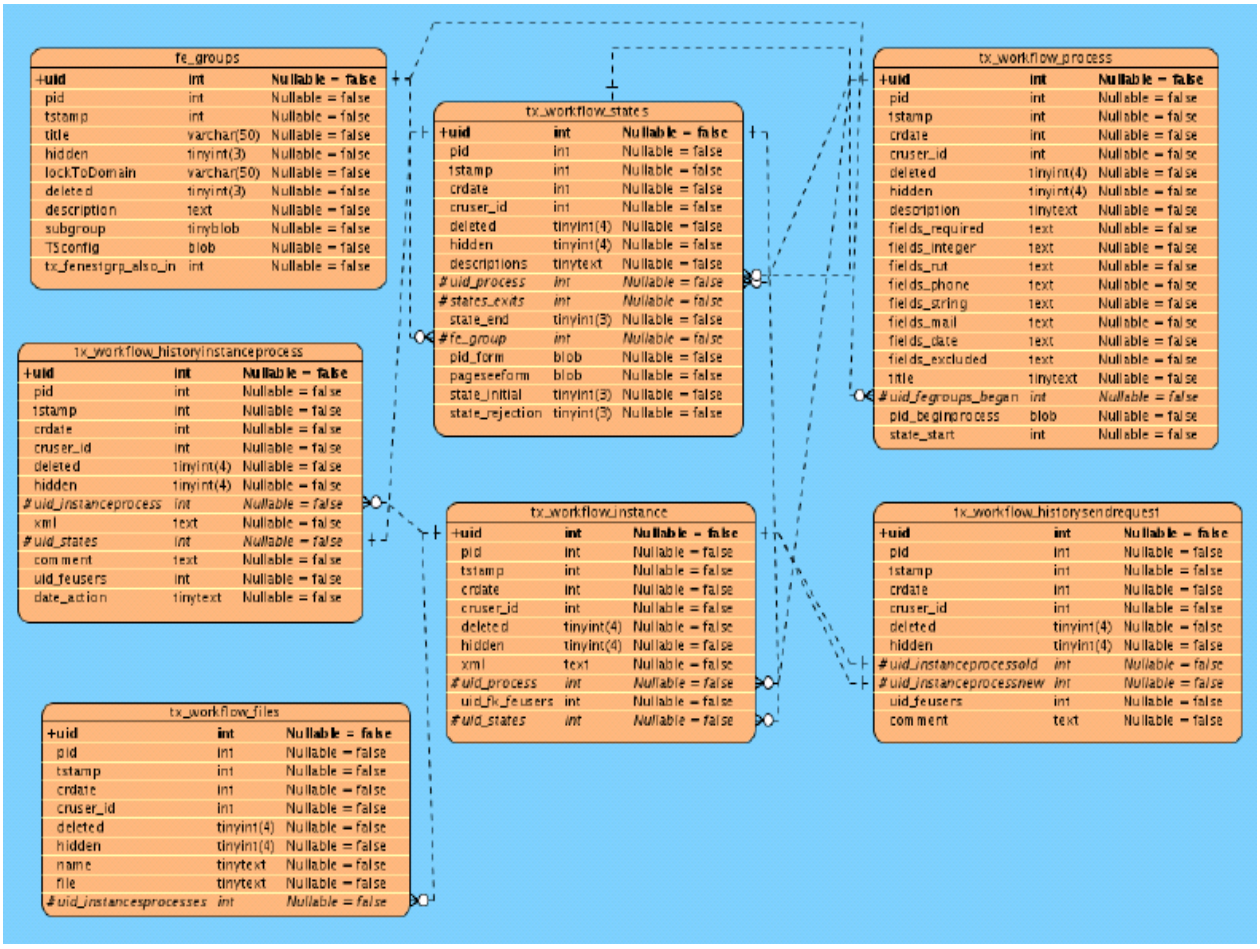
Variables Globales de la clase	
pid	Id de la instancia
arrayOption	Array con opciones de consulta
prefijoDmTabla	(deshuso)
variableXml	(deshuso)
confGlobal	Variables Globales

Funciones	
Nombre de la función	Descripción
Sql	Constructor de la clase
onlyChars_html	Evita imprimir cualquier tipo de carácter extraño
setInsert	Realiza un insert en la base de datos
setxml	Guarda la instancia del proceso
getHistorialEnvio	Obtiene el historial de envíos de la base de datos
insertHistorialEnvio	Inserta una entrada en el historial de envió
insertHistorial	Crea una instancia de proceso en el historial
updateInstance	Actualiza una instancia en la BD
getDatosFormulario	Obtiene los datos del formulario procesado
getTipo	Se configura cuales son los procesos que se deben mostrar y cuales no
getEstado	Recupera array con los estados de la base de datos
getSolicitante	Recupera array con id solicitantes
getMisProcesos	Recupera un array con los procesos
getCantidadSolicitudes	Obtiene la cantidad de solicitudes de la base de datos
getSolicitudesEnviadas	Obtiene de la base de datos la cantidad de solicitudes enviadas
GetMisPendientes	Recupera la cantidad de solicitudes pendientes en la base de datos
getNoMonitoreo	Devuelve todos los usuarios que no deben ver las solicitudes
getUserHistorico	Obtiene el historico de un usuario
getEstadosIndividual	Obtiene un solo estado
getMailGroup	Obtiene el el mail de un grupo especifico
getEstados	Devuelve varios estados
getEstadoBibur	Estados que tienen condición de paralelismo
getSolicitudes	Muestra la solicitudes asociado a un registro

getHistorial	Buscar el Historial
updatexml	Actualiza el xml de la instancia
getUser	Obtiene un usuario
getMonitoreoUser	Obtiene los usuario que monitorean
getMonitoreoGroup	Obtiene los grupos que monitorean
getSolicitudesEnviadasexcel	Obtiene las solicitudes enviadas para el excel
getMisPendientesexcel	Obtiene las solicitudes pendientes para generar un excel
TodosExcel	Obtiene todos los datos para generar un excel
getExcel	Obtiene los datos para un excel
getSolicitudesExcel	Obtiene datos para crear un excel de las solicitudes
getAreas	Obtiene la área de un usuario de acuerdo a su id
getFechas	Obtiene las fechas de la base de datos
getTiempos	Obtiene la configuración de tiempos para los excel
getIntervalos	obtiene la configuración de un tiempo configurado para el excel
getCorrelativo	Obtiene el ultimo correlativo para un determinado proceso
getProcesoIndividual	Obtiene una proceso individual
getFeriados	Devuelve un array con los días festivos
getUserByGroup	Obtiene los usuarios de un grupo
getDatosMail	Obtener los datos del email
getExcepcionesExcel	Obtiene una excepción de la base de datos asociada a un excel
getIniEnd	Restricciones de tiempos para solicitudes recibidas
getUidsByUser	Obtiene los uid por usuario

3. MODELO DE DATOS WORKFLOW

Se presenta el modelo de datos y su respectiva descripción para la extensión workflow.

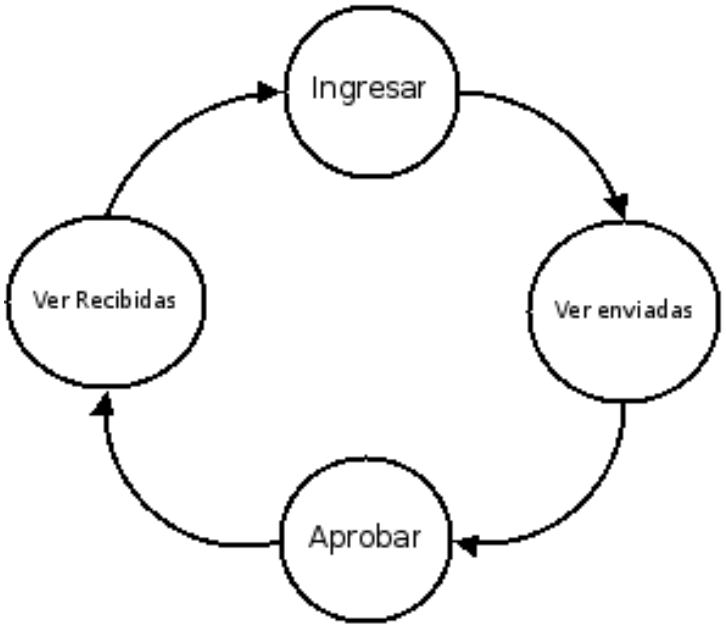


Descripción de las tablas:

Tabla	Descripción
tx_workflow_process	Tabla que contiene los procesos involucrados en el sistema
tx_workflow_states	La siguiente tabla contiene los registros de los estados de un determinado proceso.
tx_workflow_historyinstanceprocess	La siguiente tabla almacena el histórico de cambio de la solicitud
tx_workflow_instance	La siguiente tabla almacena las instancia de los procesos.
tx_workflow_historysendrequest	La siguiente tabla contiene el historial de reenvió de una solicitud. Un reenvió es cuando una solicitud llega a un estado terminal y se crea una nueva solicitud usando la misma información.
tx_workflow_files	La siguiente tabla almacena los archivos asociados a una instancia de procesos.

4. FORMULARIOS PARA EL WORKFLOW

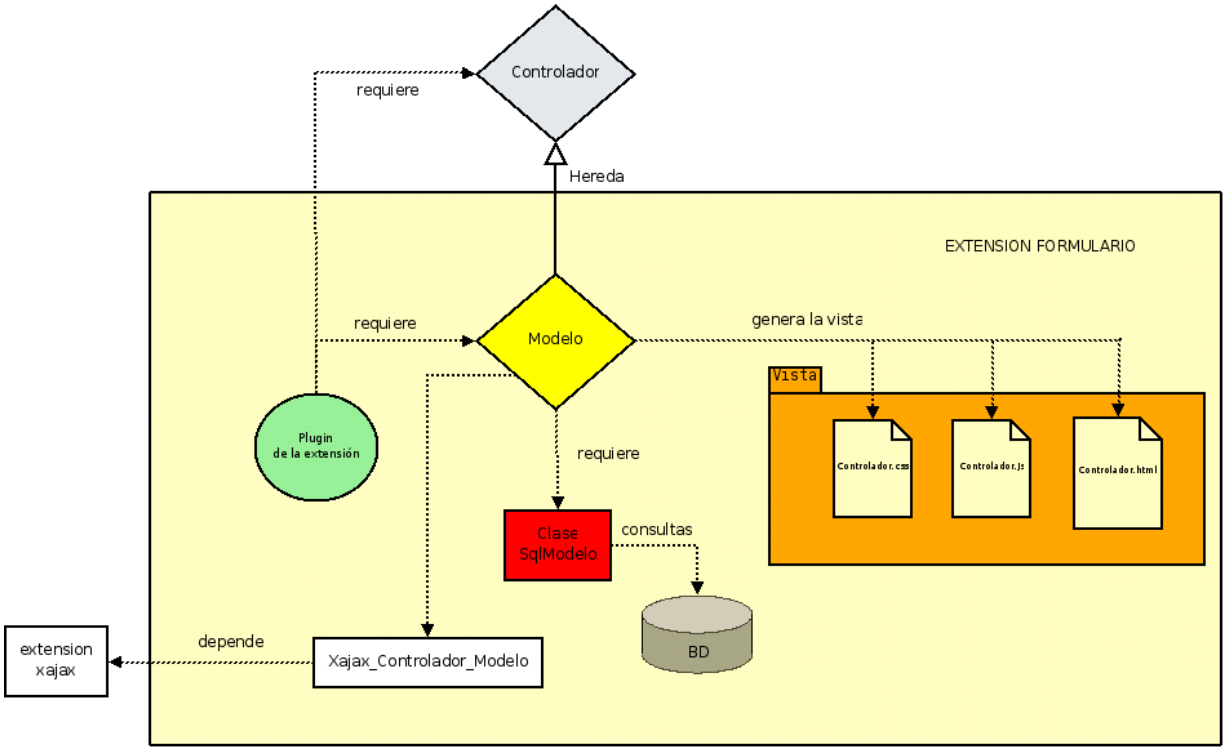
Si queremos generar un flujo especifico de algun sistema, debemos interactuar con el motor de workflow a traves de formularios dinamicos que realizen las debidas consultas por los procesos involucrados. Para generar un flujo basico, como el que se representa en la siguiente imagen, debemos tener como minimo cuatro formularios que cumplan con las siguientes operaciones.



Los formularios de mayor complejidad son los de tipo ingresar y aprobar, ya que requieren mayor cantidad de consultas a la base de datos y de formularios html mas extensos que requieren mayor dinámica. Sin embargo independiente de la complejidad de cada formulario, estos tiene una arquitectura similar, formada por los mismos componentes.

4.1 Arquitectura general que presenta un formulario para el workflow

A continuacion se presenta un diagrama con la arquietura comun que poseen los formularios



como se puede apreciar en el diagrama anterior la arquitectura es similar a la arquitectura general que posee el workflow. Los nuevos componentes se describen a continuación:

Modelo: Hereda los atributos y operacioens de la clase Controlador del workflow, este componente hace las veces del controlador de la extensión formulario.

SqlModelo: Contiene las consultas a la base de datos, propias del formulario.

Controlador.html: Archivo que contiene el template del formulario son las respectivas divisiones a traves de marcas.

4.2 Estructura de directorios de una extension formulario

Una extensión de typo3 cuenta con una extructura de directorio que segenera automaticamente cuando creamos unaextensión a traves de kickstarted. Sin embargo es necesario detenerse un momento e indentificar cuales son los archivos y directorios fundamentales.

4.2.1 Estructura fundamental

```
>> mi_formulario

    >> doc
    >> pi1
        class.tx_mi_formulario_pi1.php

    >> sv1
        >> DAO
            SqlModelo.class.php

        >> Modelo
            Modelo.class.php
            Xajax_Controlador_Modelo.class.php

        >> Vista
            Controlador.css
            Controlador.js
            Controlador.html

        class.tx_mi_formulario_sv1.php

    changelog
    ext_conf_template.txt
    ext_tables.sql
    ext_typoscript_setup.txt
```

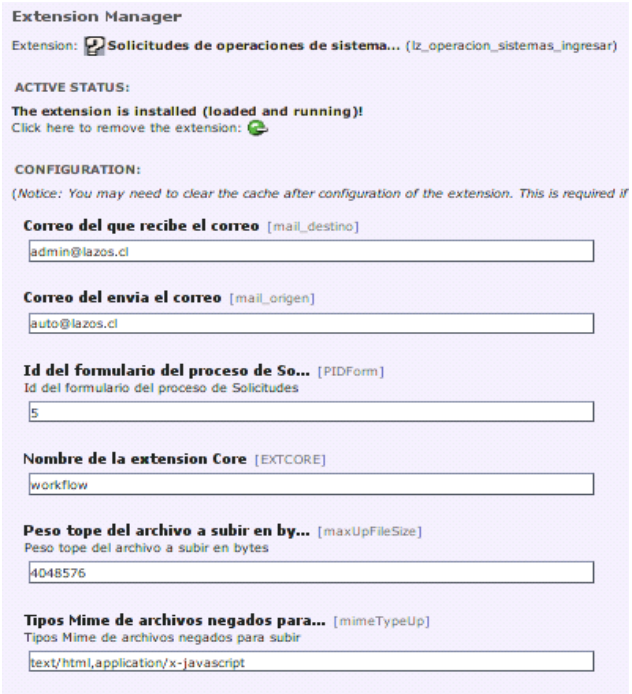
4.2.2 Descripción de la estructura de directorio

Archivos reservados de typo3

ext_conf_template.txt: Archivo utilizado por typo3 para crear un formulario de configuración en el extensión manager, con sus respectivas entradas, siendo esta la configuración por defecto del template de la extensión.

Ejemplo: El siguiente codigo genera formulario que se muestra a continuación.

```
# cat=basic//; type=string; label=Correo del que recibe el correo
mail_destino = admin@lazos.cl
# cat=basic//; type=string; label=Correo del envia el correo
mail_origen = auto@lazos.cl
# cat=basic//; type=string; label=Id del formulario del proceso de Solicitudes
PIDForm = 5
# cat=basic//; type=string; label=Nombre de la extension Core
EXTCORE= workflow
# cat=basic//; type=string; label=Peso tope del archivo a subir en bytes
maxUpFileSize = 4048576
# cat=basic//; type=string; label=Tipos Mime de archivos negados para subir
mimeTypeUp = text/html,application/x-javascript
```



ext_typoscript_setup.txt: Archivo reservado de typo3 en el cual se especifica ,a configuración por defecto de la extensión.

Ejemplo: ejemplo de un posible contenido de este tipo de archivo.

```
## LOGIN BOX

plugin.tx_lzoperacionsistemasingresar_pi1 {
   PIDForm = -1
   maxUpFileSize = -1
   COLOR1 = -1
   COLOR2 = -1
   MensajeArchivo = -1
   ArchMax = -1
   ID_MAIL_INGRESO = -1
   ID_GRUP_SOL = -1
   PID_OPERACION_PDF = -1
   PREFIJO_OPERACION = -1
   PREFIJO = -1
   UID_SOLICITANTE = -1
   UID_APRUEBA = -1
   UID_ESTADO_SOLICITANTE = -1
   UID_ESTADO_APRUEBA = -1
}
```

ext_tables.sql: cuando se instala por primera vez la extensión, typo3 lee este archivo para crear las tablas correspondientes dentro de la base de datos

changelog: archivo ubicado en el directorio raíz de la extensión , en el cual se lleva un registro de las distintas modificaciones que ha sufrido la extensión

Directorio Doc

En este directorio se guarda la documentación relativa a la extensión, es aconsejable utilizar phpDoc (http://www.phpdoc.org/) para generar una documentación de las clases, métodos y variables involucradas. En el anexo 1 “configuración del entorno de desarrollo” se explica la configuración de esta aplicación con eclipse.

Directorio Pi

En este directorio se guardan los archivos relativos a los plugins de la extensión.

class.tx_mi_formulario.class.php: archivo principal de un plugin, este se encarga de llamar a todas las clases necesarias para procesar un plugin.

Directorio SV

Este directorio representa un servicio en el contexto de typo3, en este caso el servicio que implementa un formulario es el workflow.

sub-directorio DAO: almacena los objetos de acceso a datos.

SqlModelo.class.php: contiene la clase de consultas a la base de datos

sub-directorio Modelo: almacena las clases modelo del formulario a traves de estas clases se estable la comunicación con el motor de workflow

Modelo.class.php: clase fundamental que gestiona las llamadas al workflow
Xajax_Controlador_Modelo.class.php: Clase necesaria para gestionar los formularios utilizando ajax

sub-directorio Vista: almacena los archivos necesarios para generar la vista

Controlador.css: Hojas de estilo css para la presentacion de los formularios.
Controlador.js: Contiene las validaciones necesarias para procesar el formulario.
Controlador.html: Contiene el template del formulario con las etiquetas correspondientes.

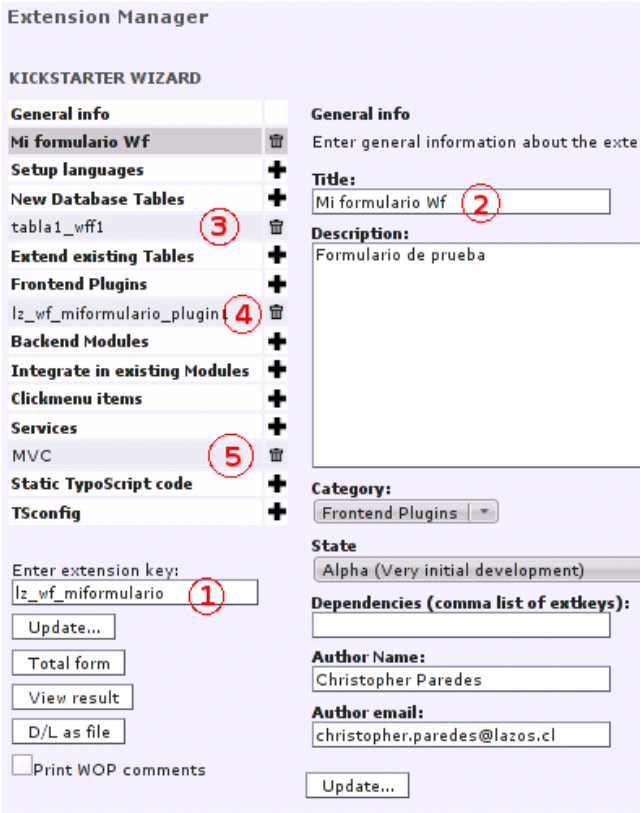
5. CREACION DE UN FORMULARIO

Basado en todo lo anterior, en esta sección procederemos a explicar como se crea un formulario que utiliza las prestaciones del servicio workflow. Para facilitar esto crearemos una secuenca de 3 pasos de a seguir

1er paso "Crear una extensión basica utilizando kickstarted"

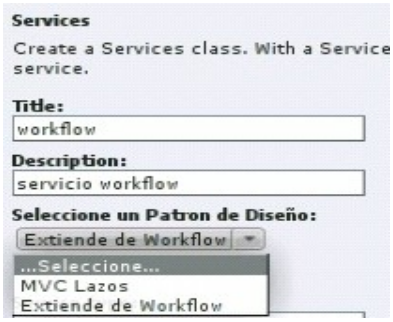
caracteristicas a crear:

1. extension_key
2. nombre de la extensión
3. crear tablas bd (solo si es necesario)
4. crear fronted plugins
5. crear un servicio llamado MVC
6. Generar una configuración para la extensión



Despues de realizar esto tendremos una estructura sobre la cual podemos comenzar a trabajar el codigo.

Existe un extensión kickstarted modificada y adaptada para la creación de formularios workflow. Esta extension nos permite seleccionar el patron adecuado para crear nuestros formularios.



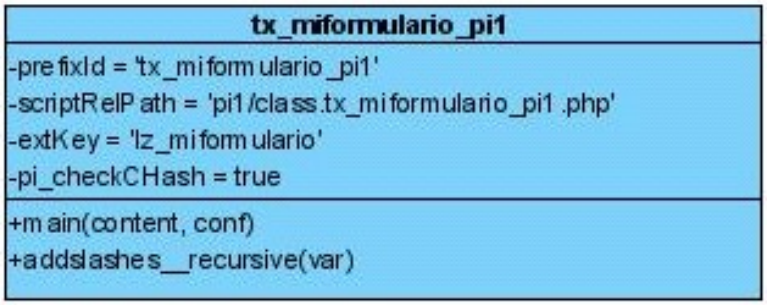
Si seleccionamos el patron de diseño “Extiende de Workflow”, esta extension nos creara la estructura de directorio necesaria para nuestro formulario.

2do paso **"Programar el plugin de la extensión formulario"**

Se deben programar los plugins que sean necesario, procesano parametros y llamadas a el servicio workflow, recordemos que los plugins se guardan en las carpetas **"pi_n"**

Plugin (class.tx_miformulario_pi1.php)

la estructura basica de la clase principal del plugin es como sigue, esta clase esta la primera que se cuarga cuando se llama a la extensión, por lo tanto es aquí donde debemos definir las llamadas al modelo.



La mayoria de los plugin formularios cuentan con esa estructura ahora definiremos las propiedades de esta clase.

Variables

Nombre	Descripción
PrefixId	Almacena el nombre de la clase
ScriptRelPath	Almacena la dirección del archivo que contiene la clase
extKey	Alamacena el nombre de la extensión key
pi_checkCHash	Variable de typo3

Metodos

Nombre	Descripción
main	Método principal de la clase
addslashes__recursive	este método elimina los contenidos que pueden causar problemas a la hora de hacer un insert a la base de datos.

Ejemplo de clase plugin:

```
class tx_lzoperacionsistemasver_pi1 extends tslib_pibase {

var $prefixId = 'tx_lzoperacionsistemasver_pi1';
var $scriptRelPath = 'pi1/class.tx_lzoperacionsistemasver_pi1.php';
var $extKey = 'lz_operacion_sistemas_ver';
var $pi_checkCHash = true;

function main($content, $conf) {
   $this->conf = $conf;
   $this->pi_setPiVarDefaults();
   $this->pi_loadLL();

   $_POST = $this->addslashes__recursive($_POST);
   $_GET = $this->addslashes__recursive($_GET);
   $_REQUEST = $this->addslashes__recursive($_REQUEST);
   $_SERVER = $this->addslashes__recursive($_SERVER);
   $_COOKIE = $this->addslashes__recursive($_COOKIE);

   global $TYPO3_CONF_VARS;
   $parameters = unserialize($TYPO3_CONF_VARS['EXT']['extConf'][$this->extKey]);

   if ($conf["PIDForm"] != -1) {
      $parameters["PIDForm"] = $conf["PIDForm"];
   }

   if ($conf["maxUpFileSize"] != -1) {
      $parameters["maxUpFileSize"] = $conf["maxUpFileSize"];
   }

   if ($conf["COLOR1"] != -1) {
      $parameters["COLOR1"] = $conf["COLOR1"];
   }

   if ($conf["COLOR2"] != -1) {
      $parameters["COLOR2"] = $conf["COLOR2"];
   }

   if ($conf["MensajeArchivo"] != -1) {
      $parameters["MensajeArchivo"] = $conf["MensajeArchivo"];
   }

require_once (t3lib_extMgm :: extPath($parameters["EXTCORE"])."sv1/Controlador/Controlador.class.php");
require_once (t3lib_extMgm :: extPath($this->extKey) . "sv1/Modelo/Modelo.class.php");

$oModelo = new Modelo($parameters["EXTCORE"], $this, $version = "v00", $confLocal = $conf, $confGlobal =
$parameters, $confFlex = null, "CONTENT_VER_INDIVIDUAL");

$oModelo->actionEvent($_POST);
$content .= $oModelo->getHtml();
$content = '<form name="frmsolicitud" method="POST" id="frmsolicitud"  enctype="multipart/form-data"> '
'<div id="div_menu" class="container" class="formu" >' .$content .'</div></form>';

$GLOBALS['TSFE']->additionalHeaderData[$this->prefixId] = $oModelo->getHeader(false, true);

return $this->pi_wrapInBaseClass($content);
}
```

```php
function addslashes__recursive($var) {
   if (!is_array($var))
   return addslashes($var);
   $new_var = array ();
   foreach ($var as $k => $v)
   $new_var[addslashes($k)] = $this->addslashes__recursive($v);
   return $new_var;
}
```

En resumen esta clase se encarga de validar los parámetros de configuración y llamar a las clases modulo y controlador para que hagan el resto. La función addslashes_recursive juega un rol muy importante para trabajar con la base de datos

Las lineas claves en esta clase son:

"llama a la configuración global de la extensión"

```php
global $TYPO3_CONF_VARS;
   $parameters = unserialize($TYPO3_CONF_VARS['EXT']['extConf'][$this->extKey]);
```

"Función para agregar slashes"

```php
function addslashes__recursive($var)
```

3er paso “programar el servicio de la extension formulario”

Los puntos fundamentales a conocer para programar el servicio asociado a un formulario

- template
- validaciones
- estilo
- modelo
- acceso a base de datos
- plugin

Template (controlador.html)

Para crear el template de la extension necesitamos conocer el sistema de marcas html utilizada por typo3 para la gestion de secciones de un template. Utilizando el sistema de marcas prdemos generar un template general y llamar mostrar solo partes del mismo de acuerdo a lo que se este procesando.

La estructura basica del template es la siguiente:

```
<title>Template extension formulario de prueba</title>
<h3>Template extension formulario de prueba</h3>

<!--###TEMPLATE### begin--->
<!--###CONTENT### -->

<!--###SECCION 1### →
  ###VAR 1###
  ###VAR 2###
  ###VAR N###
<!--###SECCION 1### -->

<!--###SECCION 2### -->
<!--###SECCION 2### -->

<!--###SECCION n### -->
<!--###SECCION n### -->

<!--###CONTENT## -->
<!--###TEMPLATE## end-->
```

cada una de las secciones son llamadas y las variables son completadas por la clase modelo del formulario. Utilizando un Array de datos que obtiene los estados necesarios.

Ejem: En el caso de un formulario del tipo Ver, este trae las solicitudes que el usuario logeado tiene permisos para ver, a traves de esta linea de codigo:

```
$this->arrayDatos = $this->oSql->getSolicitudes($_GET["uidver"],$this->usuario,$_GET["uidsolicitud"],$this->iEstado=null);
```

posteriormente, en base a una configuracion global o por typoscript, se muestra la información relacionada a un proceso:

```
$arrayDatos = $this->oSql->getDatosFormulario($this->confGlobal["PIDForm"]);
```

posteriormente completamos la información de las marcas con el array de marcas. $markerarray

por ejemplo:

En el caso de nuestro template generico, si queremos completar la marca ### VAR 1 ###, debemos realizar algo como lo que sigue:

```
$markerArray['###VAR 1###'] = "Detalle VAR 1".$arrayDatos[“var1"];
```

Esto dara resultado siempre y cuando var1 sea un parametro valido.

Validaciones (controlador.js)

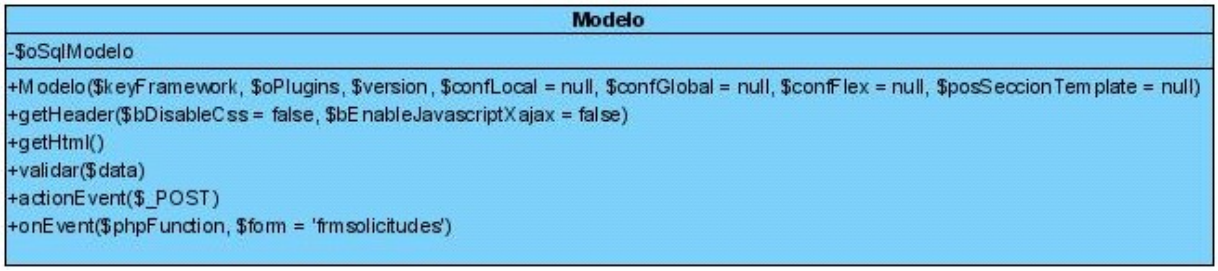
las validaciones están contenidas en el archivo de la vista controlador.js, estas son necesarias para procesar el formulario, por defecto se cargan las funciones que posee el controlador.js del workflow, si nuestra extension formulario requiere validaciones adicionales debemos incluirla en el controlador.js de la extensión.

Estilos (controlador.css)

Definimos la hoja de estilo para nuestro formulario, esta hoja de estilo esta en directa relacion con el archivo template controlador.html

Modelo (Modelo.class.php)

Esta clase hereda atributos y operaciones de la clase controlador del workflow. Una estructura basica para esta clase se muestra a continuación. Se recomienda comenzar a trabajar sobre esta estructura y posteriormente ir agregando atributos o metodos, siempre que el problema lo amerite.



Variables

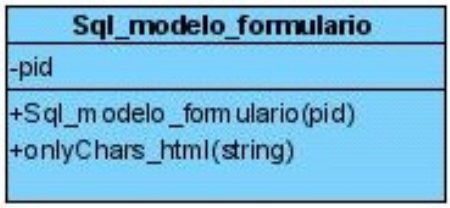
oSqlModelo: variable global de la clase utilizada para instanciar la clase consultas sql de la extensión.

Metodos

Nombre	Descripción
Modelo	Contructor de la clase
getHeader	Función que llama a la cabezera del formulario, esto se refiere a las hojas de estilo y el controlador.js
getHtml	Gestiona las marcas del template, llama a las secciones y rellena las marcas correspondientes.
validar	Método necesario para realizar las validaciones del formulario.
actionEvent	Método que controla los eventos.
onEvent	Evento

Acceso a datos (Sql.class.php)

Esta clase se utiliza para generar consultas sql a la base de datos. La estructura basica que debe contener esta clase es la siguiente:



Variables

pid: identificador de la clase

Metodos

Nombre	Descripción
Sql_modelo_formulario	Contructor de la clase
onlyChars_html	Función implementada para evitar imprimir cualquier tipo de carácter extraño.

Dependiendo la funcionalidad de la extensión. Podemos encontrar que tenemos que crear algunos métodos get o set adicionales, por ejemplo:

getFilesComentario: Consulta a la base de datos por la descripción de una solicitud y devuelve un array, con los datos.

ANEXO 1: CONFIGURACIÓN DEL ENTORNO DE DESARROLLO

Para desarrollar cualquier extensión relacionada a typo3 se recomienda contar con un entorno de desarrollo orientado al desarrollo en php y que preste funcionalidades deseadas como: autocompletación, identificación de los simbolos especiales(clases,funciones, variables) y documentación del desarrollo.

Estas carateristicas las ofrece el framework eclipse-php

Instalación

Descarge el ejecutable para su sistema operativo desde la siguiente url: http://www.eclipse.org/pdt/

Instalación y configuración de phpDocumentor

Se recomienda instalar esta aplicación para documentar los componentes de la extensión que estemos desarrollando, tambien es recomendable su integración con eclipse.

Instalación

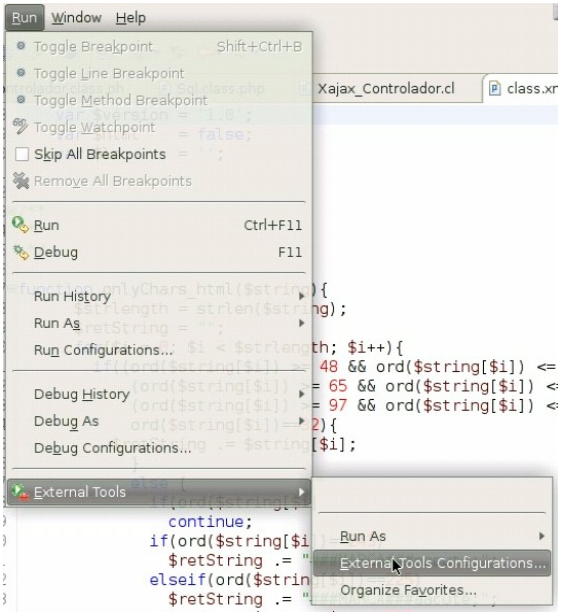
Descargamos el codigo fuente desde http://sourceforge.net/projects/phpdocu/files/

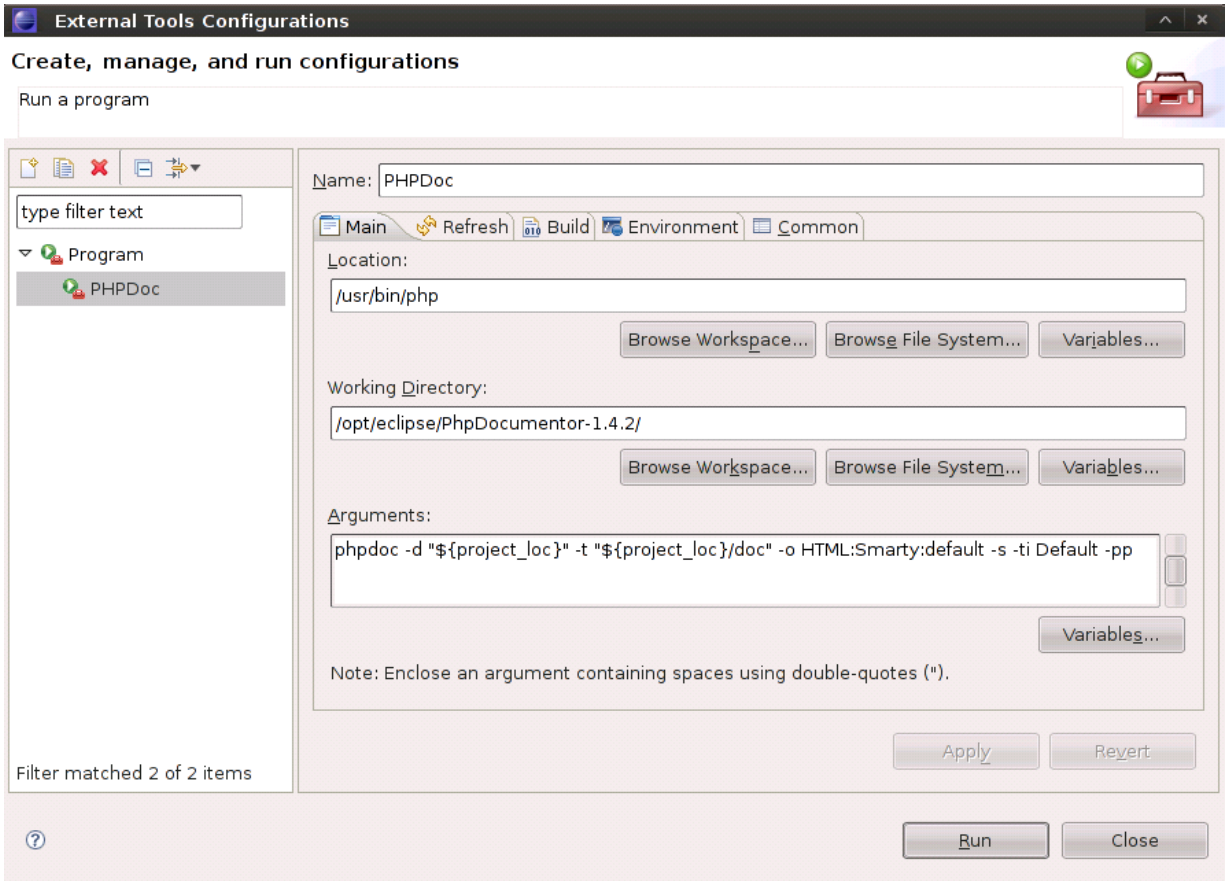
Procedemos a descomprimir el codigo en el directorio que deseemos, se recomienda descomprimir dentro del directorio de eclipse.

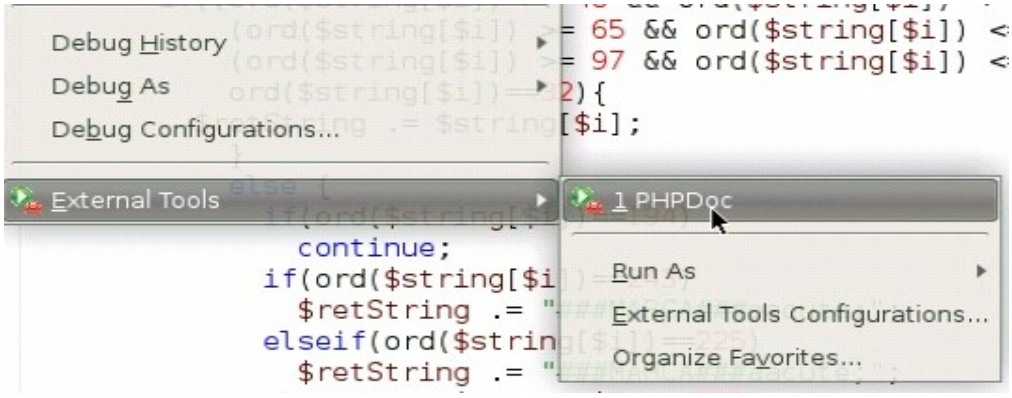
integración con eclipse

Para integrarlo con eclipse debemos agregarla como una herramienta externa(external tools)

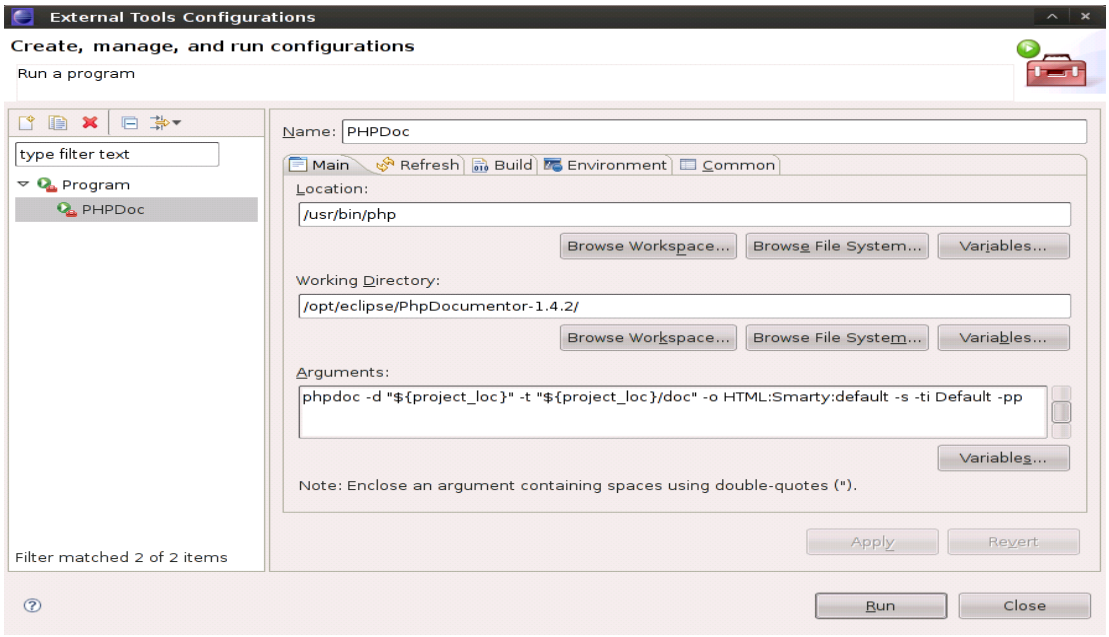
run -> external tools -> external tools configurations







Esto nos generara la documentación en el directorio doc de nuestra extensión.


External Tools Configurations
Create, manage, and run configurations
Run a program
type filter text
Program
PHPDoc
Filter matched 2 of 2 items
Name: PHPDoc
Main
Refresh
Build
Environment
Common
Location:
/usr/bin/php
Browse Workspace...
Browse File System...
Variables...
Working Directory:
/opt/eclipse/PhpDocumentor-1.4.2/
Browse Workspace...
Browse File System...
Variables...
Arguments:
phpdoc -d "${project_loc}" -t "${project_loc}/doc" -o HTML:Smarty:default -s -ti Default -pp
Variables...
Note: Enclose an argument containing spaces using double-quotes (").
Apply
Revert
Run
Close