hipergate

Guía de Creación de Plantillas XSL

Versión 7.0
Abril 2013

1 Introducción

Diseñar es resolver problemas que no pueden ser formulados hasta que éstos han sido resueltos. La forma de la respuesta es parte de la pregunta.

Plantillas para la producción de contenidos

El módulo de producción de contenidos permite crear newsletters y sitios web. Ambos tipos de publicación requieren la existencia de plantillas de presentación que definen la disposición del contenido, el estilo cromático y la tipografía en el documento final.

hipergate incorpora conjuntos de plantillas predefinidos para la creación de newsletters y microsites y permite la creación de nuevas plantillas con nuevos diseños.

Para poder comenzar con el diseño de una nueva plantilla debemos conocer como se organiza la información y la metainformación en el módulo de producción de contenidos de hipergate. En este manual veremos:

1. Estructura de archivos implicados
2. Estructura de directorios
3. Estructuras de datos en SGBDR
4. Proceso de creación de plantillas
5. Proceso de validación de plantillas

Esta guía se organiza de tal modo que una vez comprendidas las estructuras de datos podamos construir nuestros propios conjuntos de plantillas y publicar contenidos basados en los nuevos diseños.

Para una buena comprensión de éste manual es coveniente haber adquirido previamente conocimientos profundos en lenguajes de marcado (HTML, CSS, XML y XSLT) y SQL.

2 Estructura de archivos implicados

El módulo de producción de contenidos de hipergate utiliza distintos tipos de archivo para la creación de un documento, ya sea newsletter o sitio web:

1. **Archivo XML de datos**: archivo que se alimenta de la información que el usuario inserta desde la aplicación (textos e imágenes) y desea publicar.
2. **Archivo XML de metadatos**: archivo con la definición de estructuras de información permitidas para una plantilla de documentos.
3. **Archivo XML de precarga de datos**: archivos con datos ejemplo que aparecen al insertar una nueva página desde la aplicación. Existe uno por cada tipo de página de información (contenedor).
4. **Archivo XSL de presentación de página**: archivo con instrucciones para la presentación de información de una página.
5. **Archivo CSS de estilos**: archivo con estilos predefinidos utilizables desde el archivo XSL de presentación de página.

Estructura de archivos XML de datos

Los archivos XML de datos almacenan los contenidos de información editable por el usuario desde el asistente del módulo de producción de contenidos de hipergate. Dado que se trata de un archivo XML la información se organiza de forma jerárquica.

Los archivos XML de datos se crean y actualizan desde la aplicación. Por ello no es necesario crearlos de forma manual, pero sí es conveniente conocer su estructura con objeto de comprobar el correcto funcionamiento de producción de contenidos con nuevas plantillas incorporadas en hipergate.

A continuación se enumeran todos los items que componen un archivo XML de datos:

1. **Cabeceras y juego de caracteres:** se trata de un conjunto de líneas conformes al estándar XML que indican el tipo de archivo XML y el juego de caracteres utilizado en la información contenida en el archivo.

2. **Conjuntos de páginas de información:** un archivo XML de datos puede contener un conjunto de páginas (***pageset***) con un identificador único asociado (***guid***). Mediante el tag ***pageset*** se limita la zona en cuyo interior aparecerá la información insertada por el usuario y la relación con el tipo de documento que se quiere producir.
3. **Identificador de documento:** identificador único del tipo de documento, indica el tipo de plantilla que se aplicará para presentar la información de la newsletter o sitio web.
4. **Tipografía:** indica el tipo de letra seleccionado para mostrar los párrafos de texto del documento.
5. **Gama cromática:** indica el estilo visual (gama cromática y conjunto de estilos CSS predefinidos) para el documento.
6. **Página de información:** delimita mediante el tag ***page*** la información relativa a una página de contenidos. En el caso de newsletter, el archivo de datos XML sólo contiene una zona page, en el caso de sitio web puede haber múltiples zonas page, una por página de información. El tag ***page*** va acompañado del atributo ***guid*** que contiene el identificador único de página.
7. **Título descriptivo de la página de información:** cadena de texto, editable en el caso de sitio web, que difiere del resto de páginas.
8. **Identificador del contenedor base:** cada página de información está asociada a un contenedor base. Un contenedor base es una estructura de metainformación que define el tipo y cardinalidad de la información que puede aparecer en una página de información. La información se divide en bloques que pueden contener textos e imágenes. El contenedor base define qué tipos de bloque se pueden insertar en la página cuando estamos en modo edición y su definición se almacena en el archivo XML de metadatos.
9. **Conjunto de bloques de información:** en esta zona figuran los bloques de información que contiene la página. Cada bloque contiene textos e imágenes y lleva un identificador asociado.
10. **Bloque de información:** el tag ***block*** delimita la información contenida en un bloque de información. Va acompañado del atributo ***id*** que proporciona un identificador único al bloque en la página a la que pertenece.
11. **Identificador de metabloque base:** indica el tipo de metabloque que define la estructura de la información contenida en el bloque actual. Los metabloques permitidos se encuentran en el archivo XML de metadatos y describen los textos e imágenes permitidos en un bloque.
12. **Etiqueta descriptiva de bloque:** cadena de texto que diferencia los distintos bloques entre si.
13. **Conjunto de párrafos de texto:** el tag ***paragraphs*** encierra todos los párrafos contenidos en el bloque actual.

14. **Identificador de párrafo:** contiene el identificador del párrafo tal como se indica en la información existente en el archivo XML de metadatos.
15. **Datos del párrafo:** texto y, opcionalmente, url asociada de salto.
16. **Conjunto de imágenes:** el tag ***images*** encierra todos las imágenes contenidas en el bloque actual.
17. **Identificador de imágen:** contiene el identificador de la imágen tal como se indica en la información existente en el archivo XML de metadatos.
18. **Datos de la imagen:** ruta al archivo, opcionalmente url de salto, texto alternativo y dimensiones.

Estructura de archivos XML de metadatos

Los archivos XML de metadatos almacenan información acerca de los conjuntos de datos pemitidos en un documento (newsletter o sitio web), tipografías y gamas cromáticas válidas.

En el archivo XML se especifican los contenedores de datos y para cada contenedor los tipos de bloque de información insertables.

A continuación se enumeran todos los items que componen un archivo XML de metadatos:

1. **Cabeceras y juego de caracteres:** se trata de un conjunto de líneas conformes al estándar XML que indican el tipo de archivo XML y el juego de caracteres utilizado en la información contenida en el archivo.
2. **Identificador del tipo de documento:** identificador único de tipo de newsletter o sitio web.
3. **Nombre descriptivo de plantilla:** cadena de texto con un nombre descriptivo de la plantilla de documento.
4. **Tipografías permitidas:** el tag ***fonts*** contiene una lista de las tipografías permitidas que son seleccionables al editar la información del documento desde la aplicación.
5. **Gama cromática permitida:** el tag ***colors*** contiene una lista de los colores permitidos (colores de fondo, imágenes predefinidas de plantilla y estilos CSS) que son seleccionables al editar la información del documento desde la aplicación.

6. **Contenedores (páginas) de información:** el tag ***containers*** puede contener 1 o varios contenedores, según se trate de una newsletter o un sitio web.
7. **Identificador de contenedor:** cada contenedor representa la definición estructural de una página de información. Todo contenedor tiene asociado un identificador único que aparece en el atributo ***guid***.
8. **Nombre de contenedor:** cadena de texto descriptiva de la página que se generará cuando esté basada en el contenedor.
9. **Archivo de presentación asociado:** nombre del archivo XSL con la presentación asociada al contenedor.
10. **Conjunto de metabloques permitidos:** un metabloque describe la estructura que puede tener un bloque de información. Un bloque de informacón puede contener textos e imágenes. En el metabloque base se indica el tipo de contenidos y la cardinalidad. El tag ***metablocks*** contiene los distintos metabloques permitidos al utilizar este contenedor.
11. **Estructura de metabloque:** una estructura de definición de metabloque comienza con un tag ***metablock***. Tiene asociado un identificador en el atributo ***id***.
12. **Nombre del metabloque:** cadena de texto descriptiva para el metabloque.
13. **Objetos de información permitidos en el metabloque:** se indica que tipos de objeto pueden aparecer en el metabloque (textos o imágenes), con un nombre para cada objeto y su cardinalidad. La sintaxis es una lista separada por comas con el tipo, dos puntos, el nombre del objeto de información y la cardinalidad entre corchetes. El tipo de objeto puede ser ***p***, si se trata de un párrafo de texto, o ***i*** , si se trata de una imagen. La cardinalidad se especifica con 1..n.
14. **Cardinalidad máxima del metabloque:** indica el máximo número de veces que puede aparecer un bloque de este tipo en el contenedor donde se inserta. En caso de no indicarse nada en el tag ***maxoccurs*** la cardinalidad es 1.

Estructura de archivos XML de precarga de datos

Se trata de archivos de datos modelo que se aplican al crear una nueva página en un documento. Difieren de los archivos XML de datos en que los siguientes datos están parametrizados y se rellenan en run-time:

1. **gu_pageset**: identificador único del conjunto de páginas al que pertenece.

2. **gu_microsite**: identificador único del documento de usuario (newsletter o sitio web).
3. **gu_pagex**: identificador único de la página.
4. **gu_container**: identificador único del contenedor base.
5. **page_title**: descripción de la página.

Cuando insertamos una nueva página en la producción de un sitio web o creamos una nueva newsletter, la nueva página aparecerá por defecto con la información contenida en su archivo XML de precarga de datos correspondiente.

Estructura de archivos XSL de presentación de página

Cada página generada desde el módulo de producción de contenidos se basa en una plantilla XSL que define el modo en que se muestran los contenidos en pantalla. Para comprender el contenido y comportamiento de un archivo XSL de presentación de página debemos conocer los siguientes conceptos:

1. **Cabeceras y juego de caracteres:** se trata de un conjunto de líneas conformes al estándar XML que indican el tipo de archivo XML y el juego de caracteres utilizado en la información contenida en el archivo.
2. **Recogida de parámetros de aplicación:** en esta sección se recogen los siguientes parámetros de la aplicación; dominio, area de trabajo, ruta al servidor de imágenes, conjunto de páginas actual y ruta al servidor de archivos. Estos parámetros nos servirán para aplicar filtros, indicar rutas y recuperar contenidos desde atributos.
3. **Definición de la plantilla de salida:** una vez escritas las cabeceras y recogida de parámetros comienza la plantilla de salida, típicamente HTML.
4. **Estilos CSS:** cada plantilla tiene asociado un archivo CSS por gama cromática y tipografía con un conjunto de estilos de visualización. Para poder utilizarlo debemos referenciarlo en la construcción del HTML de salida, mediante el tag ***LINK*** en la parte ***HEAD***. Dichos estilos podrán aplicarse a los elementos HTML.
5. **Marcado de zonas para detección desde el módulo de producción de contenidos:** la plantilla XSL se aplica en modo visualización y en modo edición. En el último caso es posible marcar zonas donde se insertarán bloques para convertir la producción de contenidos en un proceso intuitivo. Para marcar una zona deemos emplear tags ***DIV*** o ***SPAN***, con el nombre del bloque que va a insertarse en el atributo ***id***.

6. **Nomenclatura de zonas:** para dar nombre a las zonas definidas debemos utilizar el nombre del bloque que podemos encontrar en el archivo XML de metadatos. En caso de tener cardinalidad múltiple irá seguido de un guión bajo y un número que indique la posición, empleando para ello la función XSLT ***position()***.
7. **Iteradores de información basados en un mismo metabloque:** para mostrar todos los párrafos de textos consecutivos de un mismo bloque empleamos iteradores ***for-each*** y los mostramos mediante el uso de la función XSLT ***position()***.
8. **Ordenación de información en una iteración:** para ordenar los párrafos de texto dentro de una iteración debemos incluir una cláusula ***sort*** de XSLT que ordene por el atributo ***id*** del elemento de información.

Estructura de archivos CSS de estilos

Cada plantilla tiene asociado un archivo CSS por gama cromática y tipografía. Esto significa que, si una plantilla permite el uso de cuatro gamas estilos de color y tres tipos de letra, existirán doce archivos CSS donde se definen estilos que podrán utilizarse en la construcción del HTML de cada página definida en un archivo XSLT. Los estilos que aparecen en un archivo CSS permiten definir atributos como colores de background y foreground, tipos de letra de literales que aparecen en plantilla, etc.

Los archivos CSS son opcionales, si no utilizamos ninguna referencia desde el código HTML escrito en las hojas de estilo XSLT no hace falta crear archivos CSS de estilo, aunque su uso es muy recomendable.

Nomenclatura de archivos

A continuación se muestra una relacón de la nomenclatura utilizada para dar nombre a los distintos tipos de archivo involucrados:

Archivo	Nomenclatura
Metadatos	***Nombre_contenedor***.xml
Precarga de datos	***Nombre_contenedor***_datatemplate.xml
Presentación	***Nombre_contenedor***.xsl
Estilos	***Tipografía***.css
Thumbnail	***Nombre_microsite***.gif

3 Estructura de directorios

- **Archivos de datos XML:** estos archivos los genera la aplicación al utilizar el módulo de producción de contenidos y se almacenan en *<storage>/domains/<domain>/workareas/<workarea>/apps/<app>/data*
- **Archivos de metadatos XML:** los archivos de metadatos de cada plantilla se almacenan en *<storage>/xslt/templates*
- **Archivos de precarga de datos XML:** habrá uno por contenedor de una plantilla y se almacenan en *<storage>/xslt/templates*
- **Archivos de presentación de página XSL:** existe uno por contenedor de una plantilla y se almacenan en *<storage>/xslt/templates*
- **Archivos de hojas de estilo CSS:** existe uno por color y tipografía y se almacenan en *<webserver>/styles/<template>/<color>*
- **Imágenes de plantilla GIF/JPEG:** podemos tener distintos juegos de imágenes por plantilla, dependiendo del estilo cromático, por tanto se almacenan en *<webserver>/styles/<template>/<color>*
- **Thumbnails de plantilla GIF/JPEG:** cada plantilla puede tener asociada una imagen de thumbnail representativa que aparece en el asistente de producción de contenidos cuando creamos un nuevo documento. Se almacenan en *<imgserver>/styles/thumbnails* y deben tener el mismo nombre que el contenedor asociado con extensión GIF.

4 Estructura de datos en SGBDR

La aplicación sólo mantiene una tabla con información relativa a los conjuntos de plantillas de producción de contenidos. La tabla **k_microsites** tiene un registro por cada plantilla y consta de los siguientes campos:

- **gu_microsite:** contiene un identificador único global.
- **nm_microsite:** contiene el nombre de la plantilla.
- **path_metadata:** contiene la ruta relativa al archivo de metadatos XML desde <storage>.
- **id_app:** identificador de la subaplicación que hace uso de la plantilla (newsletter o sitio web). Este identificador proviene de la tabla de lookup **k_apps**.
- **dt_created:** contiene la fecha de creación de la plantilla.
- **tp_microsite:** reservado. Contiene el tipo de plantilla.
- **gu_workarea:** reservado. Preparado para crear plantillas utilizables únicamente en una workarea.

La tabla **k_pagesets** contiene un registro por cada documento que instancia una plantillas de **k_microsites** con los siguientes campos:

- **gu_pageset:** identificador único global del documento.
- **dt_created:** fecha de creación.
- **dt_modified:** fecha de modificación.
- **nm_pageset:** Nombre del documento (típicamente el título de la página HTML)
- **gu_workarea:** GUID del Área de Trabajo a la cual pertenece el documento.
- **path_data:** ruta relativa al fichero de datos XML desde la raiz de storage.
- **id_language:** idioma del documento.
- **gu_microsite:** GUID del microsite que contiene la plantilla para este documento.
- **id_status**: Estado de publicación.
- **gu_company:** GUID de la compañía propietaria del documento.
- **gu_project:** GUID del proyecto al cual pertenece el documento.
- **tx_comments**: Comentarios.

La tabla **k_pageset_pages** tiene un registro por cada contenedor (página) en una instancia de plantilla (PageSet).

- **gu_page:** contiene un identificador único global.
- **pg_page:** Número de la página (1, 2, .. n)
- **dt_created:** fecha de creación.
- **dt_modified:** fecha de última modificación.
- **tl_page:** Título de la página.
- **gu_pageset:** GUID del PageSet donde está contenida la página.

path_page: contiene la ruta relativa al archivo de datos XML desde *storage*.

5 Proceso de creación de plantillas

Para crear una nueva plantilla de documento hemos organizado el proceso en una serie de pasos sencillos:

1. Decidir las **gamas cromáticas** y distintas **tipografías** permitidas en la plantilla.

2. Definir las estructuras de información permitidas y su agrupación en contenedores mediante la creación del **archivo de metadatos** XML.
3. Definir la información de precarga ejemplo, creando los **archivos de precarga** de datos XML.
4. Crear, si es necesario, los **archivos de estilo** CSS.
5. Definir los **archivos de presentación** XSL por cada página de información.
6. Crear las **imágenes de plantilla** GIF/JPEG.
7. Crear la **imagen de thumbnail** asociada GIF/JPEG.
8. Dar de alta la nueva plantilla en **base de datos**.

A continuación pasamos a explicar de forma detallada cada uno de los 8 pasos.

Paso 1: Gama cromática y tipografías

El primer paso en la construcción de una nueva plantilla es decidir cuáles van a ser los colores y las tipografías permitidos en la producción de documentos basados en la nueva plantilla. Dependiendo del número de colores y tipografías a incluir en la nueva plantilla tendremos que definir más o menos archivos de estilos CSS y diseñar más o menos imágenes de plantilla. La definición de gama cromática y tipografías se realiza en el archivo de metadatos en las secciones XML ***<colors>*** y ***<fonts>***. A continuación se muestra un ejemplo de declaración de dichas secciones:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<microsite xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="file:///opt/knowgate/knowgate/stora
ge/xslt/schemas/microsite.xsd"
guid="c0a80146f64790ac7c100003fcc20a22">
  <name>Contemporary</name>

  <fonts>
    <family>Arial</family>
    <family>Times</family>
    <family>Verdana</family>
  </fonts>

  <colors>
    <color>Azul-Naranja</color>
    <color>Verde-Amarillo</color>
    <color>Burdeos-Negro</color>
  </colors>
```

En éste ejemplo se definen tres estilos cromáticos (*Azul-Naranja, Verde-Amarillo y Burdeos-Negro*) y tres tipografías (*Arial, Times y Verdana*). Esto hace obligatorio la creación de varios directorios:

- <webroot>/styles/Contemporary/Azul-Naranja
- <webroot>/styles/Contemporary/Verde-Amarillo
- <webroot>/styles/Contemporary/Burdeos-Negro

En el interior de cada directorio aparecerán archivos para los siguientes items:

- Archivos de estilos CSS, uno por cada tipografía (*Arial.css, Times.css y Verdana.css*)
- Imágenes de plantilla GIF/JPEG; en cada directorio tendremos todas las imágenes correspondientes al estilo cromático.

A pesar de ir incluido en el archivo de metadatos, por su sencillez, hemos preferido comenzar por gamas cromáticas y tipografías antes de explicar el archivo de metadatos.

Paso 2: Definición del archivo de metadatos XML

El siguiente paso es diseñar la estructura de los contenidos que será permitida en la nueva plantilla. Para ello debemos decidir:

1. Número de **contenedores de información** (páginas) posibles. En el caso de estar diseñando una plantilla de newsletter sólo habrá un contenedor.

   Siguiendo con el ejemplo anterior pasamos a la creación de los contenedores de información. Por tratarse de una newsletter sólo existe un contenedor y se define de la siguiente manera:

   ```
   <containers>
         <container guid="3cd64cc2a9184d2c96b2a2fdf8f989cc">
   ```

   Los ***guids*** que generemos deben ser únicos y para ello podemos emplear a través del API el método ***generateUUID()*** de la clase ***Gadgets*** del paquete java ***com.knowgate.misc***. Para más información consultar la guía del programador.

2. Para **cada contenedor** debemos determinar el nombre del contenedor y el nombre del archivo XSL de presentación asociado.

   ```
   <name>Contemporary</name>
   <template>Contemporary.xsl</template>
   ```

El nodo ***name*** indica el nombre del contenedor. En el caso de newsletters sólo habrá uno. Sin embargo, cuando estemos diseñando una plantilla para sitio web tendremos varios contenedores con distintos nombres que darán título a las páginas del sitio web. El nodo ***template*** contiene el nombre del archivo de presentación XSL que explotará la información generada en base a este contenedor.

3. Definir para cada contenedor los posibles **metabloques** permitidos (piezas de información insertables en el contenedor). En cada metabloque se debe indicar: un identificador, un número secuencial de orden, nombre, objetos permitidos (con tipo, nombre y cardinalidad) y número máximo de ocurrencias del metabloque en el contenedor.

   A continuación se muestra un ejemplo de definición de dos metabloques. El primero es una estructura que permite insertar una imagen. El segundo es una estructura que permite insertar una imagen, un párrafo y de 1 a 9 párrafos adicionales.

A su vez el primer metabloque determina que se podrá insertar un único bloques de dicho tipo mientras que en el caso del segundo metabloque podremos insertar de 1 a 9 bloques de dicho tipo.

```
    <metablocks>

        <metablock id="LOGO_SUPERIOR" listingOrder="1">
            <name>Logotipo Superior</name>
            <template>#inline:.</template>
            <objects>i:LOGOTIPO_CENTRAL</objects>
            <maxoccurs>1</maxoccurs>
        </metablock>

        <metablock id="ARTICULO" listingOrder="2" allowHTML="false">
            <name>Articulo</name>
            <template>#inline:.</template>
            <objects>i:IMAGEN_ARTICULO,
                    p:TITULO_ARTICULO,
                    p:PARRAFO_ARTICULO_[1..10],
                    p:ENLACE_ARTICULO_[0..1]
            </objects>
            <maxoccurs>10</maxoccurs>
        </metablock>

    </metablocks>

   </container>

  </containers>

</microsite>
```

El attributo id debe ser un nombre único para cada metabloque.

El atributo listingOorder debe ser un ordinal consecutivo único para cada metabloque.

El atributo allowHTML indica si los párrafos del metabloque admitirán HTML o sólo texto plano cuando sus contenidos sean editados desde el editor WYSIWYG.

Como podemos observar debemos prestar especial atención a la sintaxis de definición de items de información internos a un metabloque. Cada item se expresa en el interior del nodo ***objects*** y, separados por comas, debemos indicar para cada uno:

- **Tipo de objeto**: puede ser ***p*** o ***i*** dependiendo de si se trata de un párrafo de texto o una imagen.
- **Nombre del objeto**: un literal en mayúsculas donde los guiones bajos son lícitos. Si va acompañado de cardinalidad debe haber un guión bajo antes del corchete de apertura.
- **Cardinalidad**: entre corchetes un rango que indica mínimo y máximo número de ocurrencias, pudiendo ser 0.

Con una estructura de metabloques como la definida anteriormente podríamos insertar desde el módulo de producción de contenidos, en una página de información, una imagen y hasta 10 bloques de información, compuestos cada uno por un párrafo de título, de 1 a 10 párrafos de texto y opcionalmente 1 párrafo de texto adicional.

Paso 3: Definición de los archivos de precarga de datos XML

En un archivo de precarga de datos para una página se define la información que aparecerá al crear una nueva página basada en un contenedor. En éste archivo no tendremos información estructural, tal como ocurre en el archivo de metadatos, sino que tendremos los contenidos en si mismos, basados en las estructuras definidas en el archivo de metadatos. Estos archivos se crean para realizar la labor de producción de contenidos desde la aplicación un proceso más intuitivo.

Lo lógico es que el responsable de producir los contenidos, una vez creada la página, vaya reemplazando estos textos e imágenes de precarga por los contenidos reales que desea publicar en el documento final.

Recordemos las nociones básicas de la estructura de un archivo de precarga de datos es muy sencilla:

1. Se define un conjunto de páginas (**pageset**).
2. En el interior del pageset puede haber varias páginas (**page**).
3. En el interior de una página puede haber varios bloques en una zona **blocks**.
4. Cada bloque puede tener varios objetos de información: párrafos de texto (**paragraphs**) e imágenes (**images**).

Siguiendo con el ejemplo anterior, vamos a definir un archivo de precarga de datos XML para el contenedor *"Contemporary"* que hemos definido en el archivo de metadatos XML. En este archivo de precarga definiremos los siguientes contenidos:

- Un bloque con un logotipo
- Un bloque de artículo con una imagen, un título y un párrafo.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl"?>
<pageset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation="file:///opt/knowgate/knowg
       ate/storage/xslt/schemas/pageset.xsd"
       guid=":gu_pageset">
  <microsite>:gu_microsite</microsite>
  <font>Verdana</font>
  <color>Azul-Naranja</color>
```

Una vez definidas las cabeceras y elegida una tipografía y color para el ejemplo procedemos a definir los bloques de información.

```
<pages>
 <page guid=":gu_pagex">
  <title><![CDATA[:page_title]]></title>
  <container>:gu_container</container>
   <blocks>
    <block id="001">
      <metablock>LOGO_SUPERIOR</metablock>
      <tag>Logotipo Superior</tag>
      <paragraphs>
       <paragraph id="REMOVABLE"></paragraph>
      </paragraphs>
      <images>
       <image id="REMOVABLE"></image>
       <image id="IMAGEN_ARTICULO">
        <url></url>
        <path>http://img.test.com/style/com/img_es.gif</path>
        <alt><![CDATA[Texto Alternativo]]></alt>
        <width>48</width>
        <height>120</height>
       </image>
      </images>
      <zone/>
    </block>
```

Como podemos ver el primer bloque se basa en el metabloque ***LOGO_SUPERIOR*** que permite incluir una imagen, la cual aparece dentro de la sección ***images*** (***LOGOTIPO_CENTRAL)***, y se define una url de salto, la ruta a la imagen, un texto alternativo y las dimensiones. Tanto la sección de párrafos como imágenes deben llevar incluido un párrafo o imagen vacíos con nombre ***REMOVABLE*** al inicio de cada sección.

```
        <block id="002">
           <metablock>ARTICULO</metablock>
           <tag>Articulo</tag>
           <paragraphs>
            <paragraph id="REMOVABLE"></paragraph>
            <paragraph id="TITULO_ARTICULO">
             <text><![CDATA[Titulo del articulo]]></text>
             <url></url>
            </paragraph>
            <paragraph id="PARRAFO_ARTICULO_1">
             <text><![CDATA[Texto del articulo]]></text>
             <url></url>
            </paragraph>
           </paragraphs>
           <images>
            <image id="REMOVABLE"></image>
            <image id="LOGOTIPO_CENTRAL">
             <url>http://www.su-dominio.es/</url>
             <path>http://img.test.com/style/com/def_es.gif</path>
             <alt><![CDATA[Texto Alternativo]]></alt>
             <width>468</width>
             <height>60</height>
            </image>
           </images>
           <zone/>
        </block>

      </container>
    </page>
  </pages>
</pageset>
```

En éste segundo bloque podemos observar que ésta basado en el metabloque ***ARTICULO***, definido en el archivo de metadatos y que la nomenclatura de los objetos internos (imágenes y párrafos) siguen la nomenclatura definida en el archivo de metadatos. Todos los literales (textos de párrafos y textos alternativos de imagen) deben ir encerrados en estructuras XML ***CDATA,*** para garantizar la correcta presentación en pantalla de todo el juego de caracteres.

De éste modo podremos crear tantos bloque de ejemplo en el archivo de precarga de datos como deseemos, siempre respetando la restricción ***maxoccurs*** indicada en la definición de cada metabloque.

El archivo debe almacenarse en su directorio correspondiente (véase capítulo 3 de ésta guía) y respetando la nomenclatura indicada en el apartado ***Nomenclatura*** del capítulo 2 de ésta guía.

Paso 4: Creación de archivos de estilos CSS

Los archivos CSS son opcionales y sólo son necesarios cuando queramos modularizar los estilos de presentación que usemos en la definición de las hojas de presentación XSL.

Debe existir un archivo de estilo CSS por color y tipo de letra, por lo que se almacenarán en ***<webserver>/styles/<template>/<color>***. El directorio indica el color al que se aplica y en dicho directorio habrá varios archivo, uno por tipografía. En el ejemplo tendremos 3 archivos en cada directorio de colores: Arial.css, Times.css y Verdana.css.

Para referenciar al archivo de estilos desde el HTML generado desde el archivo de presentación XSL debemos incluir la siguiente linea en la sección ***<HEAD>***:

```
<link
   rel="stylesheet"
   type="text/css"
   href="{$param_imageserver}/styles/Contemporary/{pageset/color}/
         {pageset/font}.css"/>
```

Para más información visite la referencia CSS del World Wide Web Consortium en http://www.w3.org/Style/CSS/.

Paso 5: Definición de hojas de presentación XSL

Cada contenedor definido en el archivo de metadatos debe tener asociado un archivo de presentación XSL. En éste archivo se define la plantilla HTML de salida para una página. Desde aquí se accederá a la información existente en el archivo de datos que introduzca el responsable de la producción de contenidos y se dará formato a dichos datos.

Dado que el diseño de una presentación es completamente libre, pasaremos a explicar los puntos clave de la construcción del archivo de presentación XSL, las estructuras de recorrido de información más útiles y el modo de acceso a la información del archivo de datos XML.

Dividiremos el proceso de creación de la hoja de presentación XSL en 8 pasos:

1. Escribir las **cabeceras** de archivo XSL
2. Recoger los **parámetros** de aplicación
3. **Inicio** de hoja de presentación XSL
4. Inicio de código **HTML**
5. Referenciar el archivo de estilos **CSS**
6. **Acceso a información** del archivo de datos XML
    a. Párrafos de texto
    b. Imágenes
7. **Iteradores de información** secuencial
    a. Iteradores de información
    b. Ordenación de contenidos
8. **Cierre** de hoja de presentación XSL

A continuación pasamos a explicar de forma detallada cada uno de los 8 pasos.

1. Cabeceras de archivo XSL

Todo archivo XSL requiere la definición de cabeceras donde se indique:

- El DTD de validación del documento XSL.
- El namespace de instrucciones XSLT.
- El método de salida y juego de caracteres utilizado.

Para ello debemos comenzar cualquier archivo XSL con las siguientes declaraciones:

```
<!DOCTYPE xsl:stylesheet SYSTEM "../schemas/entities.dtd">

<xsl:stylesheet
   version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output
   method="html"
   version="4.0"
   encoding="ISO-8859-1"
   omit-xml-declaration="yes"
   indent="no"
   media-type="text/html"/>
```

2. Parámetros de aplicación

A continuación se deben recoger los parámetros de aplicación para poder utilizarlos en la sección HTML:

```
<xsl:param name="param_domain" />
<xsl:param name="param_workarea" />
<xsl:param name="param_imageserver" />
<xsl:param name="param_pageset" />
<xsl:param name="param_storageserver" />
```

3. Inicio de la hoja de presentación

La siguiente línea indica que el acceso a la información almacenada en los nodos del archivo de datos XML va a ser referenciada desde la raíz:

```
<xsl:template match="/">
```

4. Inicio de código HTML

A partir de éste punto podemos escribir la presentación HTML. Debemos tener en cuenta que al estar insertado en un archivo XSL todos los tags deben estar balanceados.

5. Referencia al archivo de estilos CSS

Se referencia al archivo de estilos en la sección ***head*** de ***HTML***, tal como se explica en el Paso 4 del proceso de creación de plantillas.

6. Acceso a contenidos

Desde el interior del código HTML podemos recuperar información del archivo de datos XML, donde se encuentran la información editada por el responsable de producción de contenidos. Para ilustrar el modo de acceso a la información mostraremos un ejemplo de recuperación de párrafos e imágenes:

Acceso a la información contenida en un párrafo:

```
<xsl:if
      test="pageset/pages/page/blocks/
            block[metablock='BIENVENIDA']/paragraphs/
            paragraph[@id='BIENVENIDA_TEXTO']/text!=''">
```

```
<xsl:value-of
      select="pageset/pages/page/blocks/
            block[metablock='BIENVENIDA']/paragraphs/
            paragraph[@id='BIENVENIDA_TEXTO']/text"
      disable-output-escaping="yes"/>

</xsl:if>
```

- La primera etiqueta (***xsl:if***) contiene una comprobación contra el archivo de datos XML de que exista un bloque ***BIENVENIDA*** que contenga un párrafo ***BIENVENIDA_TEXTO***. La etiqueta interior (***xsl:value-of***) recupera el valor del texto del párrafo.

- **Importante:** se debe incluir el atributo ***disable-output-escaping*** con valor ***"yes"*** en todos los valores de texto de párrafos que permitan la inserción de tags HTML en el contenido (atributo allowHTML="trae" en la definición del metabloque). Para los párrafos que permitan sólo texto plano usar ***disable-output-escaping="no"***

Acceso a la información contenida en una imagen:

```
<xsl:variable
   name="logoSuperior"
   select="pageset/pages/page/blocks/
   block[metablock='LOGO_SUPERIOR']/images/
   image[@id='LOGOTIPO_CENTRAL']"/>

<img
   alt="{$logoSuperior/alt}"
   src="{$logoSuperior/path}"
   width="{$logoSuperior/width}"
   height="{$logoSuperior/height}"
   border="0"/>
```

- En éste caso definimos una variable XSL cuyo nombre es ***logoSuperior*** y recuperamos en ella la ruta al nodo de imagen del archivo de datos XML; una imagen con identificador ***LOGOTIPO_CENTRAL*** existe en el bloque de metatipo ***LOGO_SUPERIOR***. La etiqueta ***img*** utiliza la variable ***logoSuperior*** para recuperar la información de texto alternativo (***alt***), ruta de archivo (***path***), y dimensiones (***width*** y ***height***).

7. Iteradores de información

La instrucción XSL ***xsl:for-each*** nos permite recuperar conjuntos de información secuencial, normalmente varios bloques del mismo tipo o varios párrafos dentro de un bloque.

En el siguiente ejemplo veremos cómo recuperar varios bloques de un mismo tipo y en cada bloque todos sus párrafos siempre que su identificador comience por ***PARRAFO_ARTICULO***:

```
<xsl:for-each
         select="pageset/pages/page/blocks/
         block[metablock='ARTICULO']">

   <xsl:sort select="@id"/>

   <xsl:for-each
         select="paragraphs/
               paragraph[starts-with(@id,'PARRAFO_ARTICULO')]">

         <xsl:sort select="@id"/>

         <xsl:value-of select="text"/>

         <br/>

   </xsl:for-each>

</xsl:for-each>
```

- Definimos un iterador ***xsl:for-each*** que recorre todos los bloques cuyo tipo de metabloque sea ***ARTICULO.***
- Definimos un iterador ***xsl:for-each*** que recorre todos los párrafos de un bloque cuyo atributo id comience por ***PARRAFO_ARTICULO***.
- Ordenamos por identificador la lista y vamos mostrando el texto de cada párrafo.

8. Cierre de la hoja de presentación

Por último debemos cerrar la hoja con las siguientes líneas:

```
</xsl:template>
</xsl:stylesheet>
```

Recomendamos profundizar en la sintaxis de XSLT. Si desea más información visite la siguiente url: http://www.w3.org/Style/XSL/.

Paso 6: Diseño de imágenes de plantilla y thumbnail

Para que la plantilla funcione correctamente debemos guardar todas las imágenes propias de la plantilla en su directorio. Véase el Capítulo 3 de ésta guía.

Paso 7: Indexación de la plantilla en base de datos

Una vez terminada la plantilla debe insertarse un registro en base de datos, en la tabla ***k_microsites***. A continuación se muestran los valores que deben insertarse en cada campo:

- **gu_microsite**: debemos insertar el GUID de 32 caracteres que aparece en el archivo de metadatos XML, en el atributo ***guid*** del nodo microsite.
- **nm_microsite**: debemos insertar el literal que aparece en el archivo de metadatos XML, en el nodo ***name***.
- **path_metadata**: debemos insertar la ruta al archivo de metadatos xml de la siguiente manera; si el nombre del archivo es ***template1.xml*** debemos insertar ***xslt/templates/template1.xml***.
- **id_app**: si se trata de una newsletter debemos insertar el valor ***13***, si es un sitio web debemos insertar el valor ***14***. Estos valores están definidos en base de datos, en la tabla de aplicaciones ***k_apps,*** y corresponden típicamente a las aplicaciones ***Mailwire*** (Producción de contenidos con publicación en Newsletter) y ***Web Builder*** (Producción de contenidos con publicación en Sitio Web) , respectivamente.

6 Proceso de validación de plantillas

Para comprobar que la nueva plantilla funciona correctamente debemos verificar desde el módulo de producción de contenidos:

1. La plantilla está disponible desde la aplicación en la lista de plantillas que aparece al crear un nuevo documento.
2. El nuevo documento aparece con el diseño HTML y con los datos de precarga correctos.
3. Al pasar el puntero del ratón sobre los bloques existentes se iluminan correctamente las zonas del documento.
4. Al tratar de crear un nuevo bloque de datos aparece una lista con todos los bloques definidos en el archivo de metadatos.

Para validar los documentos XML podemos emplear el API de hipergate desde línea de comandos con la siguiente sintaxis:

```
java com.knowgate.dataxslt.PageSet parse <ruta_al_archivo>
```

7 Caso práctico paso a paso

Para ilustrar de un modo práctico la construcción de una plantilla vamos a seguir todo el proceso con un ejemplo real cuyos fuentes están disponibles en la distribución de hipergate. Se trata de la plantilla de newsletter ***Basic,*** y podemos encontrar los archivos en los siguientes directorios:

Archivos de partida:

- Página original XHTML:

  <webroot>/storage/xslt/z_templates_newsletters/

- Imágenes originales GIF/JPEG:

  <webroot>/storage/xslt/z_templates_newsletters/basic_archivos/

- Hoja de estilos original CSS:

  <webroot>/storage/xslt/z_templates_newsletters/basic_archivos/

Archivos resultado:

- Nuevas hojas de estilos CSS:

  <webroot>/web/styles/basic/Blanco-Negro/

- Thumbnail generado:

  <imgroot>/styles/thumbnails/

- Imágenes finales:

  <imgroot>/styles/basic/Blanco-Negro/

- Archivo de metadatos:

  <webroot>/storage/xslt/templates/

- Archivo de precarga de datos:

  <webroot>/storage/xslt/templates/

- Presentación XSL:

```
<webroot>/storage/xslt/templates/
```

El escenario de partida en el cual nos encontramos es el siguiente: disponemos de tres archivos; una página HTML, una hoja de estilos CSS y unas cuantas imágenes GIF/JPEG que son referenciadas desde la página HTML. A partir de este punto debemos crear todos los archivos necesarios para la nueva plantilla. Seguiremos los 7 pasos para la creación de nuevas plantilas.

Paso 1: Gama cromática y tipografías

En el ejemplo ***Basic*** se ha optado por tener una única cromaticidad (***Blanco-Negro***) y tres tipografías disponibles (***Geneva, Times y Verdana***). Podemos observar que en archivo original, ***style.css***, hay definidos 4 estilos:

- ***Htmlbody***: color de textos y fondo general.
- ***Title***: tipo y tamaño de letra para títulos.
- ***Subtitle***: tipo y tamaño de letra para subtítulos.
- ***Plaintext***: tipo y tamaño de letra para texto general.

Paso 2: Archivo de metadatos

A la hora de diseñar el archivo de metadatos debemos analizar el documento HTML y detectar las estructuras de información: párrafos e imágenes y posibles bloques agregadores de información.

Podemos ver que el archivo original se compone de la siguiente información:

- Logotipo principal
- Título de Newsletter
- Subtítulo de Newsletter
- Título de sumario
- Líneas de sumario
- Saludo personalizado
- Texto de bienvenida
- Artículos con título y texto

Nuestra solución es organizar la información de la siguiente manera:

- Bloque con una imagen para el ***logo superior***
- Bloque con dos párrafos (fecha y texto) para ***minicabecera***
- Bloque con 1 a 10 párrafos para ***sumario***
- Bloque ***artículo*** con 3 párrafos (título sumario, párrafo sumario y título artículo), de 1 a 10 párrafos para el artículo y un párrafo para enlace opcional.

Si abrimos el archivo de metadatos ***Basic.xml*** en el directorio ***<webroot>/storage/xslt/templates/*** podemos ver la definición de las diferentes estructuras de bloque.

Como se puede ver el proceso de creación del archivo de metadatos es un proceso de análisis y optimización de estructuras de información.

Paso 3: Archivo de precarga de datos

Como ya hemos visto en capítulos anteriores el archivo de precarga es un archivo con datos iniciales que servirán como ejemplo en la creación de un nuevo documento basado en la plantilla que estamos creando. La información de precarga debe basarse en las estructuras de bloque definidas en el archivo de metadatos.

Si observamos el contenido del archivo ***Basic.datatemplate.xml*** en el directorio ***<webroot>/storage/xslt/templates/*** encontramos la siguiente información de precarga:

- Un bloque de tipo ***MINICABECERA*** con la fecha del sistema y texto de cabecera.
- Un bloque de tipo ***SUMARIO*** con 4 párrafos de sumario.
- Un bloque de tipo ***ARTICULO*** con un título y 3 párrafos de texto del artículo.
- Un bloque de tipo ***LOGOSUPERIOR*** con la imagen del logotipo.

Además de estos bloques de ejemplo podríamos insertar bloques de tipo ***DESTACADO***, ***LINKS*** y ***BIENVENIDA***.

Paso 4: Hojas de estilos

Creamos el directorio ***<webroot>/web/styles/basic/Blanco-Negro/*** y copiamos el archivo de estilos original a 3 archivos: ***Geneva.css***, ***Times.css*** y ***Verdana.css***.

Cada uno de los 3 archivos será modificado internamente para mantener únicamente el tipo de letra que representa. En cada una de las definiciones de estilos (***title, subtitle y plaintext***) se cambia el tipo de letra.

Paso 5: Presentación XSL

A continuación paso a explicar las partes más importantes del código de la presentación XSL de la plantilla Basic:

1. **Recogida de parámetros de aplicación necesarios (líneas 5-9)**
   Estas cinco líneas declaran cinco parámetros que rellenará la aplicación con una serie de valores (***dominio***, ***workarea***, ***servidor de imágenes***, ***identificador de documento*** y ***servidor de ficheros***) y los cuales podremos utilizar desde la presentación XSL.

2. **Declaración de uso de hoja de estilos CSS (línea 21)**
   En la línea 21 vemos el modo de componer la ruta a la hoja de estilos CSS: debemos emplear el parámetro de aplicación param_imageserver y acceder a la información XML ***pageset/color*** y ***pageset/font***.

3. **Referencia a imágenes de plantilla (línea 29)**
   En la línea 29 vemos un ejemplo de acceso a una imagen de plantilla. Para componer la ruta empleamos el parámetro de aplicación ***param_imageserver*** y la información XML ***pageset/color***.

4. **Recuperación de la imagen LOGOTIPO (líneas 51-58)**
   Las líneas 51 a 58 contienen las instrucciones de recuperación del logotipo. En la línea 51 declaramos una zona iluminable, ***LOGOSUPEROR_1***, lo cual permitirá detectar con facilidad dicha parte de la plantilla desde la aplicación desde el módulo de producción de contenidos. A continuación se encuentran 2 instrucciones condicionales opuestas que verifican si la imagen tiene o no rellenas las dimensiones. En caso afirmativo se vuelca una línea HTML con un tag de imagen que contiene los atributos de dimensiones. Para recuperar la imagen del archivo de datos XML utilizamos la parte de la ruta almacenada en la variable ***logo*** (ver línea 44) y accedemos a cada pieza de información de la estructura ***image***.

5. **Recuperación de información de MINICABECERA (líneas 61-67)**
   Declaramos una zona iluminable en la línea 61. En la línea 64 se almacena en una variable la ruta XML al bloque de ***MINICABECERA***. Las líneas 65 y 66 emplean dicha variable y recuperan los diferentes párrafos de la estructura.

6. **Recuperación de información de SUMARIO (líneas 75-80)**
   En estas líneas se compone el sumario de artículos de la newsletter. Se declara un iterador que recorre cada uno de los párrafos que comienzan por ***TITULO-ARTICULO*** de los bloques de tipo ***ARTICULO***. Se ordena el resultado por el atributo ***id*** de los bloques. En la línea 78 se muestra la información contenida en el nodo ***text*** de cada párrafo con un enlace de salto a la dirección almacenada en el nodo ***url***.

7. **Recuperación de información de BIENVENIDA (líneas 85-88)**
   Declaramos una zona iluminable y almacenamos en una variable la ruta XML al bloque de tipo ***BIENVENIDA***. En la línea 86 se recupera la información del párrafo cuyo ***id*** es ***BIENVENIDA-TEXTO***.

8. **Recuperación de información de DESTACADO (líneas 90-117)**
   Se recupera la información de bloques de tipo ***DESTACADO*** de forma similar a la recuperación de información de SUMARIO.

9. **Recuperación de información de ARTICULOS (líneas 119-124)**
   Se declara un iterador que recorre cada uno de los bloques de tipo ***ARTICULO***. En cada iteración se invoca a la subplantilla de ARTICULO, definida en las líneas 166-203, pasándole como parámetro el ***id*** de bloque.

10. **Recuperación de información de LINKS (líneas 126-139)**
    Se recupera la información de bloques de tipo ***LINKS*** de forma similar a la recuperación de información de SUMARIO.

11. **Declaración de subplantilla de ARTICULO (líneas 166-203)**
    En la línea 168 recuperamos el ***id*** del bloque. Almacenamos en una variable el párrafo del bloque que comienza por ***TITULO***. En la línea 175 declaramos una zona iluminable para el artículo actual. En la línea 178 volcamos la información XML del nodo ***text*** del párrafo de título. En la línea 190 declaramos un iterador que recorre todos los párrafos del artículo actual cuyo id comienza por ***PARRAFO*** y en la línea 192 volcamos la información XML del nodo ***text***. En la línea 194 declaramos un iterador que recorre todos los párrafos del artículo

cuyo ***id*** comienza por ***ENLACE***. En la línea 196 se vuelca un link HTML compuesto por la información XML del párrafo que contiene el enlace (***url***) y el texto (***text***).

Paso 6: Imágenes de plantilla y thumbnail

Si en el archivo de precarga o en la hoja XSL referenciamos a alguna de las imágenes originales estas deben ser copiadas a una URL accesible desde la aplicación cuando se produzca el proceso de rendering. Copiamos las imágenes al directorio ***<imgroot>/styles/basic/Blanco-Negro/.*** Esta es la ruta que debe utilizarse desde el archivo de precarga o la hoja de estilo XSL para llegar a las imágenes.

Para crear el thumbnail debemos abrir el HTML original desde un browser y capturar la pantalla. Luego, desde un programa de retoque fotografico reescalamos la imagen a 242x282 pixels y la guardamos en ***<imgroot>/styles/thumbnails/*** con el nombre ***Basic.gif***.

Paso 7: Indexación en base de datos

Insertamos un registro base de datos, en la tabla ***k_microsites***. A continuación se muestran los valores que deben insertarse en cada campo:

- **gu_microsite**: '0c7c148619c944819576fcec428e7350'
- **nm_microsite**: 'Basic'
- **path_metadata**: 'xslt/templates/Basic.xml'
- **id_app**: 13

Los valores de los 3 primeros campos se obtienen del archivo de metadatos ***Basic.xml*** del directorio ***<webroot>/storage/xslt/templates/***; el atributo ***microsite.guid*** y los nodos ***name*** y ***template*** respectivamente.

El valor de ***id_app*** es 13 ya que es el valor almacenado en la tabla k_apps para Mailwire (Newsletters).

8 Referencias sobre tecnología XSL

Si desea profundizar más en la materia le animamos a que visite los siguientes enlaces donde encontrará tutoriales, herramientas y libros sobre el mundo XSL:

Tutoriales

World Wide Web Consortium	**The XSL family**	www.w3.org/Style/XSL
Página con todos los estándares sobre XSL del W3C		
DevGuru	**XSLT Tutorial**	www.devguru.com/Technologies/xslt/quickref/xslt_intro.html
Tutorial sobre XSLT		
W3Schools	**Tutorials**	www.w3schools.com
Tutoriales sobre XML, XSL, DTD, DOM, XSD, XPath		

Herramientas

Altova	**XMLSpy XML/XSLT Editor**	www.xmlspy.com
Editor muy popular de XML, XSD y XSLT, incluye un diseñador visual XSLT y depurador		
Dave Raggett	**Tidy**	tidy.sourceforge.net
Herramienta de normalización de páginas HTML a XHTML		

Bibliografía

O'Reilly/Doug Tidwell	**XSLT**	www.oreilly.com/catalog/xslt/
Enfocado en XALAN, XML-HTML		
Michael Kay	**XSLT : Programmer's Reference**	www.amazon.com
XSLT 1.0		
Bob Ducharme	**XSLT Quickly**	www.manning.com/ducharme/
Cubre el 20% de los problemas que suceden el 80% de las veces		
Khun Yee Fung	**XSLT: Working with XML and HTML**	www.amazon.com
Iniciación a XSLT		