hipergate®

Creating XSL templates

Version 7.0
April 2013

1 Introduction

Templates for contents creation

The contents production moldule allows the creation of newsletters and microwebsites. Both types of documents –newsletters and microwebsites- are created by using templates that define content disposition, color and typeset style.

hipergate includes by default a small set of predefined templates.

For creating a new template it is necessary to understand what information is required. For templates, information is divided into *Data* and *MetaData*. At this guide it is explained :

1. What is the structure of the requiered files.
2. What is the structure of directories used.
3. What information is stored at the database.
4. What is the process for creating new templates.
5. What is the process for validating new templates.

The final goal is the creation of new templates.

2 Files used by templates

The contents production module uses several file types for creating new documents:

1. **XML data file**: this file holds information entered by the end-user about the final contents themselves: texts, image urls and hyperlinks.

2. **XML metadata file**: this file contains information about the structure that a document can have.
3. **Default data XML file**: this file contains some sample texts and images for each template. This texts and images are inserted into every new document each time that the template is instantiated.

4. **XSL stylesheet**: this file describes how the data will be formated into XHTML or other ouput format.

5. **CSS cascading stylesheet**: this file holds information about colors and fonts that will we used for displaying the final document.

XML data file format for PageSets

The texts and image references set by the end-user when composing the final document are stored inside the XML data files.

These XML data files are created and updated for the application front-end, so they must not be manipulated by hand.

These are the component sections for an XML PageSet:

1. **Header and character set:** some lines that, according to the XML standards describe the document type and character set used.
```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl"?>
```

2. **Descriptor of information pages:** each PageSet XML file contains a descriptor for the set of information pages (PageSet). Each PageSet has a Global Unique IDentifier (***guid***).
```
<pageset xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="http://www.hipergate.org/x
slt/schemas/pageset.xsd"
guid="7f00000110142fe1d9d100000d6d9e3a">
```

3. **Microsite Identifier:** type of document (Microsite) that will be applied for displaying the information.
```
<microsite>c0a80146f64791d722100005b5aa3492</microsite>
```

4. **Fonts and Colors:** Font Family and color palette that will be used.
```
<font>Verdana</font>
<color>Azul-Naranja</color>
```

5. **Information for each page:** the `<page>` tag delimits information for each page. For newsletters, the XML PageSet file only contains one `<page>` tag, for websites there are several pages. Each page has its own GUID.
```
<page guid="7f00000110142fe4aef100001fb77013">
```

6. **Page Title:** editable from the end-user interface.

```
<title><![CDATA[Traditional (Jan 5 2005)]]></title>
```

7. **Container GUID:** The structure of a page is described by a metadata structure called Container. Containers are defined at the XML file for the Microsite used for displaying the current PageSet. Containers are divided into Metablocks. The Metablocks is the basic piece used for composing the structure of a page. A Metablocks con be a single image or a line of text or a discrete set of images and texts with a fixed layout.
```
<container>3cd64cc2a9184d2c96b2a2fdf8f989cc</container>
```

8. **Blocks:** Blocks are instances of Metablocks definitions. For a given pageSet its Microsite describes which Metablocks can be placed on each section on a page. Whilst Metablocks tell what can appear on a page, Blocks are what it is actually there for each final page. Each block as a numeric Id. When blocks are displayed on screen or printed, they are shown in the same order as their numeric Ids.
```
<blocks>
  <block id="001">
  <metablock>BIGTITLE</metablock>
  <tag>Title of the month</tag>
```

9. **Text paragraphs:** Inside a Block there can be zero or more text paragraphs. For each paragraph set there is a special paragraphs with id="REMOVABLE". This paragraphs is internally used by the webbuilder as the insertion point for new paragraphs and must not be removed.
```
<paragraphs>
  <paragraph id="REMOVABLE"></paragraph>
  <paragraph id="ARTICLE_TITLE">
    <text><![CDATA[Title of Article 1]]></text>
    <url></url>
  </paragraph>
  <paragraph id="ARTICLE_TEXT">
    <text><![CDATA[Text, bla, bla, bla]]></text>
    <url></url>
  </paragraph>
  <paragraph id="ARTICLE_LINK">
    <text><![CDATA[Read more]]></text>
    <url>http://www.knowgate.es/</url>
  </paragraph>
</paragraphs>
```

10. **Images:** Inside a Block there can be zero or more images.
```
<images>
  <image id="REMOVABLE"></image>
  <image id="ARTICLE_IMAGE">
  <url>http://www.knowgate.es/</url>
  <path>http://www.srv.com/styles/common/tn100.gif</path>
  <alt><![CDATA[KnowGate.es]]></alt>
```

```
  <width><![CDATA[100]]></width>
  <height><![CDATA[100]]></height>
  </image>
</images>
```

XML data file format for Microsites

Microsites are metadata sets that describe how a page structure may be. Microsites do not describe themselves how the page will look like, this is the purpose of XSL stylesheets. Microsites just define the possible page structure which will be later customixed by each PageSet definition.
Microsites, as PageSets, are stored as XML files.

A Microsite is formed by a set of one or more Containers. Each Container will become a Page when the Microsite is instantiated into a PageSet. Inside each Container there are Metablocks, the basic metainfo structure for composing the page structure.

These are the component sections for an XML Microsite:

1. **Header and character set:** some lines that, according to the XML standards describe the document type and character set used.
   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <?xml-stylesheet type="text/xsl"?>
   ```
2. **Document type identifier**.
3. **Template name:** string with a descriptive name of the template in use.
4. **Allowed fonts:** the tag ***fonts*** contains a list of all base typefaces allowed to be selected from the graphical inteface. Note that these are not necessarily all the fonts that the page use but just the generic foundation typography.
5. **Allowed color schemes:** as the tag *fonts*, the tag ***colors*** contains a list of colors predefined at the CSS stylesheet that can be selected whilst defining the document appearance from the end-user interface.
6. **Containes (pages) of information:** the tag ***containers*** may have 1 or more containers. For newsletters is just 1 page, for websites there are several pages.
7. **Container identifier:** each container has a global unique identifier ***guid***.
8. **Container name:** descriptive name of the page that Hill be generated when this container is instantiated.
9. **Name of XSL template:** name of the XSL template file to be used for this container.
10. **Set of allowed metablocks:** each page is composed of blocks. The structure of a block (tex+photo, text only, two photos, etc.) is described by a metablock. So the metablock describes the type of contents that

the final block has and the cardinality of each of the atomic pieces. The tag ***metablocks*** is the list of metablocks that can be used in this container for generating final blocks.

11. **Metablock structure**.
12. **Metablock name:**.
13. **Atomic objects incide the metablock:** se Atomic objects are text paragraphs and images. This tags stablishes which atomic objects appear in the metablock and how many times (cardinality). Each atomic object has its own name that can be easily recognized by the end-user when filling actual contents, such as: "ARTICLE-TITLE", "LEFT-PHOTOGRAPH", etc. The syntax is a list of comma separated values. Each object in the list is of the form `i:[1..9]` or `p:[1..9]` ***p*** is for paragraphs and ***i*** is for images. 1..n is the cardinality of the object.
14. **Times that the object can appear:** The default value for ***maxoccurs*** is one.

Structure of simple data XML files

When a new document is created it is initially filled with some default texts that are pre-defined in a special kind of XML data files.

Structure of XSL stylesheets

The presentation of each page is determined by an XSL template. XSL templates have the following parts:

1. **Headers and carácter set:** it is a set of lines according to Standard XML that contain document types, namespaces and character set.
2. **Parameters taken from caller application:** in this section the caller application passes some parameters always required: domain, workarea, path to image server, current pageset and path to file Server. These parameters will be usefull for aplying filtres, compose paths to files and recover contents from attributes.
3. **Output template:** this section is the actual output template, usually XHTML.
4. **CSS Styles:** each template has a different CSS file for each typography and color scheme. The CSS file to be used is referenced from grnerated XHTML.
5. **Zone markup for detection from the webbuilder module:** the XSL template may be applied in two modes: edition mode or final mode. When using edition mode, it is posible to signal zones where new blocks will be inserted. For signaling a zone a Para marcar una zona

DIV or *SPAN* tag must be employed; the ***id*** attribute of the tag must be the name of the block to be inserted.

6. **Zone naming conventions:** zones must be named as the blocks that can be found at the XML metadata file. In case of have the same block repeated several times the name will be repeated with a hyphen and a number indicating the position (B-01, B-02, B-03, etc.).

Structure of CSS styles files

Each template has one or more CSS files, one for each typography and color scheme. These mens that if a template allows four color schemes and three font types, there will be twelve CSS files.

File naming conventions

Archivo	Nomenclatura
Metadata	***Container_Name***.xml
Data pre-loading (for examples)	***Container_Name*** _datatemplate.xml
Presentation template	***Container_Name***.xsl
Styles	***Typography_Name***.css
Thumbnail	***Microsite_Name***.gif

3 Directory structure

- **XML data files:** these are the files generated by the webbuilder module. They are stored at storage/domains/*domain*/workareas/*workarea*/apps/*app*/data
- **XML metadata files:** metadata files archivos for all templated are located at storage/xslt/templates
- **XML files for pre-loading simple data:** habrá there is one per container (page) of each template and they are stored at storage/xslt/templates.
- **XSL stylesheets presentation files:** there is one per container stored at storage/xslt/templates
- **CSS files:** there is one for each typography and solor scheme pair. They are stored at under hipergate webapp root directory /styles/*template_name*/*color*.

- **GIF/JPEG images used by templates:** there may be several sets of images for a template. Images depend on the color scheme. They are stored at styles/*template_name*/*color*
- **Thumbnails de plantilla GIF/JPEG:** each template may have a small thumbnail image displayed at the application front-end.These images are stored at /styles/thumbnails and must have GIF extensión and the same name as teh container to which they are associated.

4 Data structures at the DBMS

hipergate has several tables containing the information about template sets for content production.

The table **k_microsites** contains one register for each template with the following fields:

- **gu_microsite:** template global unique identifier.
- **nm_microsite:** template name.
- **path_metadata:** relative path from storage root to the directory where the XML metadata file is located.
- **id_app:** numeric identifier of the application making use of the template (newsletter or website). This identifiers are listed at **k_apps**.
- **dt_created:** template creation date.
- **tp_microsite:** template type.
- **gu_workarea:** this columna is not used at the standard build, it is reserved for the creation of templates that are only reachable from a given WorkArea.

The table **k_pagesets** contains one register for each template instante (document) with the following fields:

- **gu_pageset:** document global unique identifier.
- **dt_created:** creation date.
- **dt_modified:** date when last modified.
- **nm_pageset:** Descriptive Name (by default is the name of the generated HTML page)
- **gu_workarea:** GUID of WorkArea to which the document belongs.
- **path_data:** relative path from storage root to the directory where the XML data file is located.
- **id_language:** document language.
- **gu_microsite:** GUID of microsite template of this document.
- **id_status**: Publishing status.

- **gu_company:** GUID of the company owner of the document.
- **gu_project:** GUID of the Project to which the document belongs.
- **tx_comments**: Comments.

The table **k_pageset_pages** containes one register (Page) for each document instance of a template (PageSet).

- **gu_page:** page global unique identifer.
- **pg_page:** page number (1, 2, .. n)
- **dt_created:** creation date.
- **dt_modified:** date when last modified.
- **tl_page:** page title.
- **gu_pageset:** GUID of PageSet where page is contained.
- **path_page:** relative path from storage root to the directory where the XML data file is located.

5 The template creation process

For creating a new template the following steps must be accoplished:

1. Define **chromatic variants** and different **typographies** for the template.
2. Define information structures allowed at the documents and their grouping in containers. This information is held at the **metadata XML file**.
3. Write the default texts preloaded when a new document is instantiated. This is done at the **data preloading XML files precarga**.
4. Create the **CSS stylesheets** for the different colors and typographies.
5. Create an **XSL rendering file** for each page.
6. Add **template images** GIF/JPEG.
7. Create a **thumbnail** for each **template image**.
8. Add the new template to the **database**.

Step 1: Chromatic variants and typographies

The first step when writting a new template is deciding which are going to be the different colours and typographies used by the documents.

Depending on the number of colours and typographies it will be necessary to create more or less CSS files and template images.
The definition of available chromatic variants and typographies is placed at the ***<colors>*** and ***<fonts>*** sections of the metadata XML file.

Here is an example of these sections:

```
<?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>Blue-orange</color>
    <color>Green-Yellow</color>
    <color>Black-Purple</color>
  </colors>
```

In this example there are three chromatic variants (*Blue-Orange, Green-Yellow* and *Black-Purple*) and three typographies (*Arial, Times* and *Verdana*). This will require the creation of three subdirectories:

- *webroot*/styles/Contemporary/Blue-Orange
- *webroot*/styles/Contemporary/Green-Yelow
- *webroot*/styles/Contemporary/Black-Purple

At these subdirectories the following ítems must be created:

- CSS stylesheets. One for each typographie (*Arial.css, Times.css* & *Verdana.css*)
- Template images (GIF/JPEG) The images of one directory correspond to a given chromatic variant.

Step 2: Creation of the XML metadata file

The next step is designing the contents structure for the new template. This task includes:

1. Number of **information containers** (pages) at the template. For newsletters templates there may be only one container.

Continuing with the previous example, the containers must be created at the XML metadata file.

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

The ***GUID*** assigned to a container must be globally unique.

GUIDs may be generated by calling method ***com.knowgate.misc.Gadgets.generateUUID()*** (see JavaDoc)

2. Each **container** must have its own name and an XSL stylesheet template for displaying it.

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

The ***name*** node is the container's name. For newsletters there is only one name, but in the case of a website there are several pages with different names. The ***template*** node is the name of the XSL stylesheet file used to display the information of the conatiner.

3. Each container has several **metablocks** in it. Metablocks are chunks of information inside the container. A metablock must have human readable identifier, a sequential number, a name, a maximum count of occurences of the metablock inside the container and a list of objects allowed to be inside the metablock (with object type, name and cardinality).

Here is and example of a metablock. The first one is for inserting an image that may occur at most once (like a page header). The second one is for inserting an image, a text paragraph and optionally another ten more paragraphs.

```
<metablocks>

	<metablock id="UPPER_LOGO" listingOrder="1">
		<name>Upper Logo</name>
		<template>#inline:.</template>
		<objects>i:UPPER_LOGOTYPE</objects>
		<maxoccurs>1</maxoccurs>
	</metablock>

	<metablock id="ARTICLE" listingOrder="2" allowHTML="false">
```

```
                <name>Article</name>
                <template>#inline:.</template>
                <objects>i:ARTICLE_IMAGE,
                       p:ARTICLE_TITLE,
                       p:ARTICLE_PARAGRAPH_[1..10],
                       p:ARTICLE_LINK_[0..1]
                </objects>
                <maxoccurs>10</maxoccurs>
            </metablock>

     </metablocks>

    </container>

   </containers>

</microsite>
```

The id attribute of each metablock must be a unique name.

The listingOrder attribute must be a unique ordinal value.

The allowHTML attribute sets whether the metablock will allow HTML tags or only plain text when editing its contents from the WYSIWYG editor.

The sintax for defining each object inside a metablock is as follows:

- **Object type**: it may be ***p*** or ***i*** depending on whether the object is a paragraph or an image.
- **Object name**: An uppercase name. If the object has a cardinality then a low hyphen must be place just before the square brackets.
- **Cardinality**: a range inside square brackets.

Step 3: Creation of pre-loaded data XML files

A pre-loaded data XML file contains sample texts that are placed by default inside every new document. A pre-loaded data XML file does not contain structural information but only texts to make data entry easier and more intuitive for the end-user.

It is the jobo f the contents writer to replace these default texts with the final ones of the desired dcocument.

A pre-loaded data XML file is composed of:

1. A set of pages (**PageSet**).

2. Inside the PageSet, each of its **Pages**.
3. Inside a page, several **zones**.
4. Inside a page, one or more **blocks**.
5. Inside a block, one or more **paragraphs** and/or **images**.

Following the previous lets, write the pre-loaded data for the *Contemporary* container, which will contain:

- A block with the logotype
- A block representing an article which will have a title, a text paragraph and an image.

```
<?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>Blue-Orange</color>
```

Once the Readers have been defined and the typography plus colours have been choosen, proceed to define the information blocks:

```
<pages>
 <page guid=":gu_pagex">
  <title><![CDATA[:page title]]></title>
  <container>:gu_container</container>
   <blocks>
    <block id="001">
      <metablock>UPER_LOGO</metablock>
      <tag>Upper Logotype</tag>
      <paragraphs>
       <paragraph id="REMOVABLE"></paragraph>
      </paragraphs>
      <images>
       <image id="REMOVABLE"></image>
       <image id="IMAGE_ARTICLE">
        <url></url>
        <path>http://img.test.com/style/com/img_es.gif</path>
        <alt><![CDATA[Alternative Text]]></alt>
        <width>48</width>
        <height>120</height>
       </image>
      </images>
      <zone/>
    </block>
```

The first block is based on the metablock UPPER_LOGO which allows a single image that appear at ***images*** section and has a link URL, an alternative text and its dimensions. At the beginning of every section an empty paragraph or image with identifier **REMOVABLE** is required.

```
    <block id="002">
      <metablock>ARTICLE</metablock>
```

```
            <tag>Article</tag>
            <paragraphs>
             <paragraph id="REMOVABLE"></paragraph>
             <paragraph id="ARTICLE_TITLE">
              <text><![CDATA[Title of the article]]></text>
              <url></url>
             </paragraph>
             <paragraph id="PARRAFO_ARTICULO_1">
              <text><![CDATA[Text of the article]]></text>
              <url></url>
             </paragraph>
            </paragraphs>
            <images>
             <image id="REMOVABLE"></image>
             <image id="UPPER_LOGOTYPE">
              <url>http://www.your-domain.com/</url>
              <path>http://img.test.com/style/com/def_es.gif</path>
              <alt><![CDATA[Alternative Text]]></alt>
              <width>468</width>
              <height>60</height>
             </image>
            </images>
            <zone/>
           </block>

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

This second block is based on the metablock ***ARTICLE***, defined at the metadata XML file and the syntax of its internal object is like the one defined at the metadata file.
Texts and image alternative words must be enclosed in CDATA tags.
The pre-loaded data may contain as many default blocks as the maxoccurs attribute of the metadata file allows.
The pre-loaded data file must be placed at the proper directory (see step 3) and following the syntax rules of step 2 .

Step 4: Creation of CSS stylesheets

CSS files are optional and are only necessary when the styles to be used are external to the XSL templates..

There must be one CSS file for each typo+colour pair. These files will be stored at ***…/styles/<template>/<colour>***. The directory corresponds to the color scheme and inside it there is one .css file for each diffent typo. At the example, there are three files for each color directory: Arial.css, Times.css and Verdana.css.

For referencing the styles file from the generated HTML the following tag must be included at the ***<HEAD>*** section.

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

The external CSS files are inlined inside each HTML MIME message just before sending it.

Step 5: Creation of XSL stylesheets

Each container defined at the metadata file must have an associated XSL presentation file. At this XSL file is where the template for HTML output is defined.

The design of the template is completely free upon the designer's desire.

The following eigth steps are recommended for creating an XSL template:

1. Write the **headers** of the XSL file
2. Read **parameters** passed to the XSL template from the application
3. **Start** the XSL template
4. Reference the **CSS** stylesheet
5. Put the **XHTML** code
6. **Access information** from the XML data file
    a. Text paragrphs
    b. Images
7. **Iterate through** repeatable blocks
8. **Close** XSL template

1. XSL headers

Every XSL files requires some headers with :

- The DTD for validating XSL tags.
- The namespace for XSLT instructions.
- The ouput method and carácter set.

The XSL file must start thus with the following lines:

```
<!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. Parameters passed from the application

The application passes to the XSL template the following parameters:

```
<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. Start of XSL templat

This lines makes XML paths relative to the root elemento f the XML file:

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

4. Reference to CSS stylesheet

The css stylesheet must be referenced at the ***head*** section of the ***XHTML*** document, as it is explained at step four of this guide.

5. Start of XHTML code

The template body is very similar to an XHTML file. It is important that all tags are well balanced <tag>…</tag> or etc.

6. Accessing XML contents

From the XHTML code of the template the <xsl> tags referente the contents put at the XML data file by the document editor.

Example of how to put the contents of a paragraph inside the XHTML:

```
   <xsl:if
         test="pageset/pages/page/blocks/
```

```
                block[metablock='WELCOME']/paragraphs/
                paragraph[@id=''WELCOME_TEXT']/text!=''">

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

</xsl:if>
```

- The first tag <***xsl:if***> checks that a block with name ***WELCOME*** exists at the contents XML file and that the block contains a paragraph with name ***WELCOME_TEXT***. The inner tag <***xsl:value-of***> retrieves the text from that paragraph.

- **Important:** set the attribute ***disable-output-escaping*** to "***yes***" for metablocks that allow HTML contents at their paragraphs (attribute allowHTML="true" at the metablock definition). Use ***disable-output-escaping="no"*** for plain text only paragraphs.

Accessing the information for an image:

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

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

- In this case a the variable ***upperLogo*** is set to the path of the image node at the XML date file; an image with id ***ULOGO*** exists at a block with metatype ***UPER_LOGO***.

7. Iterating through repeatable information

The ***xsl:for-each*** instruction allow to iterate through repeatable information blocks of the same metatype.

The next example shows how to retriever several blocks of the same metatype and then read the paragraph which name starts with ***PARAGRAPH_ARTICLE***

```
<xsl:for-each
         select="pageset/pages/page/blocks/
```

```
            block[metablock='ARTICLE']">

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

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

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

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

            <br/>

      </xsl:for-each>

</xsl:for-each>
```

- First an iterator ***xsl:for-each*** walks through all blocks of metatype ***ARTICLE.***
- Then another nested ***xsl:for-each*** iterator walks through the paragraphs and select those which name starts with *PARAGRPH_ARTICLE.*
- The list is sorted by id and then the paragraph contents are griten to the output..

8. Closing the template sheet

The last lines of the template sheet are:

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

For more information about XSL see: http://www.w3.org/Style/XSL/.

Step 6: Designing images for the template and thumbnails

All the images of a template must be stored at the template directory. See Chapter 3 of this guide for more information about image storage.

Step 7: Putting the template at the database

One the template is finished it must be referenced from the ***k_microsites*** table of the database, with the following values :

- **gu_microsite**: must be the 32 characters GUID that is placed at the ***guid*** attribute of the ***<microsite>*** node of the metadata file.

- **nm_microsite**: must be the same value as the *name* node of the metadatafile
- **path_metadata**: a relative path to the metadata XML file. For example, if the filename is ***template1.xml*** the path_metadata should be ***xslt/templates/template1.xml***
- **id_app**: insert ***13*** for newsletters and ***14*** for websites.

7 Example

This section contains a step by step example of how to create a new template by modifitying te **Basic** template.

Files to start with:

- Original XHTML page:

  <webroot>/storage/xslt/z_templates_newsletters/

- GIF/JPEG Images:

  <webroot>/storage/xslt/z_templates_newsletters/basic_archives/

- CSS stylesheet:

  <webroot>/storage/xslt/z_templates_newsletters/basic_archives/

Files produced:

- New CSS stylesheets:

  <webroot>/web/styles/basic/Black-n-White/

- Thumbnail:

  <imgroot>/styles/thumbnails/

- Imágenes finales:

  <imgroot>/styles/basic/ Black-n-White /

- Metadata files:

  <webroot>/storage/xslt/templates/

- Default data:

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

- XSL template:

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

So in this example we have several files: an XHTML web page, a CSS stylesheet, and some images shown at the web page. And with this items the goal is to create a new template.

Step 1: Color schemes and typesets

For Simplicity we shall have only one color scheme called Black-n_White. And three typesets (***Geneva, Times and Verdana***). At the ***style.css*** file for ***Basic*** template there are four styles:

- ***Htmlbody***: body background style.
- ***Title***: <H1> Titles.
- ***Subtitle***: <H2> Subtitles.
- ***Plaintext***: Body text.

Step 2: Metadata file

When designing the metadata file the source XHTML file must be examined and the information structures extracted from it: paragraphs and images which will be grouped into blocks of different metablock types.

The source file for the Basic template has the following blocks:

- Main Logo
- Newsletter Title
- Newsletter Subtitle
- Summary header
- Summary text
- Salutation
- Welcome text
- Articles composed of a title and a text body

The information can then be organizad in this way:

- A block with a single image for the upper logo
- A block with two paragraphs (date and text) for mini header
- A block with one to nine paragraphs for the summery
- A block with four paragraphs (title, summary, body, link) for the articles.

The file ***Basic.xml*** at ***<webroot>/storage/xslt/templates/*** contains the definitions for these types of blocks.

Step 3: Default Data

The default data file contains some sample data that will be placed at each new instante from a template document.

The file ***Basic.datatemplate.xml*** at ***<webroot>/storage/xslt/templates/*** contains these default blocks.

Step 4: CSS Stylesheets

The CSS stylesheets must be put at directory ***<webroot>/web/styles/basic/Black-n-White/*** In this example there will be three files copied from Basic Template: ***Geneva.css***, ***Times.css*** y ***Verdana.css***.

Step 5: XSL Template

These are the most important parts of Basic.xsl template:

1. **Read input parameters (lines 5-9)**
   These five lines are parameters passed by the application (***domain***, ***workarea***, ***image Server base URL***, ***document unique identifier*** and ***file server base path***).

2. **CSS stylesheet import (line 21)**
   The parameter param_imageserver is used at the relative path to the CSS files ***pageset/color*** and ***pageset/font***.

3. **References to images (line 29)**
   At line 29 there is an example of how to put an image inside the template. For composing the SRC="…" attribute the parameter ***param_imageserver*** and style information ***pageset/color*** are used.

4. **Putting the logotype (lines 51-58)**
   Lines 51 to 58 contain the instructions for retrieving the logotype. Note that at line 51 it is where a highlightning region is defined with name ***LOGOSUPERIOR_1***, this highlight feature is optional but makes easier to located which part of the template is being edited from the WYSIWYG editor. After the highlight span there are two conditional execution instructions which make the template behave slightly different depending on whether the image has explicitly defined dimensions or not.

5. **Retrieving information for mini header (lines 61-67)**
   At line 61 there is another highlightning region. At line 64 a variable holds the XPath to block ***MINICABECERA***. Lines 65 and 66 make use of that variable for retrieving paragraphs.

6. **Retrieving information for summary (lines 75-80)**
   These lines compose the summary of the articles from the newsletter. There is an iterator that walks through the paragraphs from blocks of metatype ***ARTICULO*** that start with ***TITULO-ARTICULO***. The paragraphs are sorted by their ***id*** attribute.

7. **Retrieving information for welcome (lines 85-88)**
   Another highlighted zone is declared and an XML path is stored into a variable pointing to the block ***BIENVENIDA***. Line 86 retrieves the information of the block which ***id*** is ***BIENVENIDA-TEXTO***.

8. **Retrieving information for articles (lines 119-124)**
   An iterator is declared for walking through the blocks of metatype ***ARTICULO***. At each loop pass the subtemplate ARTICULO (defined between lines 166 and 203) is invoked passing the block ***id*** to it as a parameter.

9. **Retrieving information for links (lines 126-139)**
   The information for blocks of metatype ***LINKS*** is retrieved much in the same way as was the information for SUMARIO.

10. **Definition of the article subtemplate (lines 166-203)**
    The block ***id*** is get at line 168 by the ARTICULO subtemplate. The paragraph of the block with the given ***id*** is stored at a variable named ***TITULO***. Line 175 declares the highlightning zone for the article. Line 178 prints the information from the ***text*** node of the title paragraph to the sheet output. At line 190 an iterator is declared which walks through all the paragraphs of the article which id starts with

PARRAFO. At line 192 the information of the ***text*** node is wriiten to the sheet ouput. At line 194 another iterator is declared for printing the paragraphs of the article which id starts with ***ENLACE***.

Step 6: Template images and thumbnail

Any image referenced directly from the XSL stylesheet or from and <image> tag of the default data must be placed at a URL reachable from the application.
By default, the images of this example are put into ***<imgroot>/styles/basic/Black-n_White/*** directory.

The template thumbnails must be .GIF files approximatedly of 242 by 282 pixels and must be put at ***<imgroot>/styles/thumbnails/*** with name of the template, in this case "Basic.gif".

Step 7: Putting a referente to the template at the database

Each template is referenced by a row at table ***k_microsites*** with the following columns:

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

The values for gu_microsite,nm_microsote and path_metadata must match ***microsite guid*** , ***name*** and ***template*** of file ***Basic.xml***.

8 References about XSL technologies

Tutorials

World Wide Web Consortium	**The XSL family**	www.w3.org/Style/XSL
DevGuru	**XSLT Tutorial**	www.devguru.com/Technologies/xslt/quickref/xslt_intro.html
W3Schools	**Tutorials**	www.w3schools.com

Tools

Altova	**XMLSpy XML/XSLT Editor**	www.xmlspy.com
Dave Raggett	**Tidy**	tidy.sourceforge.net

Books

O'Reilly/Doug Tidwell	**XSLT**	www.oreilly.com/catalog/xslt/
Michael Kay	**XSLT : Programmer's Reference**	www.amazon.com
Bob Ducharme	**XSLT Quickly**	www.manning.com/ducharme/
Khun Yee Fung	**XSLT: Working with XML and HTML**	www.amazon.com