The Construction and Administration of a Dynamic Web Site.

Jean Xech

Centre d'Initiatives
pour les technologies d'information, de communication et d'éducation de l' Université de Perpignan

1 The context

The University of Perpignan is a pluridisciplinary university comprising three teaching and research units (Faculties) and a University Technological Institute made up of five departments. It is physically located in six different sites : Perpignan, Narbonne, Carcassonne, Mende, Mont-Louis et Tautavel.

At the end of 1999 the University confirmed its interest in the new technologies by setting up a new service (the Centre for Initiatives in Information, Communication and Education Technologies : CITICE) specialising in the development of the new information, communication and education technologies. The service's missions relative to these technologies concern technology watch, training and site maintenance, and within this framework it is setting up a pedagogical intranet intended to make digital pedagogical resources available to students in addition to promoting the University's teaching and scientific activities (maintenance of the University web site). This task is conducted within a service with a staff of four under the direction of Professor Robert Marty.

2 The problem

The creation and maintenance of a web site inevitably raises problems of functional and physical organisation. It is important to define the headings offering navigational entry points. Such a structure requires a mode of physical organisation for page storage (flat files or a navigation based tree structure). Once a particular structure has been adopted it is then necessary to define an operational mode for updating the pages, in particular when they are supplied by diverse organisations or productions sources as is the case at the University , whose organisational structure is complex and constantly evolving. In this case, site maintenance is confronted by two contradictory constraints of centralisation and decentralisation : the centralisation necessary to the coherence of the site and the decentralisation required by a functional organisation of content production.

We turn now to the problems relating to the contents of a web site; these can be classified in terms of presentation and life expectancy. Among the documents making up a site are to be found the various introductory pages (concerning the activities of the university, the laboratory, the administrative services, personal web pages, information concerning collaborations), output and results (publications, pre-publications, memos, syllabuses and programmes, information disseminated by distribution lists) and events (internal seminars, lectures, news items, etc.). To the temporal categorisation of such documents (constraints on which need updating, which have a short life expectancy and which need to be stocked permanently) must be added a spatial categorisation : intra- or extra-net when the site has to serve as a platform for the exchange and sharing of documents between different partners.

A certain number of participants are involved in the « life » of a web site, and these can be classed according to two functions which often overlap: the content authors and the webmasters in charge of maintaining the site. The task of the former is often thankless, particularly when they have to « convert » documents in heterogeneous formats (Word, TeX/Latex, etc.) into a web publishing format (HTML), and then add them to the site (with index page updates or document referencing menus). Sometimes the technical barrier of producing web pages is a real problem for the content authors, while for those who have the expertise, development time is often lacking, let alone communication skills. Publication in such cases will often be entrusted to a third party, who thus becomes the site's editor in chief (the webmaster) and assumes responsibility for maintenance. In addition to such editorial functions, this person will automatically be responsible for page presentation (s/he determines the overall design of the site) and become a web editor (overseeing the coherence of the whole before putting it on-line on the server).

On further participant in the process needs to be taken into account, namely the web visitor, with his interpretative habits and his need of information, and this will require an adaptable and personal presentation of page content.

Thus a site's principal maintenance difficulty will be the grouping of all the documents to be published according to their category, updating them and, in many cases, formatting them before including them in the site. Such a manual task is almost beyond the ability of a single individual, and often involves the verification or rewriting of links. The production of concept content and presentation and their subsequent placing on-line become one and the same operation for the content producer using an HTML editor.

3 Our approach

In our approach, we have distinguished between the function of author and that of editor. Authors produce documents according to predefined frameworks and do not deal with storage or presentation. The editor may work on documents before they are published. Documents are stored and then assembled before being presented in HTML format. To simplify maintenance, the logics involved in data extraction are not centralised.

3.1 Selecting development norms

For development tasks, we decided to use wherever possible the set of new XML [1], XSL [2], CSS [3] and RDF norms. [4] XML enables us to define document structure with structural mark-up which focuses information content independently of presentation. XSLT (eXtensible Stylesheet Language: Transformations) is the language enabling the transformation of one XML document into another XML, the result being that the information presented can be selected and reorganised according to such or such a format. CCS (Cascading Stylesheets) stylesheets make sophisticated formatting possible, and are now well supported by the two commercial navigators (Explorer et Navigator). We selected a subset of metadata defined by the « Dublin Core » group [5] and determined by work on concept definition within the Ressource Description Framework (RDF). It was with this set of concepts defined by the World Wide Web Consortium (W3C) that we designed our application.

3.2 Application design

Within our approach we have tried to categorise very simply both documents and contributing agents with an aim to simplifying as much as possible creation procedures and content updating. To this end we organised the structure of the site as assembled pieces: the navigation bar, content storage and page assembly and presentation. Moreover, we distinguished the functions of author /editor, presentation editor and integrator.

As can be seen in figure 1, the application's overall design is very simple. It is based on :

- A diagrammatic description of the site [6] in the form of an XML document. It is within this document that the global information is to be found (logo or the site's URL) together with a description of the navigation elements according to their categories. This scheme is to be stored in a special (protected) directory.
- Page content is to be stored according to category in directories or in databases
- Data assembly scripts enable the production of XML documents which can be translated into HTM by the XSLT sheets.
- A set of sheets attached to the navigation and presentation elements CCS stylesheets together with the XSLT transformation sheets.

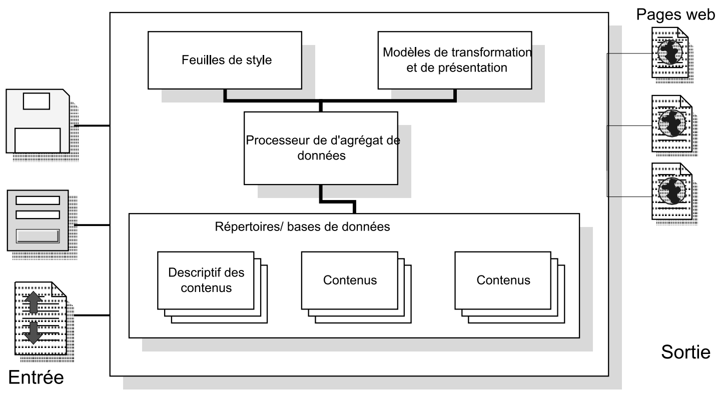
Pages web
Feuilles de style
Modèles de transformation et de présentation
Processeur de d'agrégat de données
Répertoires/ bases de données
Descriptif des contenus
Contenus
Contenus
Sortie
Entrée

Figure 1. Functional architecture

In the case of each of these items a simple solution was sought, using set standards and minimising development time (i.e. in terms of lines of code). Creating the navigation bar is the first task, and it is around this item that the whole site is organised. Content production was separated (figure 2) from the processes involved in presentation and on-line production. These are operations which are completely merged for the content producer when he uses an HTML editor. Modern XML/XSL concepts afford site designer the means to separate these functions thereby lightening the burden of the content creator (he no longer has to worry about presentation) and centralising the presentation (which becomes homogeneous and easily modified).

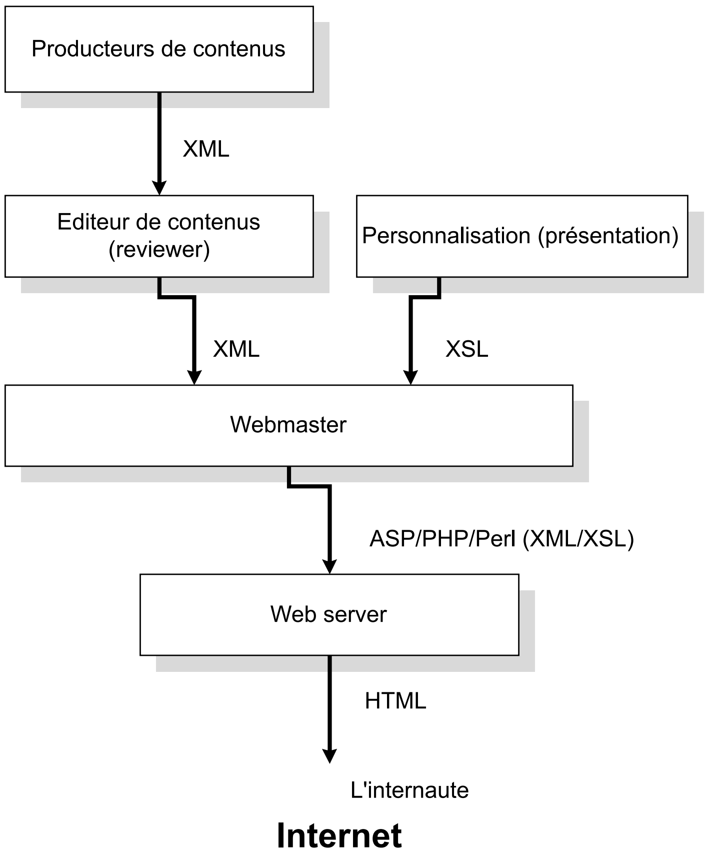

Figure 2. Functional production organisation

There was no need to define constraints concerning the platform system and web since the technologies employed are polymorphous. However, the decision was taken to produce the whole system on the server. Creating directly on the server makes it possible to ensure that the software functions correctly, particularly the XML and XSL parsers. It was nevertheless necessary to allow for levels of interpretation of the stylesheets by the navigators.

The site's initial page (the default page) is displayed with the navigation bar. As mentioned above in the presentation of the model, each navigation element refers to a procedure (executable server side : Perl, ASP or PHP) the task of which being to collect data and render it in XML format, then to apply the associated XSTL transformation before sending the HTML scripted page to the navigator.

The overall site scheme description is not rigid, and can be enriched over time provided the initial scheme is not revised.

3.3 The basic scheme

The overall scheme is stored in its own directory with access restricted tot he site administrator. It contains the following information.

- Overall specifications : the site title, the domain (in its canonical Internet form), the logo (its URL), the default CSS stylesheet, the default XSTL transformation sheet, and a default XML source file.
- The navigation elements defined by section, each referring to an element on the navigation bar ; each section is to be defined by : a name (which appears in the navigation bar), a subject, a type, an image (defined by its URL) and the source directory.

The scheme is described in XML.

3.4 The navigation bar

In our approach we opted for a very simple solution. The navigation bar is displayed in the form of a table composed of links with an effect obtained by the stylesheet when the mouse is placed over it. The code for this table is written in the form of a reusable procedure.

It is perfectly possible to write reusable code enabling the same XML scheme (the navigation <sections> in the overall scheme) to create, for example, an unfolding DHTML menu by writing the appropriate procedure.

3.5 Contents

Inn our solution, all content is organised according to a data model comprising two principal parts : the first is composed of the metadata in accordance with the RDF-DublinCore recommendation (title, author, subject (keywords), date, type, spatial cover (intranet/extranet), time cover (which can be an expiry date) and the data format), while the second is the document body.

Two roles were defined : author and editor.

- Author. Content creation takes place in two stages: content authoring followed by its transformation into an XML scheme. One way of rationalising these two stages is to provide authors with a tool enabling them to write or conceive content directly in XML. We produced a set of forms associated with processing procedures making it possible to edit XML documents directly. The content The content creator can thus capture information directly, copy/paste it in a text format without worrying about page appearance, or in HTML (with protected formatting). These forms are data specific and make it possible to use metadata to record the document automatically. The possibility of using an XML text editor (coupled to a scheme or a DTD) is currently being investigated. The author should not have to bother with document storage and archiving.

– The Editor: an editor function was created. This enables the site administrator to intervene directly on the updating of the editorial content of the site. A customisable procedure enables the automatic selection of all documents according to a set of criteria (type, date, etc.) and to mark them as « publishable. » By default, all documents are publishable.

3.6 Document storage

There are two ways of storing documents : as files or in a database.

– Storing documents in a directory : the advantage of this form of storage is in the fact that the data are stored in fragments (as files) which are easily modified, and whose structure requires no explicit definition (the XML document simply requiring to be well formed). The inconvenience of this solution is that the data that often need to be grouped for presentation procedures are split up. This was the solution that was adopted initially. We now need to set up performance measures, and, if assembly time becomes too critical, adopt the database solution.
– Database storage : the advantages of storing documents in a database are self-evident, particularly with respect to the integrity of the data involved. There is a bonus too in that the choice of a database server enables the use of stored SQL procedures and the recovery of the extracted data in a predefined XML scheme. For the extraction of « news » in our project we opted for an SQL server.

3.7 The assembly scripts

We chose not to centralise the data extraction logic. In such a solution we have an extraction/assembly script for each navigation section; this solution has the added advantage of simplifying the writing of the extraction/assembly code, and of adapting it to the documents involved, all the more so as the presentation logic was not included in the code. The scripts produce a well formed XML document which is subsequently transformed by XSLT into an HTML document. This involved choosing a script technology that included a parser enabling the loading and manipulation of the XML documents. The technology also had to interface seamlessly with an XSLT « parser .» At the present time the range of choices is limited, and for the development and installation stages we opted for the ASP/SQL Server (© Microsoft) [8] solution installed on an IIS server (Internet Information Server). Using XSLT enabled us to obtain reusable extraction/assembly codes since the code specific to presentation was assigned to the XSLT transformation stylesheet. These scripts are run on the server

3.8 The style and transformations sheets

The set of stylesheets are legible documents and were mostly written with a simple word processor.

The stylesheet makes it possible to define the background colour, title, paragraph and list styles throughout the site. This also where link colour and the navigation bar (the background colour of the table forming the navigation bar, the colour of the selected cell and the links) are defined. Various complementary styles have been defined, principally <div> boxes enabling control over page format.

The specification of the XML language is divided into three parts : XPATH [9] (a language for addressing parts of an XML document), XSLT [10] (a transformation language) (langage de transformation) and XSLF [11] (a formatting language). The transformation stylesheets we wrote employ the XPAZTH and XSLT specifications. XPATH models an XML document as a node tree, and, like an SQL request makes it possible to select only part of it. XSLT sets up the transformations by associating patterns with templates and applying them to the elements of a source tree to obtain a result tree. The major difficulty involved in writing XSL stylesheets concerns the expression of the location paths. To this end, we used a utility written in JavaScript to refine the XPATH requests. This procedure colours the nodes selected by the XPATH expression in the source tree, and has been used with Explorer 5.5.

4 Implementation

Several sites have been implemented by means of this model. The most compete form concerns the presentation of the research directory of the University of Perpignan [12].

A single standard code was written, and this consisted in the loading of the XML tree comprising the description of all the laboratories then of the XSLT stylesheet (namely the transformations to be applied) according to the navigation section, and, finally, in sending to the navigator the HTML page thus produced together with its CSS stylesheet. When all the control and error management lines have been removed, this code can be summarised as follows:

```
Set xmlsource  =  Server.CreateObject("MSXML2.DOMDocument")
set xslstyle = Server.CreateObject("MSXML2.DOMDocument")
xmlsource.async  =  false
xslstyle.async = false
xmlsource.load (Server.MapPath("/eunis/annu.xml"))
xslstyle.load(Server.MapPath("simpledir.xsl"))
Response.Write(xmlsource.transformNode(xslstyle.documentElement))
```

We provide an example transformation sheet (the selection and presentation in a table of the title of the laboratory, its director and its URL):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output indent="yes"/>
```

```
<xsl:template match="/labos">
<h1>Les Directeurs et leur E-mail (par ordre alphabétique)</h1>
<table border="0">
<tr><td>Laboratoire</td><td>Directeur (Emel)</td></tr>
<xsl:apply-templates>
<xsl:sort select="directeur"/>
</xsl:apply-templates>
</table>
</xsl:template>

<xsl:template match="labo">
<tr><td valign="top">
<xsl:value-of select="titre"/>
    </td>
  <td valign="top">
<a><xsl:attribute name="href">
    mailto:<xsl:value-of select="./adresse/emel"/></xsl:attribute>
   <xsl:value-of select="directeur"/></a>
  </td>
</tr>
</xsl:template>
</xsl:stylesheet>
```

5 Conclusion

We have tested the validity and robustness of the solution thus described. Initial implementations show that the use of a specialised editor makes the author's tasks and consolidates document capture. It now remains to develop search functions by means of the metadata already recorded using the new RDF standards of the W3C.

References

[1] XML: The Extensible Markup Language (XML) is the universal format for structured documents and data on the Web. Référence: http://www.w3c.org/XML/

[2] XSL: XSL is a language for expressing stylesheets. It consists of two parts: XSLT (http://www.w3.org/TR/xslt): a language for transforming XML documents and An XML vocabulary for specifying formatting semantics (XSL Formatting Objects). An XSL stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses the formatting vocabulary. Référence: http://www.w3.org/Style/XSL/

[3] CSS: Cascading Stylesheets (CSS) is a simple mechanism for adding style (e.g. fonts, colors, spacing) to Web documents. Référence: http://www.w3.org/Style/CSS/

[4] RDF: The Resource Description Framework (RDF) integrates a variety of web-based metadata activities including sitemaps, content ratings, stream channel definitions, search engine data collection (web crawling), digital library collections, and distributed authoring, using XML (http://www.w3.org/XML/) as an interchange syntax. Référence: http://www.w3.org/RDF/

[5] Dublin Core: The Dublin Core has become an important part of the emerging infrastructure of the Internet. Many communities are eager to adopt a common core of semantics for resource description, and the Dublin Core has attracted broad ranging international and interdisciplinary support for this purpose. Référence: http://www.purl.org/dc/about/index.htm

[6] RSS: RDF Site Summary (RSS) 1.0 (http://www.purl.org/rss/1.0/>) is a lightweight multipurpose extensible metada description and syndication format. RSS is an XML application, conforms to the W3C's RDF Specification and is extensible via XML-namespace and/or RDF based modularization.

[7] Editeurs de texte XML: MORPHON (http://www.morphon.com/), SPY (http://www.xmlspy.com/xml_editor.html), XMETAL (http://www.xmetal.com/top_frame.sq).

[8] Site XML de Microsoft: http://msdn.microsoft.com/xml/

[9] XPATH: XPath is a language for addressing parts of an XML document, designed to be used by both XSLT and XPointer. Référence: http://www.w3.org/TR/xpath

[10] XSLT: XSLT is designed for use as part of XSL, which is a stylesheet language for XML. In addition to XSLT, XSL includes an XML vocabulary for specifying formatting. XSL specifies the styling of an XML document by using XSLT to describe how the document is transformed into another XML document that uses the formatting vocabulary. XSLT is also designed to be used independently of XSL. However, XSLT is not intended as a completely general-purpose XML transformation language. Rather it is designed primarily for the kinds of transformations that are needed when XSLT is used as part of XSL. Référence: http://www.w3.org/TR/xslt11/

[11] XSLF: The indexes were extracted from the XSL Candidate Recommendation (http://www.w3c.org/TR/xsl). The reference will be upgraded when the standard is finalized. The reference uses examples from Apache FOP project (http://xml.apache.org/fop/). If you have your own examples we will be happy to add them to the reference. Référence: http://zvon.org/xxl/xslfoReference/Output/index.html

[12] Annuaire de la recherche de l'Université de Perpignan (http://www.univ-perp.fr/rch/)