Pasos de migración aplicaciones gvHidra a la versión 5

Versión 5.0.0

Tabla de contenidos

Parte I. Migración de aplicaciones gvHIDRA a la versión 5

Tabla de contenidos

Apéndice A. Pasos de migración para pasar a la versión 5.0.0

A.1. Codificación del proyecto.

A partir de la versión 5.0.0 la codificación de los proyectos gvHIDRA será UTF-8, por lo tanto, el primer paso a realizar es la recodificación de nuestra aplicación.

Revisión del código por si se han utilizado las siguientes funciones: utf8_encode y utf8_decode, ya no debe hacer falta su uso.

A.2. Cambios en la lógica de negocio.

- **Conexión a la BD**

  La conexión a la BD penaliza bastante el rendimiento, por eso se ha realizado un cambio para retardar el proceso de conexión hasta el momento necesario. Esto afectará a todo el código fuente que haga uso de la clase *IgepConexion*.

  El paso de migración consiste en buscar la cadena *new IgepConexion* y en todas aquellas líneas *incorporar* la llamada al método conectar:

```
$conexion = new IgepConexion ($dsn);
$conexion->conectar();
```

- **Migración de las listas de clase**

  En el caso de las listas definidas en clase se debe pasar la conexión como parámetro en el constructor. Por lo tanto el constructor quedará de la siguiente forma:

```
protected $_conDB;
protected $_lang;

public function __construct ($conn=null) {
  if (is_object ($conn)) {
    $this->_conDB = $conn;
  } else {
    try {
      $conf = ConfigFramework::getConfig();
      $this->_lang = $conf->getLanguage();
      $dsn = $conf->getDSN ('g_dsn');
```

```
        $this->_conDB = new IgepConexion ($dsn);
      } catch (Exception $e) {
        IgepDebug::setDebug (ERROR, 'Error en clase ' . __CLASS__ . ':\n'
              . PHP_EOL . $e->getMessage());
        throw new gvHidraException ('Error al conectar a la BD. '
              . 'Contacta con el administrador de la aplicación.');
      }
    }
  }
```

- **Revisión de paso de parámetros por $_GET**

  Se ha cambiado el sistema de comunicación a jQuery + JSON. La información que compone las matrices de datos se obtiene a través de una matriz de datos que se mantiene en JavaScript. Esto es transparente al programador excepto para **aquellas llamadas que utilizaban la url para pasar parámetros**, esto ya **NO** funcionará.

- **Nueva matriz de datos para los datos del panel FIL**

  Principalmente afecta a listados generados a partir de un panel de búsqueda. Hasta ahora, las acciones particulares que hacían uso de los datos de un panel de búsqueda, hacían referencia a la matriz de datos *external* para localizarlos. Para evitar confusiones en esta versión, estos datos van a estar almacenados en la matriz de "*visibles*". Esto significa que se deberán revisar estas acciones (típicamente en listados) y cambiar el método *setOperation*

  ```
  $objDatos->setOperation ('visibles');
  ```

- **Constructores de listas de clase y ventanas de selección**

  Cuando se utilizan clases para definir listas o ventanas de selección el constructor pasa a tener un parámetro relativo a la conexión. De modo que la definición del constructor quedará de la siguiente forma:

```
protected $_conDB;
protected $_lang;

public function __construct ($conn=null) {
  if (is_object ($conn)) {
    $this->_conDB = $conn;
  } else {
    try {
      $conf = ConfigFramework::getConfig();
      $this->_lang = $conf->getLanguage();
      $dsn = $conf->getDSN ('g_dsn');
      $this->_conDB = new IgepConexion ($dsn);
    } catch (Exception $e) {
      IgepDebug::setDebug (ERROR, 'Error en clase ' . __CLASS__ . ':\n'
            . PHP_EOL . $e->getMessage());
      throw new gvHidraException ('Error al conectar a la BD. '
            . 'Contacta con el administrador de la aplicación.');
    }
  }
}
```

A.3. Migración de los ficheros XML de configuración.

Se extrae la DTD de los ficheros xml de configuración, quedando así más limpios y sin necesidad de tener que ir migrando las DTD's conforme evoluciona el framework. Por lo tanto, hay que eliminar la DTD incluida en los ficheros XML, y añadirla de forma relativa tal y como se indica a continuación:

- **gvHidraConfig.inc.xml** Eliminar la DTD y añadir lo siguiente:

```
<!DOCTYPE gvHidraConfig SYSTEM "igep/dtd/configAPP.dtd">
```

- **menuModulos.xml** Eliminar la DTD y añadir lo siguiente:

```
<!DOCTYPE menu SYSTEM "../igep/dtd/menu.dtd">
```

- **menuHerramientas.inc.xml** Eliminar la DTD y añadir lo siguiente:

```
<!DOCTYPE menu SYSTEM "../igep/dtd/menu.dtd">
```

- **menuAdministracion.inc.xml** Eliminar la DTD y añadir lo siguiente:

```
<!DOCTYPE menu SYSTEM "../igep/dtd/menu.dtd">
```

A.4. Migración del fichero AppMainWindow.php

Solamente afecta a la función **emptyLogTable()**

```
public function emptyLogTable ($dias=60) {
  //Recogemos dsn de conexion
  $conf = ConfigFramework::getConfig();
  $g_dsnLog = $conf->getDSN ('gvh_dsn_log');
  $usuario = IgepSession::dameUsuario();
  try {
    IgepDebug::purgeDBLog ($dias, $usuario, $g_dsnLog);
  } catch (Exception $e) {
    error_log (__FILE__ . __METHOD__ . ' Error al vaciar tabla de LOG');
  }
  return 0;
}//emptyLogTable
```

A.5. Migración del fichero mappings.php

Si el botón **accion="cancelar"** no tiene definido el parámetro **action** en la plantilla *tpl*, hay que asegurarse que la regla *mapping* está definida la acción **cancelarEdicion**.

```
$this->_AddMapping ('ClaseM___cancelarEdicion','ClaseM');
$this->_AddForward ('ClaseM___cancelarEdicion',
                    'gvHidraSuccess',
                    'index.php?view=views/p_ClaseM.php&panel=listar');
```

A.6. Migración de las plantillas (TPL).

Las plantillas son las que más migración requieren. Por una parte, se ha evolucionado la versión de Smarty que incorpora el framework y, por otro lado, se han creado nuevos parámetros en los plugins, cambiado otros, y eliminados algunos plugins que ya son obsoletos.

A.6.1. Plugins obsoletos.

Se deben eliminar de las plantillas *TPL* los siguientes plugins:

- {CWContenedorPestanya} ... {/CWContenedorPestanya}
- {CWPestanya}
- {CWSelector} ... {/CWSelector}

A.6.2. Maestro/Detalle.

En las plantillas *TPL* que corresponden a un maestro-detalle hay que eliminar los tags que separaban el maestro del detalle.

```
</td></tr><tr><td>
```

A.6.3. Migración del nombre de los plugins.

Debido a la migración de Smarty en su versión 3, los plugins pasan a ser escritos en **minúsculas**, por lo tanto hay que reemplazar en todas las plantillas *tpl* el nombre de los plugins por el mismo pero en minúsculas.

Para el reemplazo se aconseja seguir el orden que se indica a continuación:

- `CWPantallaEntrada --> **cwpantallaentrada**`
- `CWVentana --> **cwventana**`
- `CWBarraSupPanel --> **cwbarrasuppanel**`
- `CWBarraInfPanel --> **cwbarrainfpanel**`
- `CWBarra --> **cwbarra**`
- `CWMenuLayer --> **cwmenulayer**`
- `CWPanel --> **cwpanel**`
- `CWBotonTooltip --> **cwbotontooltip**`
- `CWBoton --> **cwboton**`
- `CWMarcoPanel --> **cwmarcopanel**`
- `CWContendor --> **cwcontendor**`
- `CWFichaEdicion --> **cwfichaedicion**`
- `CWFicha --> **cwficha**`
- `CWTabla --> **cwtabla**`
- `CWFila --> **cwfila**`
- `CWPaginador --> **cwpaginador**`
- `CWSolapa --> **cwsolapa**`
- `CWInfoContendor --> **cwinfocontendor**`
- `CWInformation --> **cwinformation**`
- `CWDetalles --> **cwdetalles**`
- `CWGraph --> **cwgraph**`
- `CWImagen --> **cwimagen**`
- `CWLabel --> **cwlabel**`
- `CWMaps --> **cwmaps**`
- `CWAreaTexto --> **cwareatexto**`

- `CWCampoTexto --> **cwcampotexto**`
- `CWCheckBox --> **cwcheckbox**`
- `CWLista --> **cwlista**`
- `CWRichAreaTexto --> **cwrichareatexto**`
- `CWUpload --> **cwupload**`

A.6.4. Notas importantes sobre los parámetros de los plugins.

En algunos plugins se añaden parámetros que son altamente necesarios para mejorar la funcionalidad del framework.

- **{cwbotontooltip}**

  Obligatorio el parámetro **id**.

- **{cwboton}**

  Obligatorio el parámetro **id**.

  En el caso de que el parámetro

  ```
  accion="particular"
  ```

  y se quiere que el botón sea visible sin necesidad de activar el panel para edición, se debe añadir el parámetro

  ```
  visible="true"
  ```

- **{cwsolapa}**

  Obligatorio el parámetro **id**.

- **{cwlabel}**

  Obligatorio el parámetro **tipo**. Con él se indicará el tipo de etiqueta que se quiere mostrar:

  - **icon**: icono
  - **iconLabel**: icono + etiqueta
  - **label**: etiqueta
  - **link**: enlace
  - **iconLink**: icono + enlace

- **{cwmaps}**

  Nuevo parámetro **poligono** que contendrá el nombre del campo que contenga la información del polígono a dibujar en el mapa. Este parámetro solamente se debe incluir en el caso de que se vaya a dibujar un polígono en el mapa.

A.6.5. Parámetros de plugins que se deben eliminar.

Se deben eliminar algunos parámetros de ciertos plugins porque no es necesario indicarlos en las plantillas *tpl* porque se pueden obtener internamente.

- **{cwbotontooltip}**

  Eliminar el parámetro: **imagen** (Si aún se tiene este parámetro porque viene de la rama 4.2.x)

- **{cwboton}**

  Eliminar el parámetro: **imagen** (Si aún se tiene este parámetro porque viene de la rama 4.2.x)

- **{cwfichaedicion}**

  Eliminar el parámetro: **id**

- **{cwmarcopanel}**

  conPestanyas parámetro opcional con el que se puede controlar si se visualizan o no la botonera ['nueva búsqueda','listado','edición']. Por defecto su valor es "true", se visualiza.

- **{cwpanel} ... {/cwpanel}**

  Eliminar los parámetros: **method** y **tipoComprobacion**

- **{cwtabla}**

  Eliminar el parámetro: **id**

A.6.6. Parámetros de plugins que se deben renombrar.

Se renombran algunos parámetros para unificar criterios.

- **{cwbarra}**

  Los siguientes parámetros:

  ```
  usuario=$smty_usuario codigo=$smty_codigo customtitle=$smty_customTitle
  ```

  Se unifican en uno solo que será un array, por lo que deberá ser sustituido por el siguiente parámetro:

  ```
  info=$smty_info
  ```

- **{cwbarrasuppanel}**

  El parámetro *titulo* renombrarlo por *title*.

- **{cwbotontooltip}**

  - El parámetro *funcion* renombrarlo por *accion*.
  - El parámetro *titulo* renombrarlo por *title*.

- **{cwboton}**

  El parámetro *texto* renombrarlo por *label*.

- **{cwareatexto}, {cwcampotexto}, {cwcheckbox}, {cwlista}, {cwrichareatexto}, {cwupload}, {cwimagen}, {cw-label}, {cwslider}, {cwinformation}**

  - El parámetro *textoAsociado* renombrarlo por *label*.
  - El parámetro *mostrarTextoAsociado* renombrarlo por *showLabel*.

- **{cwmenulayer}**

  El parámetro *cadenaMenu="$smty_cadenaMenu"* renombrarlo por *arrayMenu=$smty_arrayMenu*

- **{cwsolapa}**

El parámetro *titulo* renombrarlo por *title*

A.6.7. Paso de la variable $smty_iteracionActual al uso de la librería IgepJS.js

En esta nueva versión del framework la comunicación cliente-servidor ha cambiado, ahora no se tienen todas las páginas cargadas en el cliente por lo que cada cambio de página es una actualización del contenido de los campo sin recarga de la página completa. Este cambio implica que la variable `$smty_iteracionActual` que se utilizaba para adaptar la interfaz según ciertos criterios, ya no es posible utilizarla porque solamente se tendrá cargado un registro cada vez (una iteración).

Para no perder esta funcionalidad se ha creado una librería javaScript, **`gvh_IgepJS.js`**, donde se encuentra una serie de funciones relacionadas con la interfaz y que permitirán realizar la adapatación de la misma. La actualización de la interfaz ahora se adaptará cuando cambie el contenido del campo. Es decir si un campo, del que depende una modificación de interfaz, altera su valor lanzará una llamada al método correspondiente de `gvh_IgepJS` y se actualizará la interfaz.

A.6.7.1. Funciones incluidas en la librería.

- ```
  /**
   * getStatePanel: Obtenemos el estado del panel ('R', 'W', 'I')
   *
   * @access public
   * @return string  Estado del panel ('R', 'W', 'I').
   */
  gvh.IgepJS.prototype.getStatePanel = function()
  ```
- ```
  /**
   * _panelVisible: Devuelve si el panel está visible o no
   *
   * @access private
   * @param panel: id del panel a consultar
   */
  gvh.IgepJS.prototype._panelVisible = function()
  ```
- ```
  /**
   * getRowComponent: Obtiene el row del componente indicado
   *
   * @access public
   * @param componente: Nombre del campo
   * @return integer
   */
  gvh.IgepJS.prototype.getRowComponent = function( componente )
  ```
- ```
  /**
   * getIdComponent: Obtiene el id del componente indicado
   *
   *  @access public
   * @param {string} componente - Nombre del campo
   * @param {boolean} external - [true / false]
   * @param {integer} [row]  - Fila del componente
   */
  gvh.IgepJS.prototype.getIdComponent = function( componente, external, row )
  ```
- ```
  /**
   * getValue: Obtiene el valor del campo
   *
  ```
  ```

```
 * @access public
 * @param componente: Nombre del campo
 * @param external: [true / false]
 */
gvh.IgepJS.prototype.getValue = function( componente, external )
```

- ```
  /**
   * setValue: Asigna valor a un campo
   *
   * @access public
   * @param componente: Nombre del campo
   * @let  valor: Valor a asignar al componente
   * @param external: [true / false]
   */
  gvh.IgepJS.prototype.setValue = function( componente, valor, external )
  ```

- ```
  /**
   * getClass: Obtiene el class del campo indicado
   *
   * @access public
   * @param componente: Nombre del campo
   * @param external: [true / false]
   */
  gvh.IgepJS.prototype.getClass = function( componente, external )
  ```

- ```
  /**
   * addClassComponent: Modifica el class de un componente
   *
   * @access public
   * @param componente: Nombre del campo
   * @param css: estilo css a añadir
   * @param external: [true / false]
   */
  gvh.IgepJS.prototype.addClassComponent = function( componente, css,
   external )
  ```

- ```
  /**
   * removeClassComponent: Modifica el class de un componente
   *
   * @access public
   * @param componente: Nombre del campo
   * @param css: estilo css a añadir
   * @param external: [true / false]
   */
  gvh.IgepJS.prototype.removeClassComponent = function( componente, css,
   external )
  ```

- ```
  /**
   * getVisible: Obtener si un campo es visible o no
   *
   * @access public
   * @param componente: Nombre del campo
   * @param external: [true / false]
   */
  gvh.IgepJS.prototype.getVisible = function( componente, external )
  ```

- ```
  /**
   * setVisible: Mostrar/ocultar un campo
   *
  ```

```
   * @access public
   * @param componente: Nombre del campo
   * @param visible:  [true / false]
   * @param external: [true / false]
   */
  gvh.IgepJS.prototype.setVisible = function( componente, visible, external )
```

- ```
  /**
   * getEnable: Obtener si un campo está habilitado o no
   *
   * @access public
   * @param componente: Nombre del campo
   * @param external: [true / false]
   */
  gvh.IgepJS.prototype.getEnable = function( componente, external )
  ```

- ```
  /**
   * setEnable: Habilitar/deshabilitar un campo
   *
   * @access public
   * @param componente: Nombre del campo
   * @param enable:  [true / false / nuevo]
   * @param external: [true / false]
   */
  gvh.IgepJS.prototype.setEnable = function( componente, enable, external )
  ```

A.6.7.2. Uso de la librería

Como se ha indicado, la actualización de interfaz se realiza con JavaScript por lo tanto la programación se debe incluir en la tpl como <script></script> con el atributo **defer** porque interesa que se ejecute lo último, cuando ya esté todo cargado, al igual que nos aseguramos que jQuery también está cargado. En el caso de que la actualización de la interfaz corresponda a un panel detalle colocar el bloque <script>...</script> antes de cerrar el {/if} del detalle.

```
{if count($smty_datosFichaM ) gt 0 }
  ...
  <script type="text/javascript" defer>
    ...
  </script>
{/if}
```

Primero se tiene que invocar al método subscribeRewireUI para tener suscrita la lógica de interfaz correspondiente a la clase manejadora del panel, y crear el objeto de la clase IgepJS de la clase manejadora para poder invocar las funciones de la librería.

Ejemplo de uso y correspondencia con el uso de la variable $smty_iteracionActual:

Uso de $smty_iteracionActual:

```
{if $smty_datosFicha.$smty_iteracionActual.edi_componente eq "S"}
   {CWCampoTexto nombre="ediDescripcion" size="40" editable="true" value=
$defaultData_ClaseManejadora.ediDescripcion}
{else}
   {CWCampoTexto nombre="ediDescripcion" size="40" editable="false" value=
$defaultData_ClaseManejadora.ediDescripcion}
{/if}
{if $smty_datosFicha.$smty_iteracionActual.edi_perfil eq "ADMINISTRACION"}
   {CWCampoTexto nombre="ediTelefono" size="10"
 editable="true" visible="true" value=
$defaultData_ClaseManejadora.ediTelefono}
```

```
{else}
   {CWCampoTexto nombre="ediTelefono" size="10"
 editable="true" visible="false" value=
$defaultData_ClaseManejadora.ediTelefono}
{/if}
```

Uso de la librería gvh_IgepJS:

```
<script type="text/javascript" defer>
$(document).ready( function() {
   // Invocación al método subscribeRewireUI para subscribir la lógica de
 interfaz (función) a la clase manejadora del panel.
   gvh.subscribeRewireUI("ClaseManejadora",function()
   {
   // Crear un objeto de la clase IgepJS pasándole la clase manejadora y el
 tipo de panel ('lis','lisDetalle','edi','ediDetalle').
      objIgepJS_ClaseManejadora = new gvh.IgepJS('ClaseManejadora','edi');

   // Uso de los métodos de gvh_IgepJS
      if (objIgepJS_ClaseManejadora.getValue('edi_componente') == 'S')
      {
        objIgepJS.setEnable('ediDescripcion', true, false);
      }
      else {
        objIgepJS.setEnable('ediDescripcion', false, false);
      }

      if (objIgepJS.getValue('edi_perfil') == 'ADMINISTRACION')
      {
        objIgepJS.setVisible('ediTelefono', true, false);
      }
      else {
        objIgepJS.setVisible('ediTelefono', false, false);
      }
   }
});
</script>
```

A.7. Migración de los views.

En el caso de pantallas tipo **Maestro-Ndetalles** hay un cambio en los **views**. Hay que añadir el tipo de panel en el array de definición de los paneles detalle.

```
$detalles = array()
array(
  "panelActivo" => "Detalle 1" ,
  "titDetalle"  => "Título detalle 1" ,
  "panel"       => "lis"
) ,
array(
  "panelActivo" => "Detalle 2" ,
  "titDetalle"  => "Título detalle 2" ,
  "panel"       => "edi"
)
```