UNIVERSITE DE MONS-HAINAUT
FACULTE DES SCIENCES
Section Informatique

Mémoire de 2ème licence :

CRÉATION DU SITE WEB D'ETRANSPARENCE AVEC DES TECHNOLOGIES LIBRES

par VINCENT BATTAGLIA

Directeur de mémoire :

JEF **W**IJSEN
Service de science des systèmes d'information (SSI)

Co-directeur de mémoire :

NICOLAS **S**ARIK
eTransparence SPRL

Année académique 2004-2005

Mémoire de 2ème licence :

CRÉATION DU SITE WEB D'ETRANSPARENCE AVEC DES TECHNOLOGIES LIBRES

par VINCENT BATTAGLIA

Remerciements

Je tiens à présenter mes remerciements les plus sincères au personnel d'eTransparence et plus particulièrement à Monsieur Nicolas Sarik pour sa disponibilité et sa patience. Un merci également adressé à Monsieur Wijsen pour ses conseils avisés. Je tiens aussi à remercier Benjamin et Nicolas pour leur aide précieuse. Pour terminer, je remercie Sandra et Roberto pour avoir lu et corrigé les fautes d'orthographes de ce document !

VINCENT **B**ATTAGLIA

Table des matières

TABLE DES MATIERES **4**

TABLE DES FIGURES **7**

CHAPITRE 1. INTRODUCTION **8**

1. PRESENTATION DU MEMOIRE 8
2. STRUCTURE DU MEMOIRE 8
   - *2.1. Chapitre 2* *8*
   - *2.2. Chapitre 3* *8*
   - *2.3. Chapitre 4* *8*
   - *2.4. Chapitre 5* *9*
   - *2.5. Chapitre 6* *9*
   - *2.6. Chapitre 7* *9*
   - *2.7. Chapitre 8* *9*
3. ACRONYMES ET ABREVIATIONS 9

CHAPITRE 2. TECHNOLOGIES UTILISEES **11**

1. PREREQUIS 11
2. TECHNOLOGIES ABANDONNEES 11
   - *2.1. Xindice* *11*
   - *2.2. eXist* *11*
   - *2.3. Struts* *11*
   - *2.4. Lenya* *12*
   - *2.5. Forrest* *12*
3. TECHNOLOGIES UTILISEES 12
   - *3.1. Tomcat* *12*
   - *3.2. RSS* *13*
   - *3.3. Wiki* *18*

CHAPITRE 3. APACHE COCOON **23**

1. DEFINITION 23
2. TECHNOLOGIES 23
3. CONCEPTS 23
   - *3.1. Formats multiples* *23*
   - *3.2. Séparation des tâches* *24*
   - *3.3. Les deux faces de Cocoon* *24*
4. HISTORIQUE 25
   - *4.1. Stefano Mazzocchi* *25*
   - *4.2. Cocoon 1* *25*
   - *4.3. Cocoon 2* *25*
5. INSTALLATION 26
6. ARCHITECTURE 26
   - *6.1. Sitemap* *26*
   - *6.2. Pipelines* *27*
   - *6.3. Les composants* *28*
   - *6.4. Récapitulatif de l'exécution d'une requête* *34*
   - *6.5. Autres concepts* *35*

CHAPITRE 4. COCOON COMPARE A PHP **39**

1. COMPARAISON 39
   - *1.1. Prix* *39*
   - *1.2. Documentation* *39*
   - *1.3. Hébergement* *40*
   - *1.4. Facilité d'installation* *41*
   - *1.5. Facilité d'utilisation* *42*
   - *1.6. Liaison avec les bases de données* *44*

1.7. Séparation des tâches ... *44*
1.8. Maintenance et réutilisabilité ... *46*
2. CONCLUSION ... 47

CHAPITRE 5. EXEMPLE RECAPITULATIF AVEC COCOON ... 48

1. MISE EN SITUATION ... 48
2. REPERTOIRES ET CANEVAS DE FICHIERS ... 49
3. AFFICHAGE D'UNE PAGE HTML ... 51
4. AFFICHAGE D'UNE PAGE PDF ... 57
5. AFFICHAGE D'UNE PAGE XML ... 59
6. INTERNATIONALISATION (I18N) ... 60
7. REPERTOIRE MUSICAL ... 63
8. STATISTIQUES ... 67

CHAPITRE 6. POLITIQUE ADOPTEE SUR LE SITE D'ETRANSPARENCE ... 71

1. MISE EN SITUATION ... 71
2. RESPECT DES NORMES ... 71
3. DESIGN "TABLELESS" ... 72
3.1. Introduction ... *72*
3.2. Positionnement CSS ... *73*
3.3. Problèmes des tableaux ... *74*
4. COMPATIBILITÉ ... 75
5. REFERENCEMENT ... 75

CHAPITRE 7. ANALYSE FONCTIONNELLE DU SITE D'ETRANSPARENCE ... 78

1. MISE EN SITUATION ... 78
2. TACHES A ACCOMPLIR ... 78
2.1. Partie commerciale ... *78*
2.2. Vente de matériel ... *79*
2.3. Partie scientifique ... *81*
2.4. Système de news ... *81*
2.5. Formulaire de contact ... *81*
2.6. Moteur de recherche interne ... *82*
2.7. Espace administrateur ... *83*
2.8. Espace client ... *83*
2.9. Version PDA ... *83*
2.10. Version WAP ... *84*

CHAPITRE 8. IMPLEMENTATION DU SITE D'ETRANSPARENCE ... 85

1. STRUCTURE GENERALE DU SITE ... 85
1.1. Arborescence de fichiers ... *85*
1.2. Structure d'une page ... *86*
1.3. Plan du site ... *91*
2. FONCTIONNALITES ... 94
2.1. Système de news (RSS) ... *94*
2.2. Zone téléchargements ... *97*
2.3. Moteur de recherche interne ... *98*
2.4. Vente de matériel ... *100*
2.5. Formulaire de contact ... *101*
2.6. Version WAP ... *103*
2.7. Version PDA ... *106*
3. TRAVAUX FUTURS ... 108
3.1. Espace administrateur ... *109*
3.2. Espace client ... *109*
3.3. Partie scientifique ... *109*
3.4. Et après? ... *112*

CHAPITRE 9. CONCLUSION GENERALE ... 113

BIBLIOGRAPHIE 114

1. LIVRES 114
2. ARTICLES 115
3. SITES INTERNET 117

Table des figures

FIGURE 1 - PRINCIPES DE SYNDICATION ET D'AGRÉGATION 13
FIGURE 2 - APERÇU DU SITE GOOGLE NEWS 14
FIGURE 3 - INDICATEUR DE FIL RSS 15
FIGURE 4 - DISTRIBUTION DES FORMATS DE SYNDICATION 16
FIGURE 5 - INTERFACE GRAPHIQUE DE FEEDREADER 17
FIGURE 6 - FONCTIONNEMENT D'UN WIKI 19
FIGURE 7 - OUTPUT DANS MOZILLA FIREFOX 21
FIGURE 8 - SÉPARATION DES TÂCHES DANS COCOON 24
FIGURE 9 - STEFANO MAZZOCCHI 25
FIGURE 10 - GÉNÉRATION D'UN RÉPERTOIRE DE FICHIERS 30
FIGURE 11 - EXÉCUTION D'UNE REQUÊTE AVEC COCOON 34
FIGURE 12 - DIAGRAMME DE SÉQUENCE DE L'EXÉCUTION D'UNE REQUÊTE 35
FIGURE 13 - TRAITEMENT D'UNE PAGE XSP 38
FIGURE 14 - UTILISATION DE PHP DANS LE MONDE AU 1ER FÉVRIER 2005 40
FIGURE 15 - INTERFACE GRAPHIQUE D'EASY-PHP 41
FIGURE 16 - COPIE D'ÉCRAN DE PHP DESIGNER 2005 43
FIGURE 17 - EXEMPLE DE SITEMAP AVEC POLLO 43
FIGURE 18 - APERÇU DE LA PAGE STATISTIQUE 49
FIGURE 19 - ARBORESCENCE DE FICHIERS DE LA MINI APPLICATION 49
FIGURE 20 - APERÇU DE LA PAGE "HISTORIQUE" 55
FIGURE 21 - APERÇU DE LA PAGE "HISTORIQUE" AVEC LE FORMATAGE CSS 56
FIGURE 22 - TRAITEMENT DE LA REQUÊTE "HISTORIQUE.HTML" 57
FIGURE 23 - LA PAGE "HISTORIQUE" AU FORMAT PDF 58
FIGURE 24 - TRAITEMENT DE LA REQUÊTE "HISTORIQUE.PDF" 59
FIGURE 25 - LA PAGE "HISTORIQUE" AU FORMAT XML 60
FIGURE 26 - LA PAGE D'ACCUEIL EN ITALIEN 62
FIGURE 27 - TRAITEMENT DE LA REQUÊTE "ACCUEIL-IT.HTML" 63
FIGURE 28 - APERÇU DE LA PAGE "MUSIQUE" 66
FIGURE 29 - CARRÉ VERT MUNI D'UN BORD ROUGE EN SVG 67
FIGURE 30 - APERÇU DES VISITES SUR LE SITE EN SVG 70
FIGURE 31 - BOUTONS DE VALIDATION XHTML ET CSS 71
FIGURE 32 - RÉSULTAT DU VALIDATEUR XHTML 71
FIGURE 33 - RÉSULTAT DU VALIDATEUR CSS 72
FIGURE 34 - APERÇU D'UNE PAGE DU SITE D'ETRANSPARENCE 79
FIGURE 35 - APERÇU DE LA PAGE "VENTE DE MATÉRIEL" 80
FIGURE 36 - APERÇU D'UN ARTICLE À VENDRE 80
FIGURE 37 - APERÇU DE LA PAGE "CONTACT" 82
FIGURE 38 - APERÇU DU MOTEUR DE RECHERCHE INTERNE 83
FIGURE 39 - ARBORESCENCE DE FICHIERS DU SITE D'ETRANSPARENCE 85
FIGURE 40 - COPIE D'ÉCRAN DU SITE D'ETRANSPARENCE 86
FIGURE 41 - VERSION IMPRIMABLE D'UNE PAGE 90
FIGURE 42 - VERSION PDF D'UNE PAGE 90
FIGURE 43 - CONSTRUCTION DES QUATRE FORMATS DE LA PAGE D'ACCUEIL 91
FIGURE 44 - PLAN DU SITE D'ETRANSPARENCE 93
FIGURE 45 - CONSTRUCTION DU PLAN DU SITE 93
FIGURE 46 - APERÇU DE LA PAGE "NEWS" 96
FIGURE 47 - FIL RSS SUR LE SITE D'ETRANSPARENCE 97
FIGURE 48 - ZONE TÉLÉCHARGEMENTS 98
FIGURE 49 - MOTEUR DE RECHERCHE INTERNE 98
FIGURE 50 - RÉSULTAT D'UNE RECHERCHE INTERNE 99
FIGURE 51 - APERÇU GRAPHIQUE D'UN FICHIER WML 103
FIGURE 52 - INTERCONNEXION DES CARTES DU SITE WAP 105
FIGURE 53 - APERÇU DU SITE WAP DANS UN NAVIGATEUR 105
FIGURE 54 - APERÇU DU SITE WAP DANS UN ÉMULATEUR 106
FIGURE 55 - APERÇU DU SITE DANS UN ÉMULATEUR PDA (AVANT) 107
FIGURE 56 - APERÇU DU SITE DANS UN ÉMULATEUR PDA (APRÈS) 108
FIGURE 57 - PAGE TYPE DU SITE WIKIPÉDIA 110

Chapitre 1. Introduction

1. Présentation du mémoire

Actuellement, Internet est sur la voie de la standardisation, contrastant avec l'anarchie qui y régnait en maître lors de sa percée au milieu des années 90. A cette époque, les langages comme le HTML et le CSS n'en étaient qu'à leurs balbutiements et chacun développait des applications Web à sa façon, donnant lieu à un grand désordre. Ce phénomène était d'autant plus répandu que les navigateurs de l'époque ne respectaient pas les normes et affichaient n'importe quel code, même erroné. Aujourd'hui, le HTML et le CSS sont des standards éprouvés et les navigateurs commencent petit à petit à se conformer à ceux-ci. Pour plus de dynamisme dans les pages Web, des langages comme PHP ou JSP sont très utilisés de nos jours mais ils commencent à montrer leurs limites. XML, de son côté, fait de plus en plus parler de lui. Ce langage permet d'envisager un futur prometteur pour le Web, de par les outils et les dialectes toujours plus nombreux qui en découlent.

Dans cette optique de renouveau, il m'a été demandé d'étudier les possibilités existantes dans le but de réaliser un site Web moderne avec des technologies XML et Open Source. Après quelques recherches, j'ai décidé d'étudier Apache Cocoon et de l'utiliser pour réaliser le nouveau site Web d'eTransparence, l'entreprise dans laquelle a été réalisé ce mémoire. Cocoon est un outil très puissant permettant de réaliser des applications Web avec l'aide de XML et XSLT en séparant le contenu, la logique et le style. D'autres technologies XML émergentes ont également été étudiées et utilisées dans le cadre de ce mémoire.

2. Structure du mémoire

2.1. Chapitre 2

Dans ce chapitre sont expliqués les différents outils et technologies qui ont été étudiés pour la réalisation du site Web d'eTransparence. D'une part, nous introduirons les outils qui ont été abandonnés, d'autre part, nous détaillerons ceux qui ont été utilisés pour la création du site d'eTransparence.

2.2. Chapitre 3

Ce chapitre traite d'Apache Cocoon, le framework que nous avons utilisé pour réaliser le site Web d'eTransparence. Tout d'abord, nous présentons l'outil et les concepts qui en découlent et ensuite nous étudions son fonctionnement interne dans les détails.

2.3. Chapitre 4

Ce chapitre consiste en une comparaison entre l'outil Cocoon et le langage dynamique PHP. La comparaison se fait sur les aspects qui ont été jugés les plus importants :

- Prix

- Documentation
- Hébergement
- Facilité d'installation
- Facilité d'utilisation
- Liaison avec les bases de données
- Séparation des tâches
- Maintenance et réutilisabilité

Après cette comparaison vient une conclusion nuancée décrivant dans quel cas Cocoon est plus approprié que PHP et inversement.

2.4. Chapitre 5

Ce chapitre montre un exemple récapitulatif de la réalisation d'une application Web avec Cocoon. Cette application possède des fonctionnalités bien spécifiques qui permettront d'avoir une meilleure idée sur la simplicité et les nombreuses possibilités de Cocoon.

2.5. Chapitre 6

Ce chapitre explique la politique que nous avons adoptée sur le site d'eTransparence afin d'obtenir un site d'un très haut niveau d'efficacité et d'accessibilité. Nous parlons ici du respect des normes, du référencement, de la conception interne des pages et de la compatibilité avec les différents navigateurs et les différentes tailles d'écran.

2.6. Chapitre 7

Ce chapitre présente une analyse fonctionnelle du site d'eTransparence. On se concentre donc uniquement sur le fonctionnement et l'aspect externe du site et non pas sur son implémentation (qui est abordée dans le chapitre 8).

2.7. Chapitre 8

Ce chapitre décrit comment a été réalisée l'implémentation avec Cocoon du site d'eTransparence et de toutes ses fonctionnalités.

3. Acronymes et abréviations

Cette section reprend la totalité des acronymes et abréviations qui sont utilisés dans ce mémoire.

Table 1.1. Acronymes et abréviations

API	Application Programming Interface
ASF	Apache Software Foundation
ASP	Active Server Pages
BCC	Blind Carbon Copy
BMP	Bitmap
CC	Carbon Copy
CMS	Content Managment System
CSS	Cascading StyleSheets

CVS	Concurrent Versions System
DNS	Domain Name Server
DOM	Document Object Model
DTD	Document Type Definition
EDI	Environnement de Développement Intégré
FAQ	Frequently Asked Questions
FTP	File Transfer Protocol
GIF	Graphics Interchange Format
GNU	GNU's Not Unix
GPL	General Public License
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
JSP	Java Server Pages
JPEG	Joint Photographic Expert Group
JVM	Java Virtual Machine
KDE	K Desktop Environment
LALR	Look-Ahead Left to Right Parser
LAMP	Linux Apache MySQL PHP
LDAP	Lightweight Directory Access Protocol
MathML	Mathematical Markup Language
MIME	Multipurpose Internet Mail Extensions
MVC	Model View Control
OASIS	Organization for the Advancement of Structured Information Standards
P2P	Peer To Peer
PC	Personal Computer
PDA	Personal Digital Assistant
PDF	Portable Document Format
PEAR	PHP Extension and Application Repository
PHP	Hypertext Preprocessor
PME	Petites et Moyennes Entreprises
PNG	Portable Network Graphics
RSS	Really Simple Syndication ou Rich Site Summary ou RDF Site Summary
SAX	Simple API for XML
SGBD	Système de Gestion de Bases de Données
SGBDR	Système de Gestion de Bases de Données Relationnelles
SGML	Standard Generalized Markup Language
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SVG	Scalable Vector Graphics
SWF	ShockWave Flash
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WBMP	Wireless Bitmap
WiFi	Wireless Fidelity
WML	Wireless Markup Language
WYSIWYG	What You See Is What You Get
WWW	World Wide Web
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XPath	XML Path Language
XQL (ou XQuery)	XML Query Language
XSL-FO	Extensible Stylesheet Language - Formatting Objects
XSLT	Extensible Stylesheet Language Transformations
XSP	Extensible Server Pages

Chapitre 2. Technologies utilisées

1. Prérequis

Nous considérons comme connus les langages Java, HTML, CSS, XML et ses dérivés (DTD, XML Schema, XHTML, XSLT, XSL-FO, XPath, XQuery, DOM, SAX, SOAP et SVG).

Si cette base n'est pas acquise, nous reportons le lecteur au mémoire de Fontaine [13], au livre d'Harold & Means [9], et au site W3Schools (http://www.w3schools.com).

2. Technologies abandonnées

Avant d'adopter définitivement Cocoon, nous avons fait quelques recherches qui nous ont mené à étudier d'autres outils, que nous avons finalement abandonnés, car ils ne correspondaient pas à nos besoins et à nos contraintes.

Pour chaque outil présenté ci-après, une brève description est donnée, ainsi que la raison pour laquelle il n'a pas été retenu.

2.1. Xindice

Xindice est une base de données XML native en Open Source. Elle est issue du projet XML Apache. Les requêtes sur Xindice se font avec le langage XPath.

Cet outil n'a pas été adopté car il ne permet pas de gérer une application complète avec le langage XML, seules les bases de données sont dans ce langage.

2.2. eXist

eXist est une base de données XML native, comme Xindice. La seule différence avec ce dernier est que les requêtes eXist doivent être formulées avec le langage XQuery.

Il n'a pas été retenu pour les mêmes raisons que Xindice.

2.3. Struts

Struts est un framework (voir **Chapitre 3, Section 1, “Définition”** pour une définition précise) pour le langage Java. Il permet de créer des applications Java qui s'exécutent côté serveur. Struts implémente le patron de conception MVC qui a pour but de séparer les tâches dans le développement d'applications logicielles (voir **Chapitre 3, Section 3.2, “Séparation des tâches”** pour une définition plus précise).

Nous n'avons pas utilisé Struts car il utilise essentiellement le langage Java alors que nous souhaitions construire notre application Web avec le langage XML. Il existe des modules

XML pour Struts [11] mais nous avons abandonné cette idée car nous avons découvert Cocoon peu de temps après.

2.4. Lenya

Lenya est un sous-projet de Cocoon. Il s'agit d'un CMS (Content Managment System) construit sur base de Cocoon. Un CMS permet de construire des applications Web sans aucune programmation. Tout est géré via une interface d'administration très élaborée.

Nous avons laissé cet outil de côté, d'une part car il est trop restrictif par rapport à nos besoins et d'autre part car il nécessite une connaissance très poussée de Cocoon avant de pouvoir comprendre son fonctionnement interne.

2.5. Forrest

Forrest est également un sous-projet de Cocoon. Il se défini comme un framework XML orienté documentation. Il simplifie quelque peu la création d'applications Web par rapport à Cocoon, entre autres en fournissant des modèles et des habillages prédéfinis. Son but avoué est de fournir un outil pour écrire facilement de la documentation logicielle. La plupart des sites de l'Apache Software Foundation sont d'ailleurs construits grâce à cet outil.

L'idée d'utiliser Forrest a été abandonnée pour les mêmes raisons que Lenya. De plus, Forrest ne s'adresse pas vraiment à nous, car notre objectif n'est pas de créer de la documentation logicielle.

3. Technologies utilisées

3.1. Tomcat

Tomcat est un serveur d'applications Java qui permet d'exécuter coté serveur des applications Java. Celles-ci peuvent être de deux types : des servlets ou des JSP.

Une servlet est une application Java qui s'exécute côté serveur. Le but premier d'une servlet est de fournir du contenu Web (en langage HTML essentiellement) dynamique sur un navigateur. On la distingue donc d'une applet Java, qui est une application qui s'exécute côté client (grâce à une machine virtuelle Java).

Une page serveur en JSP est une page Web en HTML à l'intérieur de laquelle on insère des instructions en Java qui sont exécutées par le serveur. Une instruction banale est l'interaction avec une base de données. Une fois les instructions exécutées, la page est renvoyée au client qui la lit avec son navigateur. On peut comparer le JSP avec le PHP et l'ASP, qui offrent les mêmes possibilités.

Tomcat est issu du projet Jakarta Apache. Il s'agit d'un projet ayant pour buts la création et la maintenance des solutions côté serveur pour la plate-forme Java. En plus de Tomcat, Jakarta Apache est à l'origine d'autres célèbres projets, dont Ant (outil de déploiement) et Struts (framework Java, voir **Chapitre 2, Section 2.3, “Struts”**).

Pour plus d'informations : http://jakarta.apache.org/tomcat/

3.2. RSS

3.2.1. Syndication de contenu

La syndication de contenu consiste à rendre visible sur un site α les dernières publications du site β grâce à un mécanisme automatique. En pratique, le contenu mis en commun par le site α est placé à l'intérieur d'un fichier dans un format standard ce qui permet au webmaster du site β de pouvoir importer ce contenu sur son site en utilisant un mécanisme automatique. Le fichier mis à disposition par un site est appelé un fil de news (*feed* en anglais).

Pour mieux comprendre ce concept, prenons un exemple concret. Imaginons que nous soyons des visiteurs quotidiens des sites de La Libre Belgique, du Soir, de la Dernière Heure, du Monde et du Figaro, car nous sommes très intéressés par l'actualité. Si nous tenons à lire les dernières dépêches de chacun de ces sites, cela risque de prendre un temps fou car nous devons parcourir les sites un à un.

Grâce à la syndication de contenu, nous sommes capables de rassembler (agréger) les dernières nouvelles de ces sites dans une même interface (Web ou autre). En effet, chacun de ces sites met à disposition un fichier dans un format standard (XML) reprenant les dernières dépêches du site. Avec l'aide d'un script ou d'un autre mécanisme équivalent, nous pouvons récupérer les informations distantes automatiquement dans ces fichiers et les afficher dans une même interface.

Pour mieux comprendre, voyons le schéma ci-dessous :

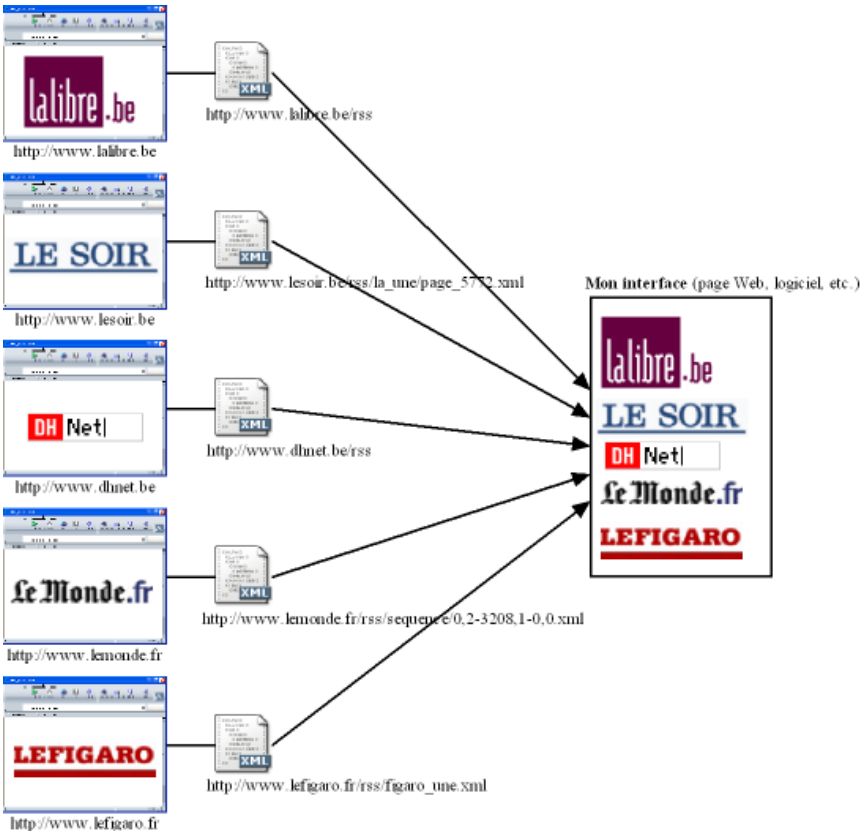

Figure 1 - Principes de syndication et d'agrégation

Les cinq sites possèdent chacun un fichier de syndication de contenu au format XML, proposant les dernières dépêches à la une. Notre interface - une page Web par exemple - qui connaît les URL de ces fichiers (enregistrées dans une base de données ou un fichier) peut aller récupérer les dépêches, grâce à un script ou à un autre mécanisme, dans les fichiers distants et les rassembler sous la même interface.

Le site Google Actualités (http://news.google.be) nous donne une idée de ce que peut donner cette mise en commun de différentes sources :

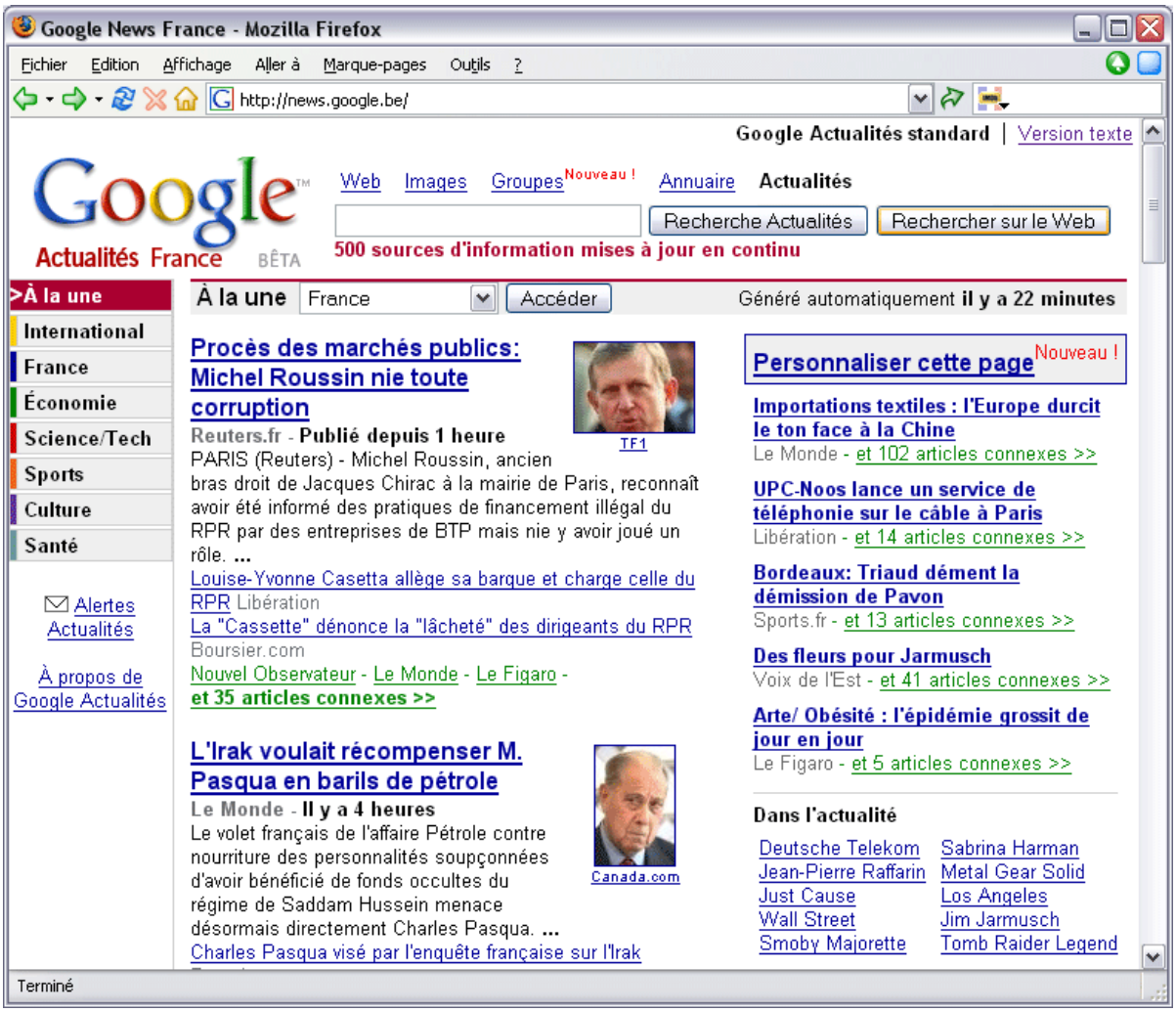

Figure 2 - Aperçu du site Google News

3.2.2. Définition

Le RSS est un format de syndication de contenu Internet fondé sur le standard XML. RSS est l'acronyme de RDF Site Summary, Rich Site Summary ou Really Simple Syndication (selon la version).

En RSS, le contenu se présente sous la forme d'un fichier au format XML respectant une DTD déterminée (disponible en annexe i). Il est assez simple de détecter un site proposant un fil RSS, il est affublé d'un rectangle orange siglé XML ou RSS.

Figure 3 - Indicateur de fil RSS

3.2.3. Aperçu d'un fichier RSS

L'exemple ci-dessous nous montre un exemple de fichier RSS. Il s'agit d'une partie du fil RSS provenant d'EuroNews.net, le site officiel de la célèbre chaîne d'information. Ce fil RSS est disponible sur http://www.euronews.net/rss/euronews_fr.xml.

```
<?xml version="1.0" ?>
<rss version="2.0">
 <channel>
  <title>EuroNews</title>
  <link>http://www.euronews.net</link>
  <description>
   The leading European News TV-channel. All the International news in seven
   languages.
  </description>
  <language>fr</language>

  <item>
   <title>
    Paris : les partisans de la Constitution europ&#233;enne en piste au Cirque
    d'Hiver
   </title>
   <link>
    http://www.euronews.net/create_html.php?page=detail_info&amp;lng=2&amp;option=0
   </link>
   <description>
    A 10 jours du r&#233;f&#233;rendum les saltimbanques ont laiss&#233; la place
    aux leaders europ&#233;ens de gauche et aux dirigeants de l'Union comme Josep
    Borrel.
   </description>
   <pubDate>Thu, 19 May 2005 07:11:02 +0200</pubDate>
  </item>

  <item>
   <title>
    L'Espagne adopte d&#233;finitivement la Constitution europ&#233;enne
   </title>
   <link>
    http://www.euronews.net/create_html.php?page=detail_info&amp;lng=2&amp;option=1
   </link>
   <description>
    Le S&#233;nat a tr&#232;s largement vot&#233; la ratification du trait&#233;,
    par 225 voix pour, 6 contre et 1 abstention.
   </description>
   <pubDate>Thu, 19 May 2005 07:11:02 +0200</pubDate>
  </item>

  <item>
   <!-- ... -->
  </item>

  <!-- ... -->

 </channel>
</rss>
```

Ce fichier est très simple à comprendre. Tout d'abord, il donne des informations sur le fil lui-même : son titre (`<title/>`), l'URL du site auquel ce fil RSS appartient (`<link/>`), une description (`<description/>`), la langue du fil (`<language/>`). Ensuite viennent les

actualités, chacune encadrée par la balise (`<item/>`). Chaque actualité est définie par un titre (`<title/>`), un lien (`<link/>`) qui pointe vers la dépêche sur le site lui-même, une description (`<description/>`) et une date de publication (`<pubDate/>`).

Connaissant l'URL et la syntaxe de ce fil RSS, il est très simple de l'importer et de l'afficher sur son site. En PHP, il faut utiliser un parseur XML comme Magpie RSS (http://magpierss.sourceforge.net). Si on travaille avec XML, une simple transformation du fichier XML distant avec une feuille de style XSLT peut faire l'affaire.

3.2.4. Formats de RSS

Malheureusement, un fichier RSS ne se présente pas toujours de la même manière. En effet, RSS n'est pas une norme et il en existe plusieurs formats. Le format du fil RSS d'EuroNews.net est le RSS 2.0. Les autres formats principaux de RSS sont le RSS 0.91 et le RSS 1.0. Il existe également d'autres formats non-RSS de syndication de contenu. Le plus répandu d'entre eux est Atom. Certaines versions sont plus utilisées que d'autres par les webmasters de sites d'actualités. Le schéma ci-dessous montre la répartition actuelle des formats de syndication de contenu sur Internet :

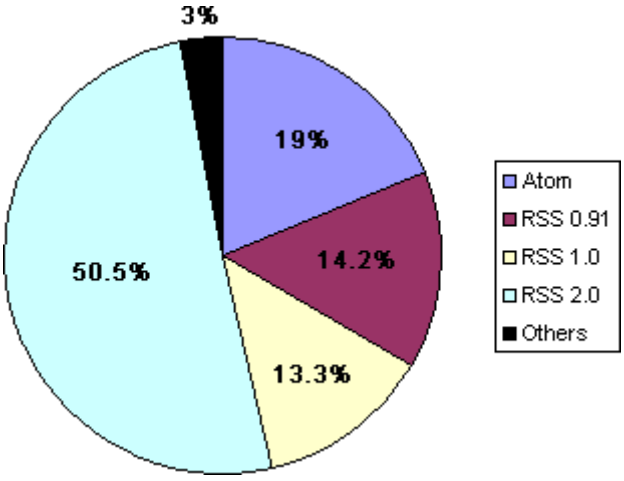

Figure 4 - Distribution des formats de syndication

En ce moment ont lieu des discussions entre les tenants du RSS 2.0 et ceux d'Atom pour une éventuelle fusion de ces deux formats, afin de voir émerger un standard unique pour la syndication de contenu sur Internet.

Pour plus d'informations sur l'histoire et sur les différents formats de RSS, le lecteur est invité à lire le document de Frédéric Laurent [29] à ce sujet.

Les spécifications :

- RSS 0.91 : http://my.netscape.com/publish/formats/rss-spec-0.91.html
- RSS 1.0 : http://web.resource.org/rss/1.0/spec
- RSS 2.0 : http://blogs.law.harvard.edu/tech/rss

3.2.5. Lecteurs RSS

Les lecteurs RSS, également appelés agrégateurs RSS, sont simples d'emploi et généralement gratuits. Ceux-ci servent à rassembler le contenu de plusieurs sites d'actualités en utilisant leurs fichiers RSS.

Des services en ligne tels que Bloglines (http://www.bloglines.com) ou RSS4you (http://www.rss4you.com) permettent de consulter à distance une liste de fils RSS et de les consulter de n'importe où.

Certains navigateurs ou gestionnaires de courriels ont directement intégré des lecteurs RSS. C'est le cas de la version 8 d'Opera ou de Mozilla Thunderbird.

Il existe également des logiciels dédiés à installer sur son ordinateur. Parmi les plus connus, citons Feeddemon (http://www.bradsoft.com/feeddemon), Feedreader (http://www.feedreader.com) et Sharpreader (http://www.sharpreader.net).

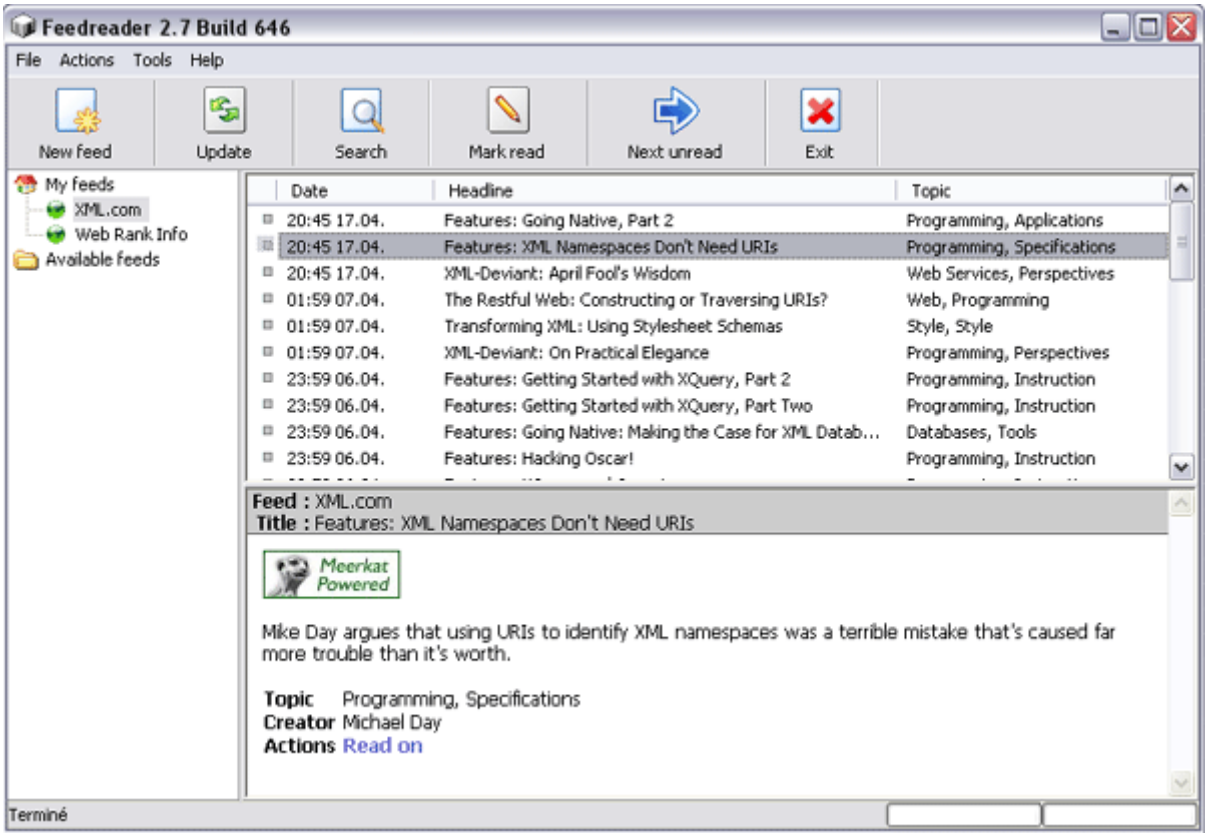

Figure 5 - Interface graphique de FeedReader

3.2.6. Validation RSS

De la même manière qu'il est possible de valider le HTML et le CSS (voir **Chapitre 6, Section 2, "Respect des normes"**), il est possible de valider un fil RSS ou Atom pour vérifier s'il est conforme à la syntaxe établie dans son DTD. Il en existe plusieurs sur Internet, Feed Validator (http://www.feedvalidator.org/) fait partie des plus performants. Il suffit d'entrer l'adresse du fil et le validateur se charge de détecter le format de celui-ci et de vérifier sa conformité.

3.2.7. Utilité?

Sur le site d'eTransparence, nous utiliserons la technologie RSS pour importer automatiquement les dépêches de sites distants directement dans l'interface de notre site. Par exemple, nous allons importer des fichiers RSS qui contiennent les dernières nouvelles sur le XML ou les dernières alertes aux virus. Cela permettra à l'administrateur du site de ne pas devoir écrire lui-même les news et fera donc gagner beaucoup de temps pour la maintenance.

Voir **Chapitre 8, Section 2.1, “Système de news (RSS)”**.

3.3. Wiki

3.3.1. Définition

Un Wiki est un site Web dynamique où tout visiteur peut modifier les pages à volonté. Il s'agit donc d'un outil collaboratif et communautaire très puissant, voué à remplacer les newsgroups et les FAQ à long terme.

Un Wiki est en réalité un site Web assez commun, la seule particularité est que chaque page du site est modifiable par n'importe quel visiteur, si celui-ci veut corriger une erreur ou s'il veut ajouter de l'information.

L'édition d'une page dans un Wiki se fait via un formulaire contenant le texte de la page. Ce texte se présente avec une syntaxe spéciale pour la mise en page, très simple et à la portée de tous. Aucune connaissance en HTML et, de manière plus générale, en programmation Web, n'est donc requise pour modifier une page.

L'actualisation d'une page après une modification se fait directement et sans aucune validation de la part d'un administrateur. Cela permet d'avoir un aperçu rapide de la page fraîchement modifiée et de remédier à cette modification si le résultat n'est pas celui escompté.

La question qu'on est en droit de se poser est : "Est-ce que le concept de Wiki est sûr?". En effet, on pourrait imaginer qu'un visiteur malveillant vienne effacer tout le contenu d'une page et y placer des insultes à la place ! Heureusement, chaque version d'une page est enregistrée et il est toujours possible de revenir à une version précédente, un peu comme le ferait un logiciel de gestion de versions (CVS).

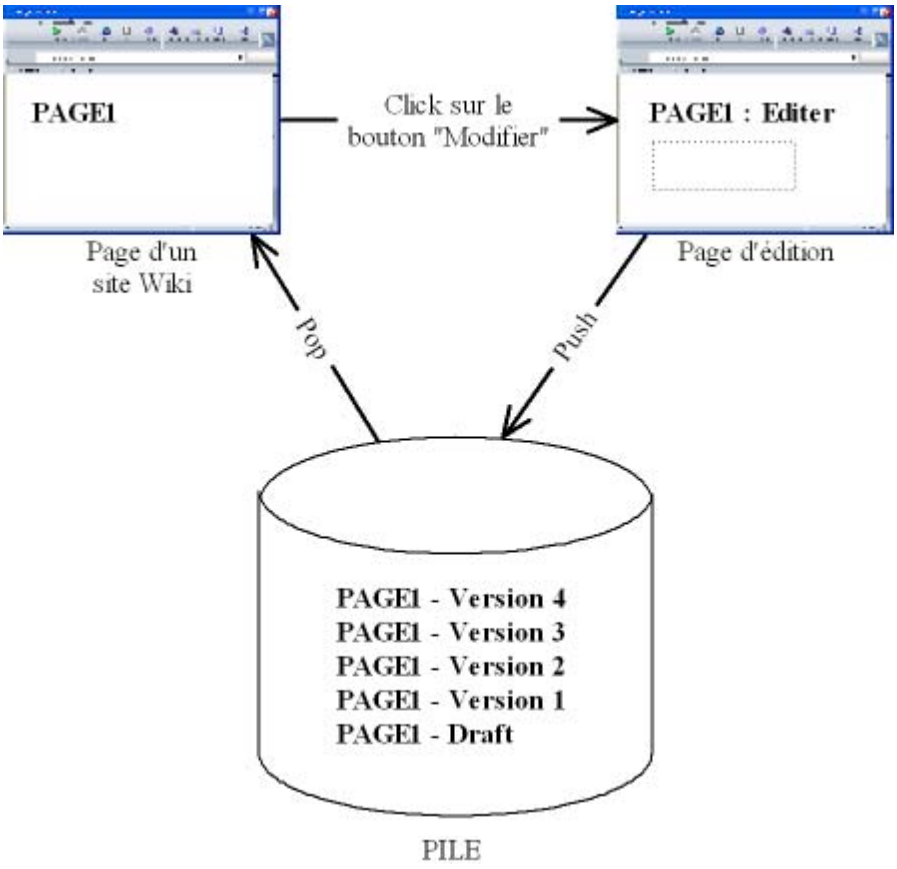

Figure 6 - Fonctionnement d'un Wiki

Le schéma ci-dessus montre le fonctionnement d'un Wiki. La page "PAGE1" comporte des informations qu'il est possible de modifier en cliquant sur un bouton. Cela nous envoie vers un formulaire contenant le contenu de "PAGE1". Après modification et soumission du formulaire, nos modifications sont enregistrées dans une structure de données que l'on pourrait comparer à une pile. La version la plus récente est placée au sommet de la pile et c'est cette version qui est affichée sur la page "PAGE1". Si l'administrateur remarque que la dernière version n'est pas conforme, il peut l'effacer et ce sera donc l'avant dernière version qui sera affichée.

Un exemple de Wiki est l'encyclopédie Wikipédia (http://fr.wikipedia.org) où n'importe quel visiteur est libre de créer ou de modifier des articles dans une gigantesque encyclopédie en ligne.

3.3.2. Syntaxe

Comme nous l'avons dit, les Wiki possèdent une syntaxe spéciale pour la rédaction du contenu. Il s'agit d'une syntaxe simple qui a l'avantage d'être facilement compréhensible et de produire du code HTML valide. Il existe en réalité plusieurs syntaxes Wiki, qui ont de nombreux points communs mais certaines différences. Toutefois, la syntaxe Wiki est actuellement sur le chemin de la standardisation.

Le tableau ci-dessous présente les rudiments de la syntaxe utilisée sur Wikipédia. Dans la première colonne nous avons la syntaxe Wiki, dans celle du milieu l'équivalent HTML de cette syntaxe Wiki et dans la dernière une brève description.

Table 2.1. Syntaxe Wikipédia

Wiki	Équivalent HTML	Description
`[[LienInterne]]`	`<a href="LienInterne">LienInterne</a>`	Deux crochets de part et d'autre d'un mot permettent de faire un lien interne au site sur lequel on se trouve.
`[[LienInterne|Ceci est un lien interne]]`	`<a href="LienInterne">Ceci est un lien interne</a>`	En ajoutant la barre verticale, on obtient le même lien mais avec une description du lien au choix.
`http://www.siteexterne.com`	`<a href="http://www.siteexterne.com" target="_blank"> http://www.siteexterne.com </a>`	Une URL dans le texte crée automatiquement un lien vers le site qui s'ouvrira dans une nouvelle fenêtre.
`[http://www.siteexterne.com Site externe]`	`<a href="http://www.siteexterne.com" target="_blank"> Site externe </a>`	Pour plus de lisibilité, on utilise un simple crochet, l'URL, un espace, le titre, et un crochet.
`'''texte en gras'''`	`<b>texte en gras</b>`	Le texte en gras est entouré de trois apostrophes de part et d'autre.
`''texte en italique''`	`<i>texte en italique</i>`	Le texte en italique est entouré de deux apostrophes de part et d'autre.
`* element 1 * element 2`	`<ul> <li>element 1</li> <li>element 2</li> </ul>`	Les étoiles permettent de faire des listes.
`# element 1 # element 2`	`<ol> <li>element 1</li> <li>element 2</li> </ol>`	Les carrés permettent de faire des listes numérotées.
`== nouvelle section ==`	`<h2>nouvelle section</h2>`	Deux signes égal de part et d'autre permettent de créer une nouvelle section.
`=== nouvelle sous-section ===`	`<h3>nouvelle section</h3>`	Trois signes égal de part et d'autre permettent de créer une nouvelle sous-section.
`==== nouveau paragraphe ====`	`<h4>nouveau paragraphe</h4>`	Quatre signes égal de part et d'autre permettent de créer un nouveau paragraphe.
`----`	`<hr/>`	Quatre tirets permettent la création d'une ligne horizontale.
`[Image:oiseau.gif]`	`<img src="oiseau.gif"/>`	Pour insérer une image, on utilise un simple crochet, le terme "Image:" et le nom de l'image.

3.3.3. Wiki vers HTML

Les sites de type Wiki possèdent un moteur interne puissant permettant de transformer la syntaxe Wiki en HTML pour l'affichage. Il existe quelques outils Open Source réalisant ce travail. Le plus utilisé par les Wiki actuellement est Wiki2xhtml (http://www.neokraft.net/sottises/wiki2xhtml/).

3.3.4. Exemple

Voici un exemple de texte écrit avec la syntaxe Wikipédia (voir **Chapitre 2, Section 3.3.2, "Syntaxe"**) :

```
== Introduction ==
Ce document vous expliquera ce que sont '''Tomcat''' et '''Cocoon''' dans
le cadre du développement d'une ''application Web''.
== Technologies utilisées ==
=== Tomcat ===
Tomcat est un serveur d'applications Java coté serveur.
Il gère :
* Les servlets
* Les JSP
Pour en savoir plus : [http://jakarta.apache.org/tomcat Site officiel de Tomcat]
=== Cocoon ===
[Image:cocoon.gif]
Cocoon est un framework de publication Web basé sur Java et XML.
Pour en savoir plus : [http://cocoon.apache.org Site officiel de Cocoon]
```

Après transformation en HTML, l'output que nous obtenons est le suivant :

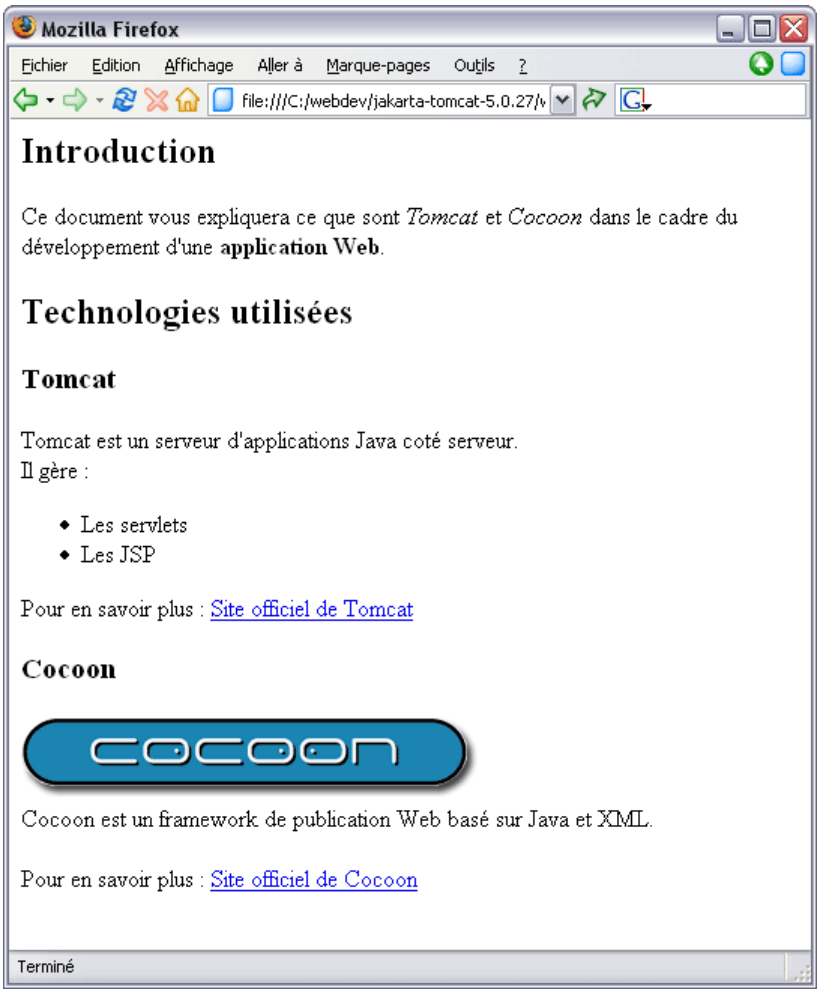

Figure 7 - Output dans Mozilla Firefox

3.3.5. Utilité?

La partie scientifique du site d'eTransparence sera capable d'importer dynamiquement des articles depuis l'encyclopédie Wikipédia. Comme c'était le cas pour les nouvelles importées de différents sites d'actualités, ce mécanisme a l'avantage de ne nécessiter aucune mise à jour de la part de l'administrateur du site. Comme les articles importés sont en syntaxe Wiki, le défi sera de transformer cette syntaxe spéciale en HTML pour l'afficher sur le site.

Voir **Chapitre 8, Section 3.3, “Partie scientifique”**.

Chapitre 3. Apache Cocoon

1. Définition

Cocoon est un framework permettant de réaliser des applications Web (applications s'exécutant sur un navigateur Web) flexibles et sophistiquées sur base des langages XML et Java.

Un framework est un ensemble de bibliothèques et d'outils qui permettent un développement plus rapide d'une application, du fait que beaucoup de choses ont déjà été implémentées. Ces composants sont organisés pour être en interaction les uns avec les autres et pour fonctionner comme un tout.

Cocoon est un projet Open Source de l'Apache Software Foundation (ASF). Ses principaux atouts sont la publication des pages en de multiples formats et la séparation des tâches.

2. Technologies

Le coeur de Cocoon est entièrement codé en Java. Cependant, pour développer des applications avec Cocoon, aucune programmation n'est nécessaire, tout se décrit dans des fichiers XML.

Cocoon est une servlet qui nécessite un serveur d'applications pour fonctionner. Nous avons choisi Tomcat pour assurer ce rôle. Il existe néanmoins d'autre serveurs d'applications tout aussi performants, par exemple Jetty.

3. Concepts

3.1. Formats multiples

Cocoon a pour but de changer l'organisation et la publication de l'information sur le Web. En effet, il facilite la mise à disposition de l'information sous de multiples formats et plate-formes.

Les données sources de l'information peuvent être de natures multiples telles que des fichiers textes, des fichiers XML (surtout), des bases de données, etc. Il suffit de modifier le module de présentation pour modifier l'apparence ou le format de rendu final (HTML, WML, TXT, PDF, ZIP, JPEG, SWF, etc.).

Cette fonctionnalité est réellement pratique lorsque des données doivent être mises à disposition des clients sous différents formats. Cocoon permet donc de publier rapidement des sites multi-supports et résout de manière simple les problèmes d'incompatibilité entre navigateurs.

3.2. Séparation des tâches

Le concept de séparation des tâches (*Separation of Concerns* en anglais) permet de distinguer les différents "métiers" que l'on retrouve dans un cycle de publication d'informations sur le Web.

Il est composé de quatre contextes de travail différent :

- Gestion du site : décisions sur le contenu, le comportement et l'apparence du site.
- Contenu : écriture et gestion du contenu du site.
- Style : présentation de l'information, aspect et apparence graphique du site.
- Logique : intégration avec les technologies de génération de contenu dynamique et les systèmes de base de données.

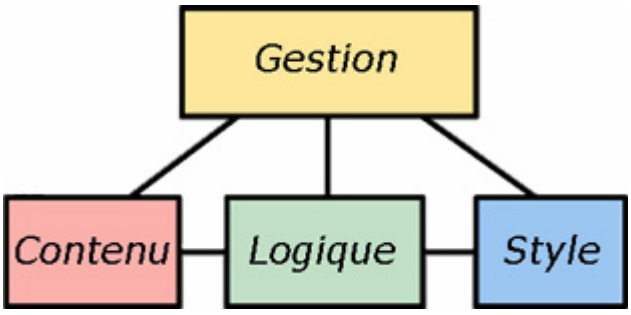

Figure 8 - Séparation des tâches dans Cocoon

Cette séparation permet à chaque couche d'être conçue, créée et gérée indépendamment. Cela permet une meilleure gestion des ressources humaines, en fournissant à chaque individu son travail et en réduisant au minimum les discussions croisées entre différents contextes de travail.

La séparation des tâches de Cocoon est une implémentation de l'architecture MVC. MVC (*Model-View-Control*) est un patron de conception (*design pattern* en anglais) pour le développement d'applications logicielles qui sépare le modèle de données, l'interface utilisateur et la logique de contrôle. Cocoon n'implémente pas "à la lettre" l'architecture MVC. C'est pour cela qu'on parle parfois de MVC+ lorsque l'on parle de la séparation des tâches dans Cocoon.

3.3. Les deux faces de Cocoon

Cocoon propose deux aspects différents : l'aspect utilisateur et l'aspect développeur.

L'aspect utilisateur est ce que nous allons étudier dans ce mémoire, il s'agit de créer des applications Web avec les possibilités (nombreuses) offertes par Cocoon. L'utilisateur de Cocoon n'a besoin que de connaître le XML (et ses dialectes) comme prérequis. En effet, pour créer un application avec Cocoon, il suffit de déclarer les composants que l'on utilise dans un fichier XML (voir **Chapitre 3, Section 6.1, "Sitemap"**). C'est d'ailleurs pour cette raison qu'on dit que Cocoon est déclaratif[1].

[1] Par opposition, par exemple, aux paradigmes tels que le procédural, l'orienté objet ou le fonctionnel.

L'aspect développeur permet à celui s'en sentant capable ou en ressentant le besoin de modifier ou d'étendre les fonctionnalités de Cocoon. Ceci est rendu possible par le fait que Cocoon est totalement Open Source. Étant donné que Cocoon est développé entièrement en Java, la connaissance de ce langage est requise pour pouvoir se lancer dans le développement de nouveaux composants. Cet aspect ne sera pas étudié dans ce mémoire.

4. Historique

4.1. Stefano Mazzocchi

Figure 9 - Stefano Mazzocchi

Cet informaticien italien, membre du groupe Apache, est le fondateur du framework Cocoon. C'est en 1998, à l'age de 23 ans, alors qu'il n'était à l'époque que stagiaire chez Apache, qu'il a eu l'idée de créer un système permettant de séparer les tâches et d'automatiser la production de pages Web.

4.2. Cocoon 1

Le projet Cocoon a donc été initié en Janvier 1999 par Stefano Mazzocchi. Cette première mouture de Cocoon, qui ne comptait qu'une centaine de lignes de code, servait uniquement à transformer du XML avec l'aide de l'API DOM (Document Object Model). L'utilisation de cette API d'accès aux données a vite montré ses limites. En effet, elle entraînait des problèmes de vitesse et une utilisation peu rationnelle de la mémoire.

4.3. Cocoon 2

Le développement de Cocoon 2 a commencé en parallèle à celui de Cocoon 1. Cocoon 2 a ainsi vu le jour en Novembre 2001. Il apporte de nombreuses améliorations et corrige notamment les problèmes de vitesse et d'utilisation mémoire.

L'utilisation de l'API SAX (Simple API for XML Parsing) pour accéder aux données XML permet de travailler avec de gros fichiers XML sans pénaliser les performances.

Parmi les améliorations techniques il faut particulièrement retenir les fonctions de pré-compilation et de mises en caches qui ont été introduites dans cette version. Pour simplifier l'utilisation de Cocoon, les fonctions de management ont été revues et centralisées. Cocoon est depuis sa version 2 parfaitement adapté à une utilisation en production.

5. Installation

Voir annexe ii.

6. Architecture

6.1. Sitemap

6.1.1. Vue d'ensemble

Le sitemap est le fichier central d'une application Cocoon. Son but est de permettre aux non-programmeurs de créer des applications Web à partir de composants logiques et de documents XML. Le sitemap est défini à la racine de chaque site dans le fichier `sitemap.xmap`.

Le sitemap est un simple fichier XML respectant la DTD définie pour les sitemaps. Il est ainsi très simple d'éditer ce fichier pour y ajouter de nouveaux éléments. Ce fichier XML est compilé préalablement pour gagner en performances.

6.1.2. Structure du fichier

Le sitemap a deux rôles principaux :

- Déclarer les composants qui seront ensuite utilisés dans les pipelines
- Définir les pipelines qui utilisent les composants déclarés

Voici la configuration minimale d'un sitemap :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!------------------------>
<!----- sitemap.xmap ----->
<!------------------------>
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">

    <map:components>
        <map:generators default="file"/>
        <map:transformers default="xslt"/>
        <map:readers default="resource"/>
        <map:serializers default="html"/>
        <map:selectors default="browser"/>
        <map:matchers default="wildcard"/>
    </map:components>

    <map:pipelines>
        <map:pipeline>
            <!-- respond to *.html requests with
                 our docs processed by doc2html.xsl -->
            <map:match pattern="*.html">
                <map:generate src="{1}.xml"/>
                <map:transform src="doc2html.xsl"/>
                <map:serialize type="html"/>
            </map:match>
        </map:pipeline>
    </map:pipelines>

</map:sitemap>
```

A ce stade, il n'est pas nécessaire de comprendre dans les détails le contenu de ce fichier mais seulement d'en connaître la structure générale.

Le fichier se décompose en deux parties : la déclaration des composants (`<map:components/>`) et la déclaration des pipelines (`<map:pipelines/>`). Les composants qu'il est possible de déclarer sont les generators, les transformers, les readers, les serializers, les selectors et les matchers. Les pipelines correspondent à l'empilement de ces différents composants pour déterminer le traitement complet d'une requête. Dans notre fichier, nous avons fait se succéder un matcher qui permet de correspondre avec une requête, un generator qui permet de générer un fichier XML, un transformer qui permet de transformer ce dernier et un serializer qui permet de créer l'output final.

Toutes ces nouvelles notions sont détaillées dans les sections qui arrivent.

6.2. Pipelines

Cocoon utilise la notion de pipelines de composants, chaque composant de la chaîne étant spécialisé dans une opération particulière. Cette architecture permet d'utiliser une technique de type Lego pour construire des applications Web en assemblant les différents composants en pipelines sans aucune programmation. L'assemblage des composants permet d'obtenir un système complet.

Un pipeline "minimal" doit contenir au moins trois composants qui sont :

- Le matcher qui permet d'associer une URI à un pipeline.
- Le generator qui créé un flux XML comme entrée du pipeline.
- Le serializer qui convertit la structure XML pour l'affichage final.

Un pipeline "classique" ajoute une étape supplémentaire entre la lecture par le generator et la restitution par le serializer. Ce travail est effectué par un autre composant : le transformer qui permet de modifier les données XML.

Voici un exemple de pipeline dans le sitemap :

```
<map:pipeline>
    <!-- respond to index.html requests with
         our docs processed by doc2html.xsl -->
    <map:match pattern="index.html">
        <map:generate src="index.xml"/>
        <map:transform src="doc2html.xsl"/>
        <map:serialize type="html"/>
    </map:match>
</map:pipeline>
```

Si l'URI entrée est `index.html`, on génère un flux à partir du fichier `index.xml`, on transforme ce flux par l'intermédiaire de la feuille de style `doc2html.xsl` et pour terminer, on sérialise au format HTML, c'est-à-dire on crée la page HTML qui sera affichée sur le navigateur.

6.3. Les composants

Les composants sont les différentes briques que l'on assemble pour déterminer un pipeline complet. Il existe une petite dizaine de composants. Nous ne détaillerons ici que les plus importants : les matchers, les generators, les transformers, les serializers, les readers, les selectors et les actions.

6.3.1. Matchers

Les matchers permettent de faire l'association entre une URI (une requête) et un pipeline défini dans le sitemap de Cocoon. Cocoon effectue les traitements en fonction de la requête du client. Lorsqu'une requête correspond à un matcher d'un pipeline, le traitement de ce pipeline débute.

Il existe plusieurs types de matchers dans Cocoon. Le matcher par défaut s'appelle *WildcardURI-Matcher*. Son fonctionnement est expliqué dans le tableau ci-dessous.

Table 3.1. Fonctionnement du WildcardURI-Matcher

Pattern	Description
**	Matche zéro ou plusieurs caractères en incluant le caractère slash ("/")
*	Matche zéro ou plusieurs caractères en excluant le caractère slash ("/")
\	Le caractère backslash est utilisé comme un échappatoire. "*" matche le caractère astérisque ("*") et "\\" matche le caractère backslash ("\")

Les patterns * et ** sont remplacés par des variables lors du match d'une requête. Ces variables sont accessibles en utilisant la chaîne de caractère "{N}" où N est le numéro de la variable, dans l'ordre de lecture de la requête (de gauche à droite). {0} représente la requête complète.

Pour mieux comprendre, voici un exemple :

On considère le matcher : `*/*/*/**.html` et la requête : `a/request/for/a/cocoon/resource.html`.

- {1} = `a`
- {2} = `request`
- {3} = `for`
- {4} = `a/cocoon/resource`
- {0} = `a/request/for/a/cocoon/resource`

Il est important de noter que Cocoon utilise le premier pipeline qu'il rencontre pouvant répondre à la requête (un peu à la manière des templates de XSLT). L'ordre de définition des pipelines dans le fichier sitemap est donc très important. Les matchers plus spécifiques doivent apparaître avant les matchers génériques.

Vous trouvez ci-dessous un exemple complet qui démontre toute la puissance des matchers :

```
<map:pipeline>

 <map:match pattern="contact.html"> <!-- 1 -->
  <map:generate src="contact.xml"/>
  <map:transform src="sheet2.xsl"/>
  <map:serialize type="xhtml"/>
 </map:match>

 <map:match pattern="*.html"> <!-- 2 -->
  <map:generate src="{1}.xml"/>
  <map:transform src="sheet1.xsl"/>
  <map:serialize type="html"/>
 </map:match>

 <map:match pattern="**.html"> <!-- 3 -->
  <map:generate src="others/{1}.xml"/>
  <map:transform src="sheet1.xsl"/>
  <map:serialize type="html"/>
 </map:match>

 <map:match pattern="**"> <!-- 4 -->
  <map:generate src="errors/404.xml"/>
  <map:transform src="sheet1.xsl"/>
  <map:serialize type="html"/>
 </map:match>

</map:pipeline>
```

Nous avons défini quatre pipelines numérotés de 1 à 4. Posons quelques requêtes et voyons à quels pipelines elles correspondent.

- La requête `contrat.html` va matcher avec le pipeline 2
- La requête `contact.html` va matcher avec le pipeline 1
- La requête `another/contact.html` va matcher avec le pipeline 3
- Les requêtes `contact.htm` et `another/contact.htm` vont toutes les deux matcher avec le pipeline 4

Contrairement à nos habitudes, l'URI ne fait donc pas référence à un fichier sur le serveur mais à un pipeline du sitemap. C'est pour cela qu'on parle d'URI (Uniform Resource Identifier, qui fait référence à une ressource sans la localiser) et pas d'URL (Uniform Resource Locator, qui fait référence à une ressource en la localisant).

Pour plus d'informations : http://cocoon.apache.org/2.1/userdocs/matchers/matchers.html

6.3.2. Generators

Un generator est le point d'entrée d'un pipeline XML. Il génère du contenu XML et initialise le traitement du pipeline. Chaque pipeline commençant par un generator doit se terminer par un serializer.

Le generator par défaut est le *File Generator* c'est-à-dire le générateur de fichier.

En voici un exemple :

```
<map:generate src="document.xml" type="file"/>
<!-- L'attribut type peut être omis car il s'agit du générateur par défaut. -->
```

Un autre générateur intéressant est le *Directory Generator* qui génère un fichier XML décrivant les fichiers d'un répertoire donné, par exemple :

```
<map:generate src="files/articles" type="directory"/>
```

Voici un exemple de résultat de ce générateur :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<dir:directory name="files/articles"
               lastModified="1110797896078"
               date="14/03/05 11:58"
               size="0"
               sort="name"
               reverse="false"
               requested="true"
               xmlns:dir="http://apache.org/cocoon/directory/2.0">
 <dir:file name="article.pdf" lastModified="1092486882000"
           date="14/08/04 14:34" size="75638"/>
 <dir:file name="data.txt" lastModified="1110797857203"
           date="14/03/05 11:57" size="347"/>
 <dir:file name="test.xml" lastModified="1114634865027"
           date="27/04/05 22:47" size="797"/>
</dir:directory>
```

Ce qui donne de manière schématique :

Figure 10 - Génération d'un répertoire de fichiers

Le *Directory Generator* a donné naissance à quelques enfants. *Imagedirectory Generator* s'applique aux répertoires d'images. Il donne des attributs supplémentaires comme la hauteur et la largeur de l'image. *Mp3directory Generator* est utilisé pour les répertoires contenant de la musique au format MP3. Les attributs supplémentaires sont la fréquence, le bitrate, le mode (mono, stéréo), etc.

Pour plus d'informations : http://cocoon.apache.org/2.1/userdocs/generators/generators.html

6.3.3. Aggregators

Un aggregator n'est pas vraiment un composant d'un pipeline, il s'agit plutôt d'une alternative au generator, permettant de concaténer plusieurs fichiers à la génération.

En voici un exemple :

```
<map:match pattern="home">
 <map:aggregate element="document"> <!-- document = new root element -->
   <map:part src="menu.xml"/>
   <map:part src="home.xml"/>
 </map:aggregate>
 <map:serialize type="xml"/>
</map:match>
```

Le fichier résultat sera la concaténation du fichier `menu.xml` et du fichier `home.xml` englobée par une nouvelle racine : l'élément `document`.

6.3.4. Transformers

Un transformer est le point central d'un pipeline. Il permet de faire des modifications sur les données avant le rendu final par le serializer. Les transformers sont toujours placés dans un pipeline entre un generator et un serializer. Il est possible de faire se succéder plusieurs transformers pour arriver au résultat désiré.

Le transformer par défaut est le *XSLT Transformer*.

En voici un exemple :

```
<map:pipeline>
    <map:match pattern="index.html">
        <map:generate src="index.xml"/>
        <map:transform src="doc2html.xsl"/>
        <map:serialize type="html"/>
    </map:match>
</map:pipeline>
```

Le contenu XML du fichier `index.xml` est transformé en contenu HTML par l'intermédiaire de la feuille de style `doc2html.xsl`.

Il existe de nombreux autres transformers. Le plus intéressant d'entre eux est le *I18N Transformer* (transformer d'internationalisation) qui permet de choisir la langue dans laquelle sera affiché le document, grâce à des catalogues prédéfinis (voir **Chapitre 5, Section 6, “Internationalisation (I18N)”**).

Pour plus d'informations :
http://cocoon.apache.org/2.1/userdocs/transformers/transformers.html

6.3.5. Serializers

Un serializer est la dernière étape d'un pipeline. Il transforme la sortie du dernier transformer en flux de données binaires ou de caractères pour le rendu final au client.

Le serializer par défaut est le *HTML Serializer*. Les autres serializers les plus souvent utilisés sont le *XHTML Serializer* qui renvoie du XHTML standard, le *XML Serializer* qui crée un document XML, le *Zip Serializer* qui crée une archive de fichiers, le *PDF Serializer* qui crée un document PDF, etc.

Exemple utilisant le *PDF Serializer* :

```
<map:pipeline>
    <map:match pattern="test.pdf">
        <map:generate src="index.xml"/>
        <map:transform src="doc2fo.xsl"/>
        <map:serialize type="fo2pdf"/>
    </map:match>
</map:pipeline>
```

Ce serializer implique qu'on ait transformé le flux XML de départ avec XSL-FO. Nous montrons un exemple d'une telle transformation dans l'exemple récapitulatif.

Pour plus d'informations : http://cocoon.apache.org/2.1/userdocs/serializers/serializers.html

6.3.6. Readers

Un reader est à la fois le point d'entrée et la sortie d'un pipeline. Il remplace en une seule déclaration le trio generator-transformer(s)-serializer (les trois composants de base d'un pipeline). Les readers sont très pratiques pour obtenir du contenu binaire tel que des images.

En voici un exemple :

```
<map:match pattern="images/*.jpg">
   <map:read mime-type="images/jpg" src="images/{1}.jpg"/>
</map:match>
```

Ce pipeline permet de pouvoir afficher toutes les images JPEG (c'est-à-dire ayant le type MIME : `images/jpg`) de l'application Web.

Pour plus d'informations : http://cocoon.apache.org/2.1/userdocs/readers/readers.html

6.3.7. Selectors

Les selectors dans Cocoon ont un rôle semblable aux matchers mais ils permettent une plus grande flexibilité. Les selectors sont conçus pour évaluer des expressions booléennes simples concernant une partie particulière de l'environnement. Le résultat de ces tests permet de déterminer quels sont les traitements à effectuer à l'intérieur le pipeline.

Contrairement aux matchers qui ne permettent que d'exécuter ou non un pipeline, les selectors permettent de construire des sélections plus complexes. Typiquement, les matchers peuvent être considérés comme de simples "if" tandis que les selectors correspondent à des "if-then-else" ou à des "switch-case".

Le selector par défaut est le *Browser Selector* qui permet de détecter le navigateur utilisé par le client à partir de son User Agent. L'User Agent est une chaîne de caractère qui permet d'identifier le navigateur du client ainsi que sa version (par exemple `Mozilla/5.0 (Windows; U; Windows NT 5.1; fr-FR; rv:1.7.7) Gecko/20050414 Firefox/1.0.3` est un User Agent possible de Mozilla Firefox).

L'exemple ci-dessous présente un pipeline utilisant le *Browser Selector*. La première chose à faire est de déclarer le composant dans le sitemap :

```
<map:components>

<map:selectors default="browser">
  <map:selector name="browser"
    src="org.apache.cocoon.selection.BrowserSelector"
    logger="sitemap.selector.browser">
    <browser name="explorer" useragent="MSIE (...)"/>
    <browser name="mozilla" useragent="Mozilla (...)"/>
  </map:selector>
</map:selectors>

<!-- ... -->

</map:components>
```

L'élément `<browser/>` permet d'associer un nom à un certain User Agent.

Ensuite, nous allons créer un pipeline qui réagit différemment selon que le navigateur est Internet Explorer, Mozilla ou autre chose.

```
<map:match pattern="*.html">
  <map:generate src="{1}.xml"/>

  <map:select type="browser">
    <map:when test="mozilla">
      <map:transform src="stylesheets/mozilla.xsl" />
    </map:when>
    <map:when test="explorer">
      <map:transform src="stylesheets/ie.xsl" />
    </map:when>
    <map:otherwise>
      <map:transform src="stylesheets/default.xsl" />
    </map:otherwise>
  </map:select>

  <map:serialize/>
</map:match>
```

Dans cet exemple, le selector s'applique à la transformation mais cela peut s'applique aussi à d'autres composants ou groupes de composants.

Un autre selector intéressant est le *Parameter Selector*. Il permet de faire le test, non plus sur le type de navigateur utilisé, mais sur un paramètre qu'on lui définit :

```
<map:match pattern="*.html">

  <map:select type="parameter">
    <map:parameter name="parameter-selector-test" value="{1}"/>
    <map:when test="home">
      <map:generate src="homepage.xml"/>
      <map:transform src="stylesheets/home.xsl"/>
    </map:when>
    <map:otherwise>
      <map:generate src="{1}.xml"/>
      <map:transform src="stylesheets/default.xsl"/>
    </map:otherwise>
  </map:select>
```

```
  <map:serialize type="xhtml"/>
</map:match>
```

L'instruction `<map:parameter name="parameter-selector-test" value="{1}"/>` définit la valeur de `test` en fonction du contenu de la requête. Si la requête est `home.html`, on applique un autre traitement que pour les autres pages se terminant par `.html`.

Pour plus d'informations : http://cocoon.apache.org/2.1/userdocs/selectors/selectors.html

6.3.8. Actions

Les actions permettent de mettre des données externes à disposition du sitemap, pour qu'il puisse exécuter des fonctionnalités spéciales, comme l'envoi d'un e-mail, l'appel à un script ou à une base de données, etc.

Pour plus d'informations : http://cocoon.apache.org/2.1/userdocs/actions/actions.html

6.4. Récapitulatif de l'exécution d'une requête

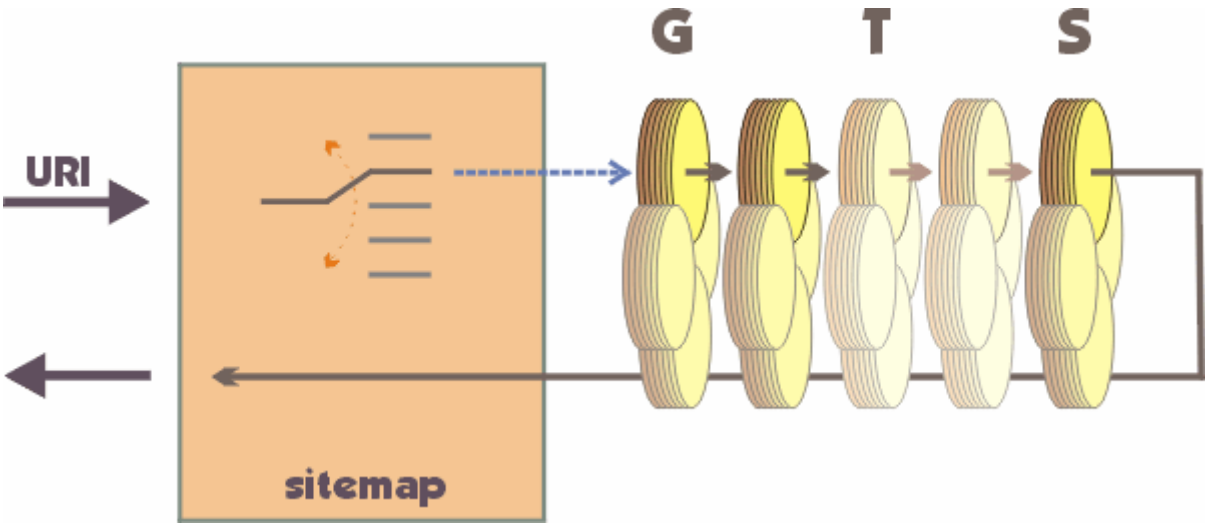

Figure 11 - Exécution d'une requête avec Cocoon

Le schéma ci-dessus montre les étapes parcourues depuis la requête initiale (sous forme d'une URI) jusqu'au rendu final à l'utilisateur. Le matcher est représenté sous la forme d'un interrupteur qui se "branche" sur un pipeline et qui l'exécute. Cette exécution commence par un generator, ensuite un ou plusieurs transformers sont appliqués et pour terminer, nous avons un serializer. Le résultat est finalement renvoyé au navigateur du client.

L'interaction entre le client et le serveur peut être représentée par le diagramme de séquence UML ci-dessous :

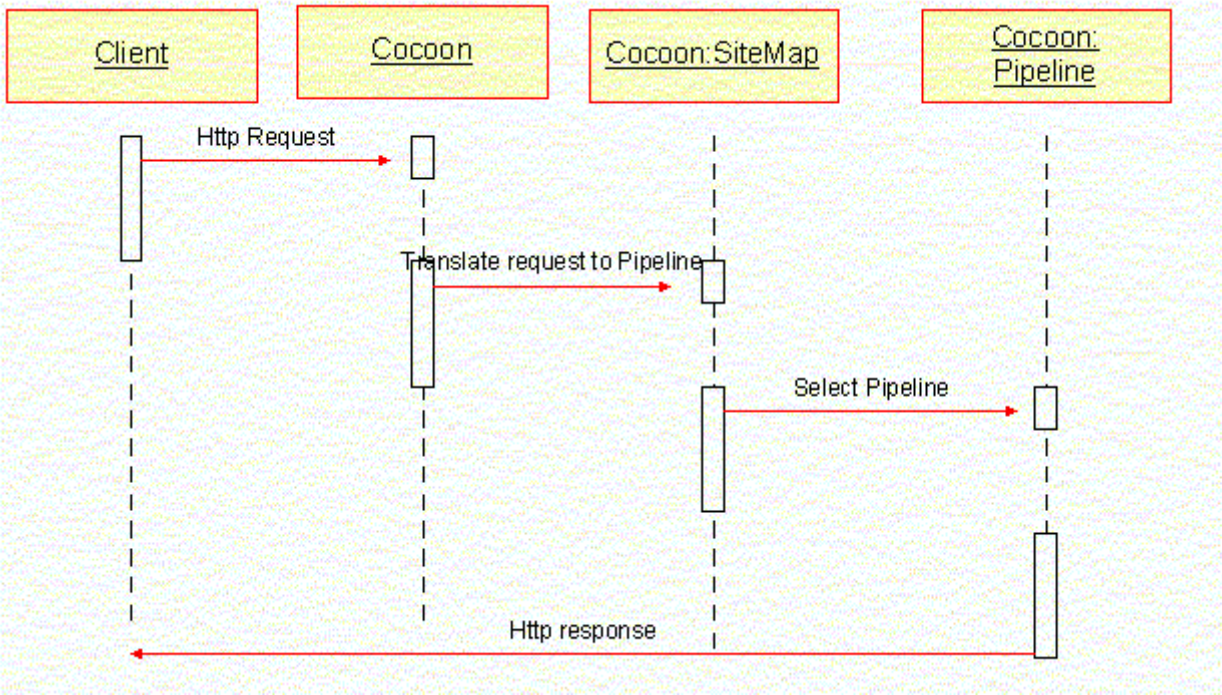

Figure 12 - Diagramme de séquence de l'exécution d'une requête

6.5. Autres concepts

6.5.1. Redirections

Lors du match d'une requête, il est possible de faire une redirection vers une autre URI. C'est la balise `<map:redirect-to/>` qui permet de réaliser cela :

```
<map:match pattern="*.htm">
   <map:redirect-to uri="{1}.html"/>
</map:match>
```

Cet exemple n'est pas pris au hasard, il permet de ne pas pénaliser les personnes ayant omis le "l" final de l'extension ".html".

Une autre utilisation des redirections est la suivante :

```
<map:pipeline>
    <map:match pattern="index.html">
        <map:generate src="index.xml"/>
        <map:transform src="doc2html.xsl"/>
        <map:serialize type="html"/>
    </map:match>
    <!-- ... -->
    <map:match pattern="**">
        <map:redirect-to uri="index.html"/>
    </map:match>
</map:pipeline>
```

Cette technique permet de rediriger le visiteur vers la page d'accueil si aucun pipeline ne matche avec l'URI demandée, sans quoi une page d'erreur aurait été affichée.

Pour plus d'informations : http://cocoon.apache.org/2.1/userdocs/concepts/redirection.html

6.5.2. Sous-sitemaps

Il est possible de déléguer certaines tâches du sitemap à des sous-sitemaps. Cela permet de garder des fichiers sitemaps lisibles et d'une taille raisonnable. Dans le vocabulaire de Cocoon, on dira qu'on "monte" un sous-sitemap.

Exemple :

```
<map:match pattern="files/*">
   <map:mount src="files/sitemap.xmap" uri-prefix="files"/>
</map:match>
```

Si la requête émise commence par `files/`, on monte le sitemap situé à l'emplacement défini par la valeur de l'attribut `src`. La valeur de l'attribut `uri-prefix` est la partie qui doit être supprimée de la requête. Par exemple, si la requête est `files/logo`, le préfixe `files/` est supprimé de l'URI et `logo` est passé au sous-sitemap situé à l'emplacement `files/sitemap.xmap`.

Pour plus d'informations :
http://cocoon.apache.org/2.1/userdocs/concepts/sitemap.html#Mounting+sitemaps

6.5.3. XSP (Extensible Server Pages)

XSP est une technologie propriétaire de Cocoon construite sur les idées de JSP. Le langage XSP permet des créer des feuilles logiques (*logicsheets* en anglais) contenant des balises dynamiques. Ces balises dynamiques sont remplacées par du contenu statique lors de l'exécution de la page. On peut donc comparer le fonctionnement de XSP à ceux d'ASP, PHP et JSP. XSP ressemble d'autant plus à JSP que les instructions contenues dans ses balises dynamiques sont en langage Java.

Les feuilles logiques brisent un peu l'objectif de Cocoon d'arriver à séparer les tâches complètement. Elles insèrent des instructions dynamiques au beau milieu du contenu de la page.

Ci-dessous, nous présentons un exemple de page XSP (`page.xsp`) :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!------------------------>
<!----- page.xsp --------->
<!------------------------>
<xsp:page language="java" xmlns:xsp="http://apache.org/xsp">
 <page>
  <title>Votre première page XSP</title>
  <content>
   <para>
    <!-- instruction XSP -->
    <xsp:logic>
```

```
     String msg = "Hello, world!";
    </xsp:logic>
    <!-- affichage du contenu d'une variable -->
    <xsp:expr>msg</xsp:expr>
   </para>
   <para>How are you?</para>
  </content>
 </page>
</xsp:page>
```

Les instructions logiques sont placées à l'intérieur de la balise `<xsp:logic/>`. La balise `<xsp:expr/>` sert à imprimer à l'écran la valeur d'une variable.

Voici le pipeline que va traverser notre page XSP dans le sitemap :

```
<map:match pattern="*.xsp">
  <map:generate type="serverpages" src="{1}.xsp"/>
  <map:transform src="stylesheets/xspsheet.xsl"/>
  <map:serialize type="html"/>
</map:match>
```

Par rapport au traitement d'une page XML statique, seul le générateur varie.

Voici le contenu de la feuille de style XSLT (`xspsheet.xsl`) :

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

 <xsl:template match="/"> <!-- création du document -->
  <html>
   <head>
    <title><xsl:value-of select="//page/title"/></title>
   </head>
   <body>
    <h1><xsl:value-of select="//page/title"/></h1>
    <xsl:apply-templates select="//page/content"/>
   </body>
  </html>
 </xsl:template>

 <xsl:template match="content"> <!-- création du bloc principal -->
  <div id="text">
   <xsl:apply-templates/>
  </div>
 </xsl:template>

 <xsl:template match="para"> <!-- création de paragraphes -->
  <p>
   <xsl:value-of select="."/>
  </p>
 </xsl:template>

</xsl:stylesheet>
```

L'output que nous obtenons est le suivant :

```
<html>
```

```
 <head>
  <title>Votre première page XSP</title>
 </head>
 <body>
  <h1>Votre première page XSP</h1>
  <div id="text">
   <p>Hello, world!</p>
   <p>How are you?</p>
  </div>
 </body>
</html>
```

Pour mieux comprendre le traitement de notre page XSP dans le pipeline, observons le schéma ci-dessous :

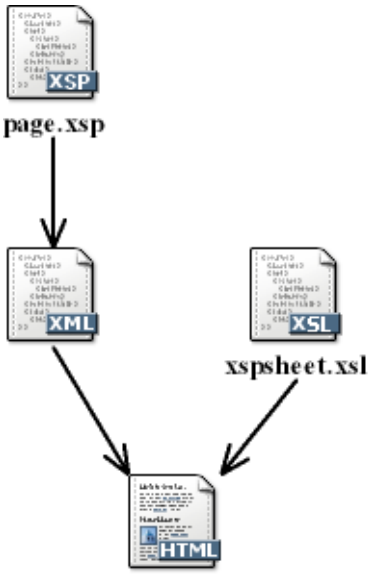

Figure 13 - Traitement d'une page XSP

Tout d'abord, notre page XSP va être interprétée par le serveur qui va renvoyer du contenu en XML, de la même manière qu'un serveur Apache pourrait interpréter une page PHP et retourner du contenu en HTML. Ensuite, nous allons transformer ce contenu XML en HTML grâce à une feuille de style XSLT et pour finir, nous allons sérialiser au format HTML.

Pour plus d'informations : http://cocoon.apache.org/2.1/userdocs/xsp/index.html

Chapitre 4. Cocoon comparé à PHP

Ce chapitre a pour but de comparer le framework Cocoon à un langage isolé. Le langage qui a été choisi est le PHP, d'une part car c'est le langage dynamique le plus répandu sur Internet en ce moment et, d'autre part parce qu'il nous est très familier.

Dans un premier temps, nous ferons la comparaison, point par point, en toute neutralité. Ensuite, nous ferons une conclusion de cette comparaison pour déterminer concrètement dans quel cas PHP est meilleur que Cocoon et inversement.

La connaissance du PHP est supposée connue. Si tel n'est pas le cas, nous reportons le lecteur à [2] et [8].

1. Comparaison

1.1. Prix

Cocoon et PHP sont tous les deux distribués sous licence GNU GPL (GNU General Public Licence).

L'objectif de la licence GNU GPL est de garantir à l'utilisateur les quatre libertés suivantes :

- Liberté 0 : liberté d'exécuter le logiciel, pour n'importe quel usage.
- Liberté 1 : liberté d'étudier le fonctionnement d'un programme et de l'adapter à vos besoins.
- Liberté 2 : liberté de redistribuer des copies.
- Liberté 3 : liberté d'améliorer le programme et de rendre publiques vos modifications afin que l'ensemble de la communauté en bénéficie.

Cette licence est libre et ne nécessite donc pas le moindre coût d'achat.

1.2. Documentation

Cocoon et PHP possèdent tous les deux une documentation très élaborée sur leurs sites respectifs.

PHP étant plus ancien et possédant une communauté gigantesque d'utilisateurs, il est beaucoup plus simple de trouver de la documentation "non-officielle" sur PHP que sur Cocoon. Il suffit de faire une recherche sur Google pour s'en rendre compte. Une recherche avec le mot-clé "php" donne presque un milliard et demi de résultats alors qu'une recherche avec le mot-clé "apache cocoon" donne seulement un peu plus de 900.000 résultats (*Source : Google Fight*).

A ce jour, nous n'avons recensé aucun site sérieux en français sur Cocoon alors que PHP en possède énormément.

Du point de vue des livres, on ne compte plus ceux consacrés à PHP alors qu'il n'en existe seulement que trois ou quatre pour Cocoon.

La carence de documentation de Cocoon a été un frein énorme à son étude dans le cadre de ce mémoire.

1.3. Hébergement

A part dans certains cas[2], la réalisation d'un site Web conduit toujours à sa mise en ligne, une fois celui-ci terminé. Pour mettre un site en ligne, il doit être hébergé "quelque part". Cet hébergement peut se faire soit sur sa propre machine (il faut alors installer les outils nécessaires, voir **Chapitre 4, Section 1.4, "Facilité d'installation"**), soit sur un serveur distant qui propose déjà tous les outils indispensables. La plupart du temps, c'est la deuxième solution qui est adoptée, surtout en entreprise.

Quasiment tous les hébergeurs au monde proposent le langage PHP dans sa version 4, beaucoup moins nombreux sont ceux qui proposent le récent PHP 5, mais ce n'est qu'une question de temps. C'est bien simple, un hébergeur sérieux n'envisage même pas de ne pas proposer à ses clients ou utilisateurs le langage PHP.

PHP est utilisé sur plus d'un site Web sur trois dans le monde ce qui représente plus de 17 millions de domaines et 1.400.000 adresses IP ! Plus de la moitié des serveurs Apache (49,96% au 1er septembre 2004) fonctionnent avec PHP. Apache est quant à lui utilisé par plus des deux tiers des sites Web ce qui représente 73,34% des sites Web publics dans le monde au 1er août 2004 (*Source :* *Netcraft*).

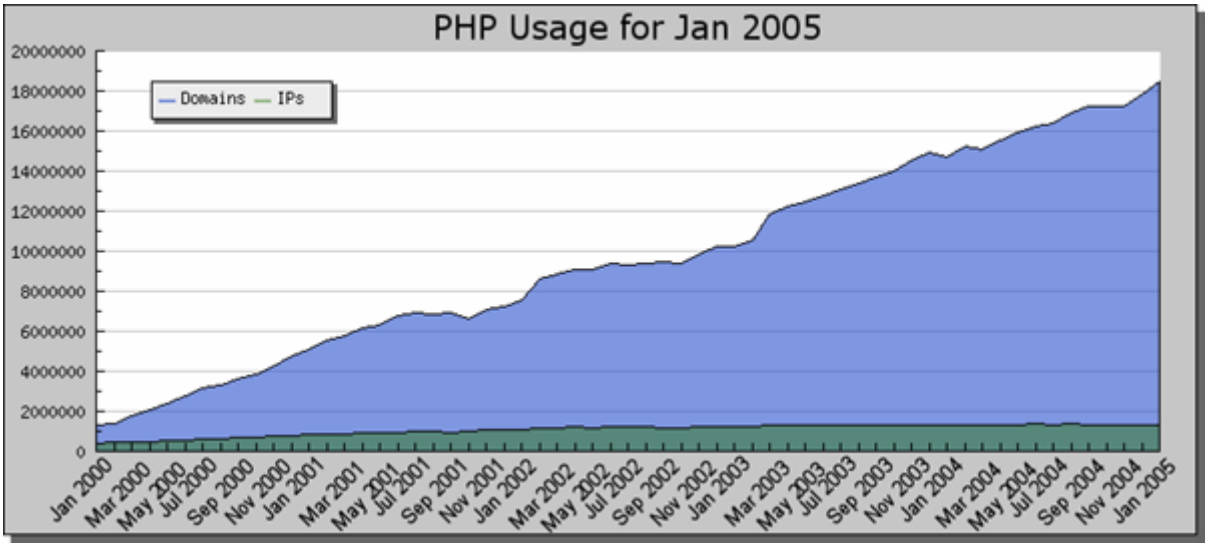

Figure 14 - Utilisation de PHP dans le monde au 1er février 2005

Pour Cocoon, le constat est moins glorieux, Apache recense sur son site (http://cocoon.apache.org/link/hosting.html) seulement 22 hébergeurs supportant Cocoon dont

[2] S'il s'agit d'un site Intranet, par exemple.

un seul gratuit ! Il recense également le nombre de sites Web "Powered by Cocoon" c'est-à-dire réalisés avec Cocoon. Il en dénombre seulement une grosse centaine. Cette information est toutefois à prendre avec des pincettes, car ce sont les webmasters eux-mêmes qui inscrivent leur(s) site(s) dans cet annuaire (ils ne sont pas obligés de le faire). Toutefois, cela nous donne une bonne idée du fossé énorme qui sépare Cocoon et PHP de ce point de vue là.

1.4. Facilité d'installation

Si l'on décide d'héberger des applications Web sur sa propre machine, il faut installer les serveurs sur celle-ci. Bien que cela fonctionne relativement bien sous Windows, on préférera installer nos serveurs sur une machine Linux pour des questions - qui ne sont plus à débattre - de performances et de stabilité.

L'installation de PHP sur une machine implique l'installation d'un serveur Web (Apache) et d'un SGBD (MySQL) si l'on veut travailler avec des bases de données. Pour information, le quatuor Linux, Apache, MySQL et PHP forme ce qu'on appelle la plate-forme LAMP, acronyme formé à partir de la première lettre de chacun de ses composants.

Pour un novice, l'installation peut s'avérer extrêmement difficile à surmonter, malgré les nombreux tutoriaux qui foisonnent sur Internet à ce sujet. Quoi qu'il en soit, quelques recherches et quelques questions sur des forums plus tard, on obtient la stabilité d'un serveur Apache personnel avec PHP et MySQL.

Notons que pour Windows, il existe des logiciels très pratiques qui permettent de faire tout ce travail à notre place. Les plus connus sont Easy-PHP (http://www.easyphp.org) et WAMP (par analogie à LAMP, http://www.wampserver.com). En général, ceux-ci installent Apache, PHP et MySQL via un programme d'installation classique. Une fois l'installation terminée, il est possible de gérer les composants grâce à une interface graphique sobre.

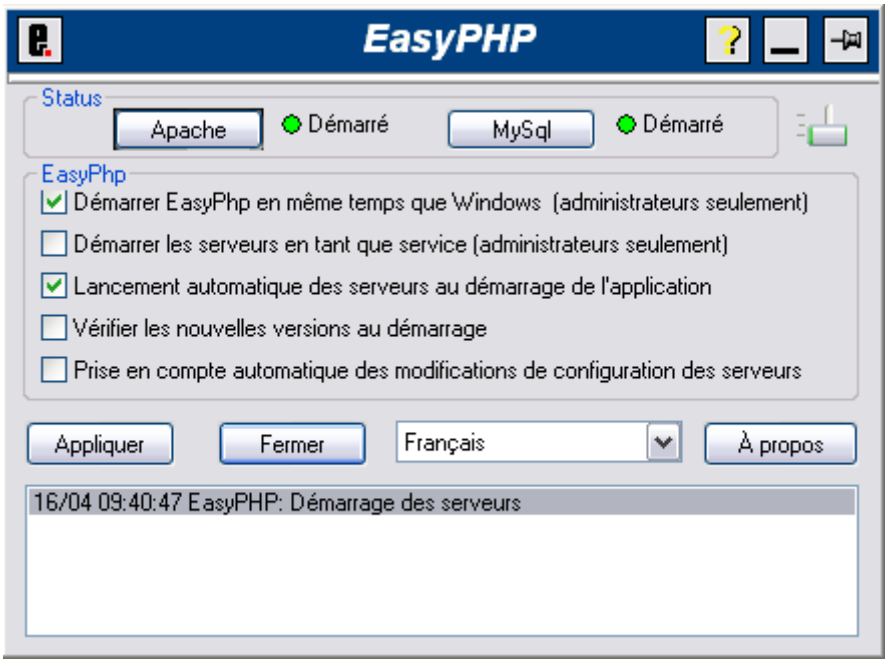

Figure 15 - Interface graphique d'Easy-PHP

L'installation de Cocoon (voir annexe ii) quant à elle implique l'installation d'une machine virtuelle Java et de Tomcat. Intrinsèquement, elle n'est pas plus aisée ni plus difficile que l'installation de PHP. Malheureusement, le fait que Cocoon dispose d'une documentation moindre que PHP est également valable pour son installation.

1.5. Facilité d'utilisation

De part son paradigme procédural, PHP est très simple à utiliser pour quiconque à déjà touché aux langages C ou Pascal, par exemple. De plus, le passage d'un site statique en HTML vers un site dynamique en PHP se fait très facilement car il suffit d'ajouter le code PHP à l'intérieur de la page HTML et de changer l'extension du fichier en `.php`.

Cocoon est un peu déroutant, car il utilise un paradigme (le paradigme déclaratif) qui est très éloigné de ce dont on a l'habitude de manipuler. Cela fait perdre un peu de temps au début et pousse même certains développeurs à aller voir ailleurs. Cependant, avec un minimum d'expérience, écrire en Cocoon devient très rapide, car il ne nécessite aucune programmation (à quelques exceptions près : les pages XSP et certains scripts demandent de programmer en Java et/ou JavaScript), contrairement à PHP.

Un point très important lorsqu'on développe dans un langage de programmation est de disposer d'un bon éditeur de texte ou d'un bon EDI. Les qualités principales de ces types d'outils sont :

- Coloration du code (pour la lisibilité)
- Auto-complétion (pour gagner en rapidité d'écriture)
- Détection et correction d'erreur (pour ne pas perdre un temps fou de debuggage)
- Possibilité de compilation (si le langage l'impose)
- Prévisualisation
- ...

Grâce à sa popularité, PHP dispose d'un grand nombre d'éditeurs et d'environnements de développement intégré. Le site PHP Editors recense les meilleurs éditeurs pour PHP pour toutes les licences et pour toutes les plates-formes (http://www.php-editors.com/review). Ceux que nous retiendrons sont PHP Designer 2005 pour Windows (voir copie d'écran ci-dessous) et Quanta Plus pour Linux (KDE).

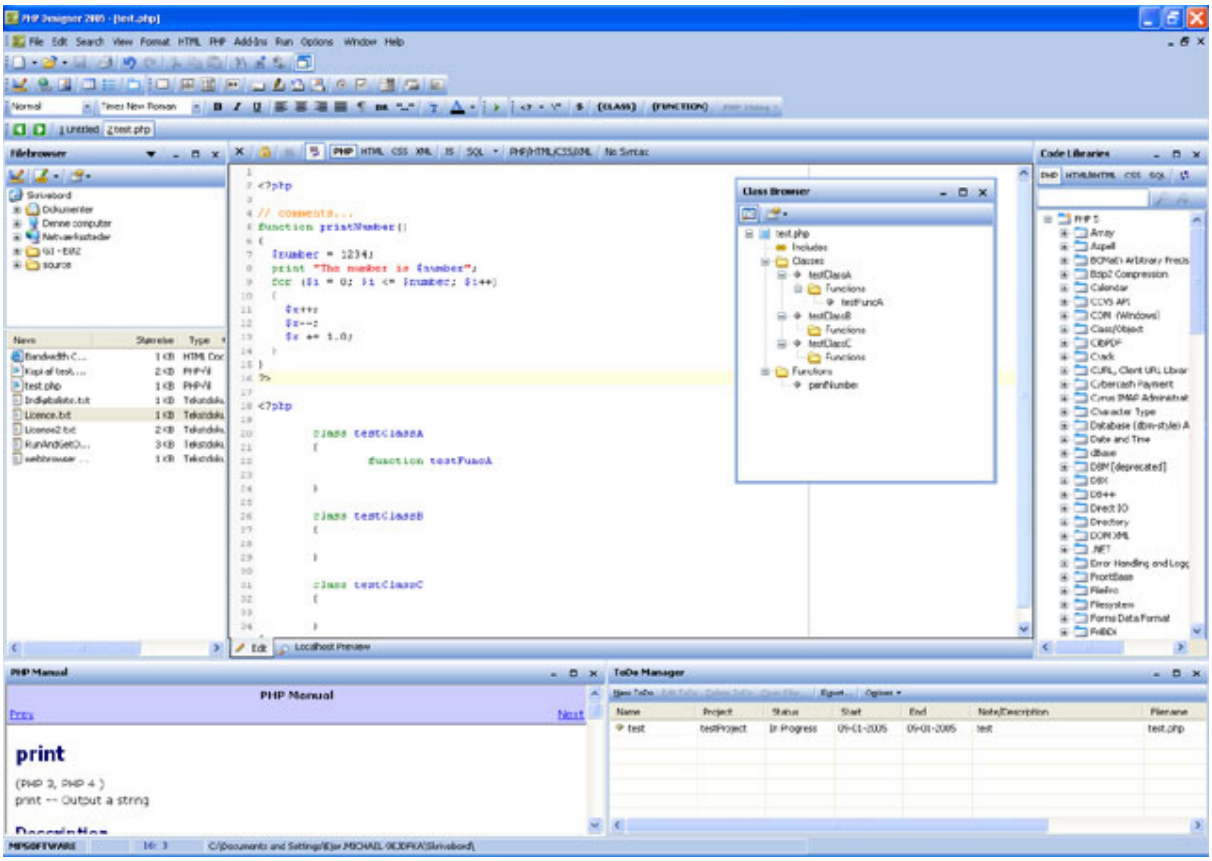

Figure 16 - Copie d'écran de PHP Designer 2005

Pour développer en Cocoon, n'importe quel éditeur XML peut faire l'affaire. De tels éditeurs sont assez nombreux sur le marché et il en apparaît de nouveaux chaque jour. De ce point de vue, XML n'a rien à envier à PHP.

Les éditeurs XML spécialisés pour Cocoon se font beaucoup plus rares. La plupart du temps, ils permettent de ne gérer que certains aspects isolés de Cocoon. Pollo (voir copie d'écran ci-dessous) fournit une aide pour l'écriture des sitemaps de Cocoon. Il s'agit d'un projet Open Source développé en Java. Malheureusement, son interface graphique très austère en rebutera plus d'un.

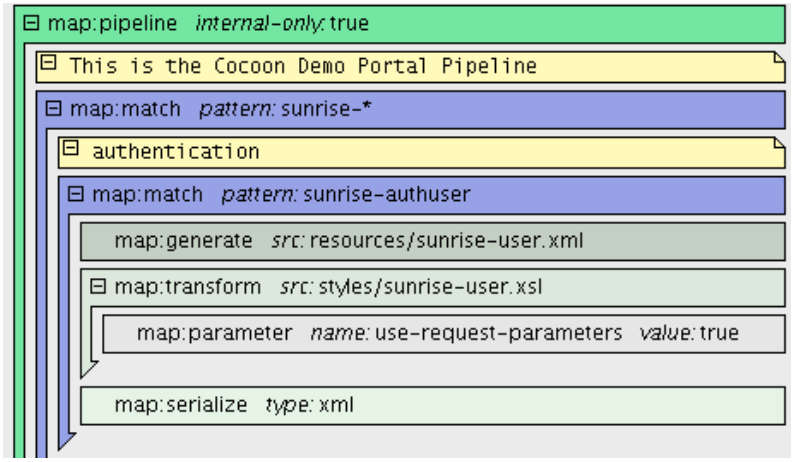

Figure 17 - Exemple de sitemap avec Pollo

Pour les utilisateurs de l'EDI Eclipse, certains modules Cocoon sont déjà disponibles, d'autres sont encore en phase de développement. Ils sont développés par la société Anyware Technologies (http://www.anyware-tech.com) qui est dirigée par Sylvain Wallez, membre haut placé de la Fondation Apache. Malgré l'aide précieuse qu'ils apportent, notamment pour l'écriture de sitemaps, il n'est pas encore possible de gérer un projet Cocoon de A à Z avec Eclipse.

Lepido, qui est actuellement en cours de développement, devrait en être capable. Les développeurs d'Anyware Technologies présentent Lepido comme un EDI pour Cocoon, basé sur Eclipse. Il devrait fournir beaucoup d'aide et inclura des fonctionnalités intéressantes : des éditeurs pour la plupart des dialectes XML, des éditeurs de formulaires, des debuggers, des éditeurs graphiques, etc. La sortie de Lepido est prévue courant 2006. En attendant, tout ce qu'il faut savoir sur ce futur EDI se trouve sur http://www.eclipse.org/proposals/eclipse-lepido.

1.6. Liaison avec les bases de données

PHP peut être utilisé avec de nombreux SGBD (MySQL, PostgreSQL, Oracle, Interbase, Microsoft SQL Server, Sybase, etc.) via l'installation de modules. Malheureusement, les hébergeurs ne proposent le plus souvent que MySQL, ce qui fait que celui-ci est presque devenu une obligation lorsque l'on veut utiliser des bases de données en PHP. Cette situation est regrettable, car MySQL n'est pas (et de loin) le SGBD le plus robuste qui existe. De plus, il n'implémente pas entièrement le standard SQL.

Cocoon a l'avantage de proposer un SGBD directement dans sa distribution. Il s'agit de HSQL (http://hsqldb.sourceforge.net), SGBDR Open Source en Java, sans grandes prétentions, mais largement suffisant pour de petites applications. Cocoon permet aussi, tout comme PHP, de travailler avec d'autres SGBD, en installant des modules au préalable.

1.7. Séparation des tâches

Comme nous le savons déjà, Cocoon a été bâti sur base du concept de séparation des tâches. Cela permet à chaque membre d'une équipe de travailler sur un aspect différent en ayant un minimum de dépendance avec les autres. C'est tout aussi utile pour un programmeur seul qui peut travailler sur un aspect à la fois, sans devoir tout mélanger (et finalement, ne plus s'y retrouver).

Le langage PHP ne traite pas du tout la séparation des tâches. Comme nous l'avons vu, le code PHP s'intègre à l'intérieur même de la page HTML. Avant l'apparition du CSS, il n'était pas rare d'avoir une page HTML dans laquelle se côtoyaient des instructions de formatage, des instructions de logique et des données. Si l'on rajoute les instructions de contrôle en PHP, cela nous donne quatre aspects différents mélangés dans une seule page. Ci-dessous, nous pouvons voir un exemple de code source rassemblant tous ces aspects :

```
<html>
<body bgcolor="
<!-- CONTROLE -->
<?php
 $timestamp = time();
 if($timestamp%2) echo "yellow";
 else echo "red";
```

```
?>
">
<center> <!-- FORMATAGE -->
<table border="0" width="650">
 <tr> <!-- LOGIQUE -->
 <td align="left" width="130">
  <font size="-1">EPFL-DI-LITH <!-- CONTENU --></font>
  <br/><font size="-1">IN- Ecublens</font>
  <br/><font size="-1">CH 1015 Lausanne</font>
  <br/><font size="-1">SWITZERLAND </font>
 </td>
 <td width="390">
  <center>
   <font size="+3">Computer Science Theory Laboratory (LITH)</font>
   <p>Director: Prof. Giovanni Coray</p>
  </center>
 </td>
 </tr>
</table>
</body>
</html>
```

Avec l'apparition de CSS, il est possible de gérer le formatage séparément, mais la logique, le contrôle et les données restent rassemblés.

Actuellement, beaucoup d'équipes de développeurs, se rendant compte de la carence de PHP concernant la séparation des tâches, se sont lancés dans le développement d'outils pour pallier ce manque. Le tableau ci-dessous dresse une liste non exhaustive des outils gratuits existants.

Table 4.1. Séparation des tâches en PHP

Nom	Description	URL
Ambivalence	Framework MVC pour PHP basé sur un framework MVC pour Java et J2EE nommé Maverick	http://amb.sourceforge.net
binarycloud	Framework pour PHP qui procure une infrastructure modulaire des applications	http://www.binarycloud.com
Horde Application Framework	Framework d'applications pour PHP faisant partie du projet Horde, connu principalement pour son Webmail (IMP)	http://www.horde.org/horde
InterJinn	Framework pour PHP	http://www.interjinn.com
Krysalis	Plate-forme permettant de transformer du XML en (X)HTML grâce à XSLT et PHP	http://www.kompletecms.com/products/krysalis
Mojavi	Framework MVC pour PHP	http://www.mojavi.org
php.MVC	Framework implémentant le design pattern MVC pour la création d'applications Web en PHP	http://www.phpmvc.net
Phrame	Framework MVC pour PHP	http://phrame.sourceforge.net
Seagull	Framework Orienté Objet pour PHP	http://seagull.phpkitchen.com
Smart	Framework pour PHP rapide et facile à utiliser (ne pas confondre avec Smarty)	http://developer.berlios.de/projects/smart
Smarty	Moteur de templates pour PHP. Il est intégré notamment dans Phrame, Mojavi et binarycloud	http://smarty.php.net

Nom	Description	URL
Studs	Framework MVC pour PHP créé sur base de Struts	http://www.mojavelinux.com/projects/studs
Wact	Boîte à outils pour la création d'applications Web	http://wact.sourceforge.net
Xaraya	Framework et CMS pour PHP	http://www.xaraya.com

Mise à part l'utilisation de ces outils, une solution pour produire du code PHP plus "propre" est sans aucun doute de laisser tomber la programmation procédurale pour passer à l'orienté objet, parfaitement intégré dans PHP 5. En effet, on bénéficie dans ce cas des nombreux avantages de ce paradigme.

1.8. Maintenance et réutilisabilité

Il a été prouvé par de nombreuses études, que la maintenance est la partie la plus importante du cycle de vie d'un logiciel. Il arrive souvent que 80% du coût total d'un logiciel soit dû à sa maintenance. Cet aspect n'est donc pas à négliger.

Une application en PHP demande énormément d'attention lors de sa création pour que l'on puisse la maintenir aisément par après. Il faut en effet s'imposer une politique de développement et laisser des portes ouvertes à des modifications futures. Malheureusement, cet idéal n'est pas souvent respecté, car il ralentit considérablement le développement. De plus, il est très difficile de prédire à l'avance ce qui pourrait venir s'ajouter ou être modifié à l'application existante.

Cocoon, grâce au concept de séparation des tâches, rend cette maintenance très facile. En PHP, la modification d'un aspect, par exemple le contrôle, nécessite la modification du fichier qui contient également des informations de contenu et de logique. Dans Cocoon, tout se trouve dans des fichiers séparés. De plus, un fichier PHP dans lequel tous les aspects sont mélangés est beaucoup plus difficile à comprendre après relecture.

En ce qui concerne la réutilisabilité, Cocoon s'en sort mieux, toujours grâce à la séparation des tâches. De nombreux éléments comme les feuilles de styles XSLT et certaines parties du sitemap peuvent être reprises telles quelles, ou doivent subir de minimes modifications. Pour illustrer ce fait, beaucoup d'éléments qui avaient déjà été développés pour l'exemple récapitulatif sur Cocoon (**Chapitre 5, “Exemple récapitulatif avec Cocoon”**) ont été repris pour le site d'eTransparence et inversement. Ce phénomène permet de réduire le temps de développement de manière importante au cours du temps.

Développer en PHP ne produit pas vraiment du code réutilisable, à cause de son mélange des aspects. Cependant, certains scripts et certaines fonctions, s'ils ont été définis de manière assez abstraite, peuvent très bien resservir pour des applications futures. L'avantage de PHP est qu'il possède une communauté gigantesque de développeurs. Dès lors, de nombreux sites proposent des scripts "tout fait" à installer dans sa propre application. PEAR (PHP Extension and Application Repository, http://pear.php.net) est un exemple de tel site. Il s'agit d'une collection de codes sources libres pour PHP structurée en packages, avec un système de distribution et de maintenance. PEAR a été créé en 1999 et est devenu depuis un sous-projet très actif de PHP. Il est même considéré comme la prochaine révolution de PHP.

2. Conclusion

PHP et Cocoon possèdent chacun leurs qualités et leurs défauts. Aucun n'est à jeter, leurs champs d'application sont tout simplement différents.

La création d'une application Web avec le langage PHP est recommandée pour les personnes travaillant seules et désirant obtenir un résultat rapidement. Ce langage est également à conseiller aux personnes ayant déjà une bonne expérience de la création de sites Web statiques en HTML et désirant dynamiser leurs travaux. En effet, la quantité de supports disponibles sur Internet ou en librairie permet de maîtriser assez vite le langage. La gratuité de PHP est une de ses grandes forces par rapport à ASP (Microsoft) et ColdFusion (Macromedia), pour ne citer qu'eux. PHP se prête très bien à la création de sites personnels et de portails. Cependant, PHP s'impose assez bien en entreprise de nos jours. En effet, de gigantesques multinationales comme Google, Yahoo!, Wanadoo, Tiscali, Lycos Europe, Time Warner/AOL, etc. utilisent PHP pour des fonctions spécifiques ou la totalité de leurs services.

Cocoon, quant à lui, se prête plus au monde de l'entreprise. Le travail en équipe et la maintenance se voient facilités par l'utilisation du framework d'Apache. Aussi étonnant que cela puisse paraître, le point faible de Cocoon en entreprise est le fait qu'il soit totalement gratuit et Open Source ! En effet, les entreprises s'inquiètent beaucoup de la qualité et de la fiabilité des logiciels produits en mode collaboratif. De plus, les entreprises aiment bien se retourner contre "quelqu'un" lorsqu'une défaillance apparaît ce qui est impossible si on utilise des logiciels Open Source. Néanmoins, les mentalités commencent à changer et les entreprises se rendent enfin compte des économies gigantesques qu'elles pourraient réaliser en se tournant vers le libre.

Pour les développeurs isolés et les PME, le manque de documentation et le fait qu'aucun hébergement gratuit ne soit encore disponible ne place pas Cocoon en position de force. Pour les développeurs non-anglophones comme nous, le fait que tout ce qui tourne autour de Cocoon soit décliné uniquement dans la langue de Shakespeare est un obstacle supplémentaire à sa compréhension et à son utilisation.

Dans l'état actuel des choses, Cocoon, malgré ses atouts indéniables, est encore un peu jeune pour s'imposer face à l'ogre PHP qui a fait son trou depuis plus de dix années. L'évolution et l'élargissement de la communauté de Cocoon dans le monde entier devraient lui permettre de gagner du terrain dans le futur et de s'imposer aussi chez les particuliers.

Chapitre 5. Exemple récapitulatif avec Cocoon

1. Mise en situation

Cette section reprend la plupart des concepts étudiés auparavant dans un exemple récapitulatif très simple.

La mini application comportera les pages suivantes :

- Une page d'accueil qui affiche un message de bienvenue.
- Une page historique qui donne les dates de commencement et d'achèvement de l'application.
- Une page musique qui permet de télécharger des fichiers musicaux au format MP3.
- Une page statistique qui affiche un graphique des visites sur l'application.

Elle sera capable de :

- Présenter le contenu de la page d'accueil et de la page historique dans les formats HTML (affichage par défaut), XML (pour voir le code source) et PDF (pour télécharger la page et éventuellement l'imprimer).
- Présenter le contenu de la page d'accueil en français, italien, espagnol et anglais.
- Inclure un menu de navigation sur chaque page. Cela permettra, à partir d'une page, d'accéder directement à l'une des trois autres.
- Lister le contenu d'un répertoire contenant des fichiers MP3 et les afficher sous la forme d'un tableau en reprenant les caractéristiques du fichier et en faisant un lien permettant de télécharger celui-ci. Le tableau ci-dessous nous donne un aperçu de l'output possible.

  Table 5.1. Aperçu du tableau de MP3

Nom du fichier	Dernière modification	Taille	Fréquence	Mode	Bitrate
Beck - Loser.mp3	18/03/05	3,4 Mo	44.1 Khz	Stéréo	128 Kbps
Metallica - Frantic.mp3	10/12/04	1,8 Mo	44.1 Khz	Mono	64 Kbps
U2 - Vertigo.mp3	02/05/05	4,1 Mo	44.1 Khz	Stéréo	192 Kbps

- Afficher un graphique de statistiques à partir de données numériques contenues dans un fichier XML. Ce graphique est un histogramme où la hauteur des bâtons représente le nombre de visites par mois sur l'application. Le fichier sur lequel nous nous basons contient le nombre de visites correspondant à chaque mois passé. Ce fichier est statique car le calcul du nombre de visiteurs n'est pas pris en charge par notre application. Le dessin ci-dessous nous donne un aperçu de l'output possible.

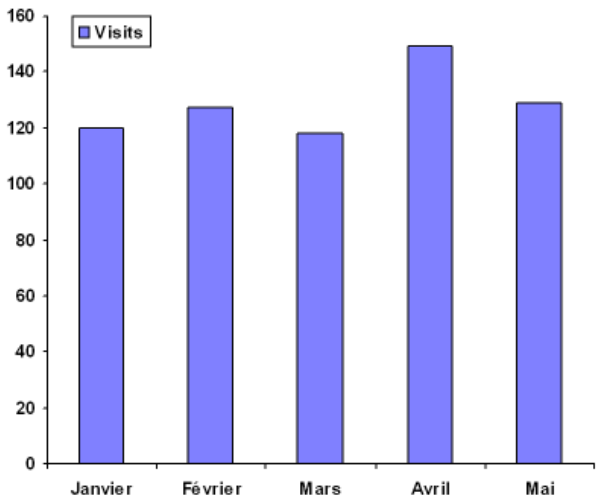

Figure 18 - Aperçu de la page statistique

2. Répertoires et canevas de fichiers

Voici comment sont organisés les répertoires et les fichiers de cet exemple :

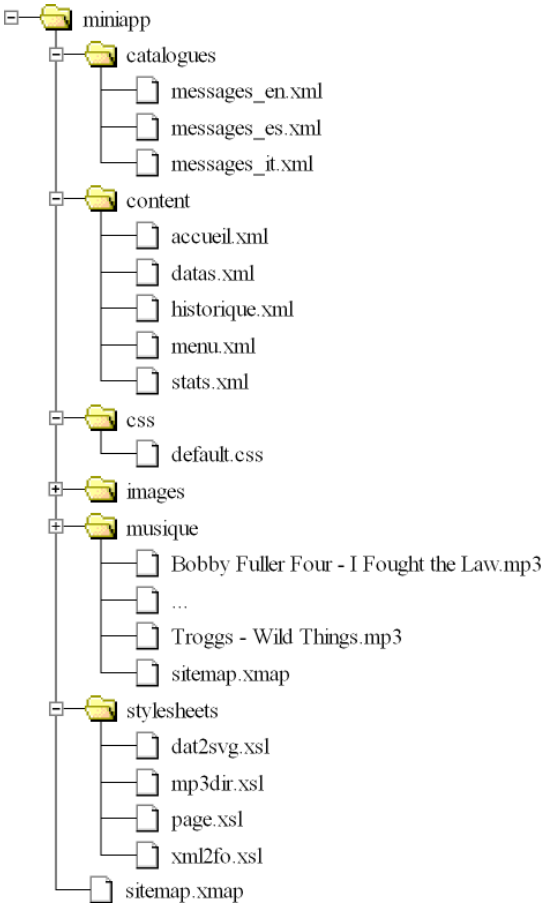

Figure 19 - Arborescence de fichiers de la mini application

Le répertoire `catalogues` contient les catalogues de langues qui seront utiles pour l'internationalisation, `content` contient les fichiers XML de contenu, `css` contient les feuilles de style CSS, `images` contient les images, `musique` contient des fichiers MP3 et `stylesheets` contient les feuilles de style XSLT. Le fichier `sitemap.xmap` est le sitemap général (racine) de l'application.

Pour les fichiers XML de contenu (du répertoire `content`), nous avons déterminé la structure suivante :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<page id="nomdufichier">
 <title/>
 <views>
  <xml/><pdf/>
 </views>
 <content>
  <para/>
  <para/>
  <!-- ... -->
 </content>
</page>
```

L'élément racine est `<page/>`, la valeur de son attribut `id` est le nom du fichier sans son extension (par exemple : `accueil` si le fichier est `accueil.xml`). `<title/>` contient le titre, `<views/>` détermine dans quels autres formats que le HTML sera disponible la page. Les enfants possibles de `<views/>` sont `<xml/>` et `<pdf/>`. L'élément `<content/>` renferme le contenu proprement dit de la page. Il se décompose en paragraphes (éléments `<para/>`).

A titre d'exemple, voici le contenu de la page `historique.xml` :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!-------------------------->
<!----- historique.xml ----->
<!-------------------------->
<page id="historique">
 <title>Historique</title>
 <views>
  <xml/><pdf/>
 </views>
 <content>
  <para>
   Cette mini application a été initiée le 29 avril 2005 et a été terminée le 1er
   mai 2005.
  </para>
 </content>
</page>
```

Le menu, quant à lui, se présente de la manière suivante :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<menu>
 <menuitem>
  <label/>
  <link/>
 </menuitem>
 <!-- ... -->
</menu>
```

L'élément racine est `<menu/>`. Chaque `<menuitem/>` représente un élément du menu et possède un titre (élément `<label/>`) et un lien (élément `<link/>`).

Le contenu de notre menu (`menu.xml`) est le suivant :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!-------------------->
<!----- menu.xml ----->
<!-------------------->
<menu>
 <menuitem>
  <label>Accueil</label>
  <link>accueil</link>
 </menuitem>
 <menuitem>
  <label>Historique</label>
  <link>historique</link>
 </menuitem>
 <menuitem>
   <label>Musique</label>
   <link>musique/liste</link>
 </menuitem>
 <menuitem>
   <label>Statistiques</label>
   <link>stats</link>
 </menuitem>
</menu>
```

3. Affichage d'une page HTML

Pour afficher une page au format HTML, nous devons transformer notre page XML avec l'aide d'une feuille de style XSLT.

Dans le sitemap, cela va se présenter de cette manière :

```
<map:match pattern="*.html">
  <map:aggregate element="document">
    <map:part src="content/menu.xml"/>
    <map:part src="content/{1}.xml"/>
  </map:aggregate>
  <map:transform src="stylesheets/page.xsl"/>
  <map:serialize type="html"/>
</map:match>
```

Lorsqu'on va matcher avec une URI se terminant par `.html`, on va générer un flux qui sera l'agrégation de `menu.xml` et du fichier XML correspondant à la requête, englobés dans la nouvelle racine `<document/>`. Ensuite, on applique la feuille de style `page.xsl` pour transformer le XML en HTML. Pour terminer, on sérialise au format HTML.

Supposons que la requête soit `historique.html`, on va agréger les fichiers `menu.xml` et `historique.xml` avec `<document/>` comme nouvel élément racine ce qui va donner :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<document>

 <menu>
  <menuitem>
   <label>Accueil</label>
   <link>accueil</link>
  </menuitem>
  <menuitem>
   <label>Historique</label>
   <link>historique</link>
  </menuitem>
  <menuitem>
    <label>Musique</label>
    <link>musique/liste</link>
  </menuitem>
  <menuitem>
    <label>Statistiques</label>
    <link>stats</link>
  </menuitem>
 </menu>

 <page id="historique">
  <title>Historique</title>
  <views>
   <xml/><pdf/>
  </views>
  <content>
   <para>
    Cette mini application a été initiée le 29 avril 2005
    et a été terminée le 1er mai 2005.
   </para>
  </content>
 </page>

</document>
```

Ensuite, on applique la feuille de style `page.xsl` :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!-------------------->
<!----- page.xsl ----->
<!-------------------->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

 <xsl:template match="/">
  <html>
   <head>
    <link rel="stylesheet" type="text/css" href="css/default.css"/>
    <title>MiniApp | <xsl:value-of select="//page/title"/></title>
   </head>
   <body>
    <h1><xsl:value-of select="//page/title"/></h1>
    <xsl:apply-templates select="//views"/>
    <xsl:apply-templates select="//menu"/>
    <xsl:apply-templates select="//page/content"/>
   </body>
  </html>
 </xsl:template>

 <!-- création d'un bloc pour les icônes -->
```

```
 <xsl:template match="views">
  <div id="views">
   <xsl:apply-templates/>
  </div>
 </xsl:template>

 <!-- affichage de l'icône XML -->
 <xsl:template match="xml">
   <span class="view">
    <a>
     <xsl:attribute name="href">
     <xsl:value-of select="//page/@id"/>.xml</xsl:attribute>
      <img src="images/xml.gif" alt="Version XML"/>
    </a>
   </span>
 </xsl:template>

 <!-- affichage de l'icône PDF -->
 <xsl:template match="pdf">
   <span class="view">
    <a>
     <xsl:attribute name="href">
     <xsl:value-of select="//page/@id"/>.pdf</xsl:attribute>
      <img src="images/pdf.gif" alt="Version PDF"/>
    </a>
   </span>
 </xsl:template>

 <!-- création du menu sous la forme d'une liste -->
 <xsl:template match="menu">
  <ul id="menu">
   <xsl:apply-templates/>
  </ul>
 </xsl:template>

 <!-- création et affichage d'un élément du menu -->
 <xsl:template match="menuitem">
  <li class="menuitem">
   <a>
   <xsl:attribute name="href">
    /cocoon/miniapp/<xsl:apply-templates select="link"/>.html
   </xsl:attribute>
    <xsl:apply-templates select="label"/>
   </a>
  </li>
 </xsl:template>

 <!-- création d'un bloc pour le contenu -->
 <xsl:template match="content">
  <div id="text">
   <xsl:apply-templates/>
  </div>
 </xsl:template>

 <!-- création et affichage d'un paragraphe -->
 <xsl:template match="para">
  <p>
   <xsl:apply-templates/>
  </p>
 </xsl:template>

</xsl:stylesheet>
```

Nous obtenons l'output HTML suivant :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
                      "http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
  <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <link href="css/default.css" type="text/css" rel="stylesheet">
  <title>MiniApp | Historique</title>
 </head>
 <body>
  <h1>Historique</h1>

  <div id="views">
   <span class="view">
    <a href="historique.xml"><img alt="Version XML" src="images/xml.gif"></a>
   </span>
   <span class="view">
    <a href="historique.pdf"><img alt="Version PDF" src="images/pdf.gif"></a>
   </span>
  </div>

  <ul id="menu">
   <li class="menuitem">
    <a href="/cocoon/miniapp/accueil.html">Accueil</a>
   </li>
   <li class="menuitem">
    <a href="/cocoon/miniapp/historique.html">Historique</a>
   </li>
   <li class="menuitem">
    <a href="/cocoon/miniapp/musique/liste.html">Musique</a>
   </li>
   <li class="menuitem">
    <a href="/cocoon/miniapp/stats.html">Statistiques</a>
   </li>
  </ul>

  <div id="text">
   <p>
    Cette mini application a &eacute;t&eacute; initi&eacute;e le 29 avril 2005
    et a &eacute;t&eacute; termin&eacute;e le 1er mai 2005.
   </p>
  </div>

 </body>
</html>
```

Ce qui donne à l'écran :

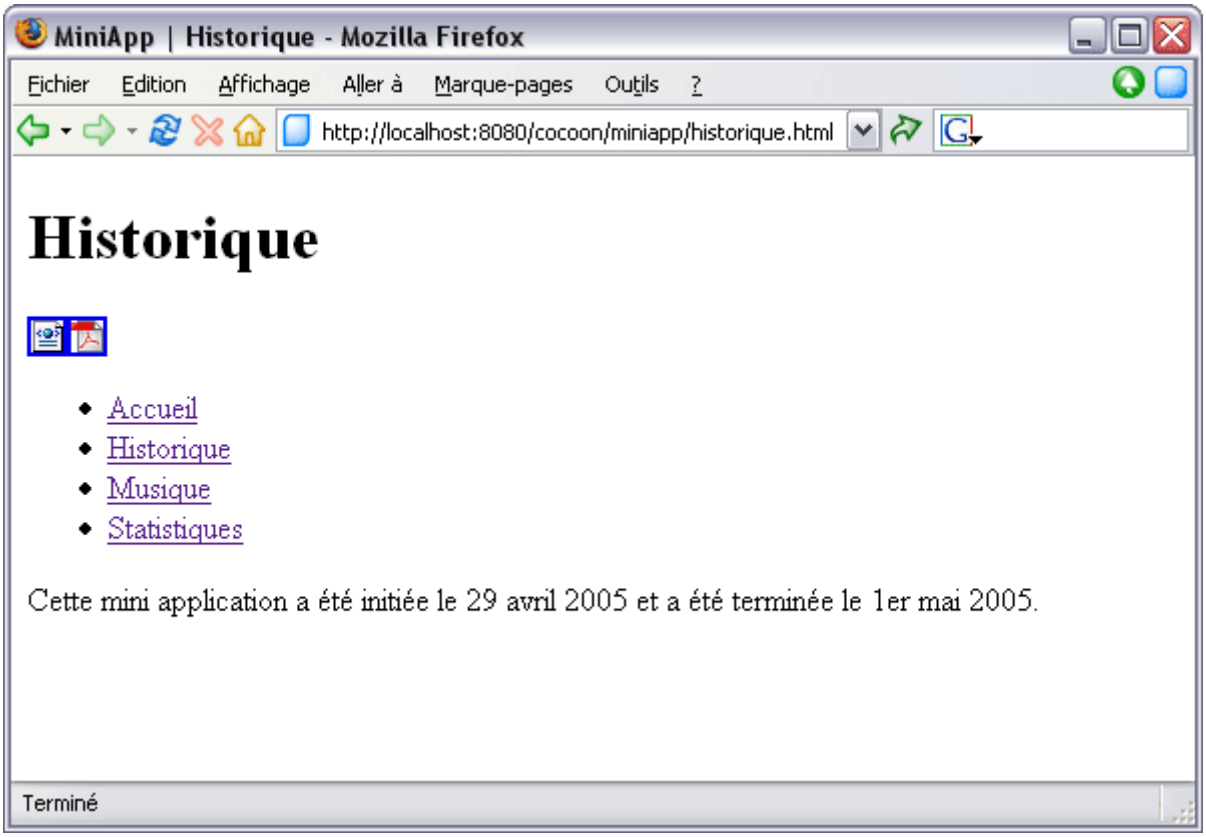

Figure 20 - Aperçu de la page "Historique"

Appliquons la feuille de style CSS `default.css` à ce fichier HTML :

```
<!----------------------->
<!----- default.css ----->
<!----------------------->

body
{
 background-image:url('../images/bird.gif'); <!-- image d'arriere plan -->
 background-repeat:no-repeat; <!-- non repetition de l'image d'arriere plan -->
 font-family:sans-serif; <!-- police de caractere -->
 font-size:10pt; <!-- taille de la police -->
}
h1
{
 border:1px solid black; <!-- bordure du bloc -->
 padding:5px; <!-- espacement interieur -->
}
a:hover
{
 color:orange; <!-- couleur du texte -->
 text-decoration:none; <!-- pas de soulignement au texte -->
}
a
{
 color:navy; <!-- couleur du texte -->
}
```

```
img
{
 border:0px; <!-- pas de bordures autour des images -->
}
span.view
{
 padding:3px; <!-- espacement interieur -->
}
<!-- STYLES POUR LA LISTE DE MP3 -->
table#mp3table
{
 border:1px solid black; <!-- bordure du bloc -->
}
th
{
 background-color:#DDDDDD; <!-- couleur d'arriere plan -->
 border:1px solid orange; <!-- bordure du bloc -->
 padding:3px; <!-- espacement interieur -->
}
td
{
 border:1px solid #DDDDDD; <!-- bordure du bloc -->
 padding:3px; <!-- espacement interieur -->
}
```

Nous obtenons une page plus élégante :

Figure 21 - Aperçu de la page "Historique" avec le formatage CSS

Schématiquement, voici comment se déroule le traitement de la requête `historique.html` :

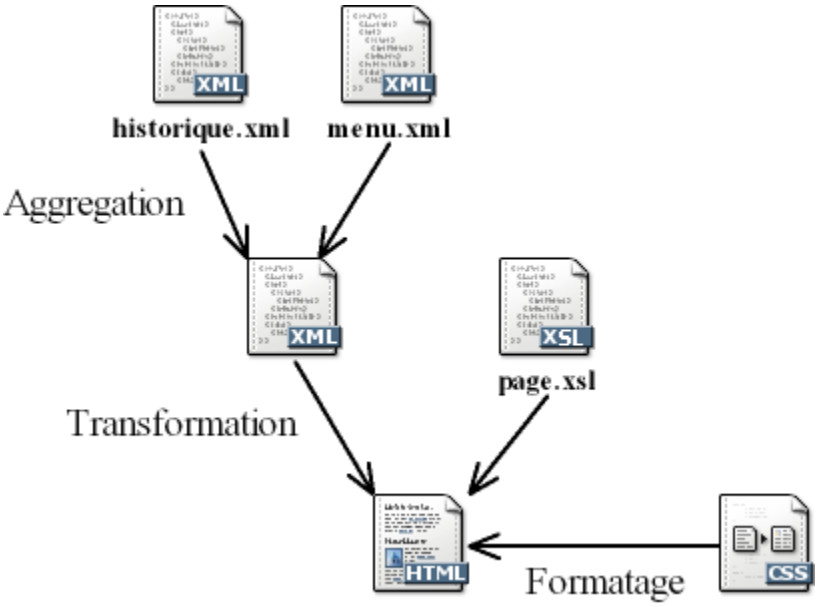

Figure 22 - Traitement de la requête "historique.html"

4. Affichage d'une page PDF

Par rapport à l'affichage d'une page au format HTML, seuls la feuille de style XSLT et le serializer changent :

```
<map:match pattern="*.pdf">
  <map:generate src="content/{1}.xml"/>
  <map:transform src="stylesheets/xml2fo.xsl"/>
  <map:serialize type="fo2pdf"/>
</map:match>
```

Le menu n'a pas besoin d'être inclus dans le fichier PDF.

Le contenu de la feuille de style `xml2fo.xsl` est le suivant :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!----------------------->
<!----- xml2fo.xsl ----->
<!----------------------->
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">

  <!-- informations générales -->
  <xsl:template match="/">
   <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

    <fo:layout-master-set>
     <fo:simple-page-master master-name="page"
                  page-height="29.7cm"
                  page-width="21cm"
                  margin-top="1cm"
                  margin-bottom="2cm"
```

```
                  margin-left="2.5cm"
                  margin-right="2.5cm">
      <fo:region-before extent="3cm"/>
      <fo:region-body margin-top="3cm"/>
    </fo:simple-page-master>
    <fo:page-sequence-master master-name="all">
      <fo:repeatable-page-master-alternatives>
       <fo:conditional-page-master-reference master-reference="page"
           page-position="first"/>
      </fo:repeatable-page-master-alternatives>
    </fo:page-sequence-master>
   </fo:layout-master-set>

   <fo:page-sequence master-reference="all">
     <fo:flow flow-name="xsl-region-body">
       <xsl:apply-templates/>
     </fo:flow>
   </fo:page-sequence>

  </fo:root>
 </xsl:template>

 <!-- affichage du titre -->
 <xsl:template match="title">
   <fo:block font-size="36pt" text-align="center">
    <xsl:value-of select="."/>
   </fo:block>
 </xsl:template>

 <!-- création et affichage d'un paragraphe -->
 <xsl:template match="para">
   <fo:block font-size="12pt" space-before.optimum="12pt" text-align="left">
    <xsl:apply-templates/>
   </fo:block>
 </xsl:template>

</xsl:stylesheet>
```

Cette feuille de style va transformer le XML en XSL-FO. Le serializer *fo2pdf* va se charger de transformer le XSL-FO en PDF ce qui donnera à l'écran :

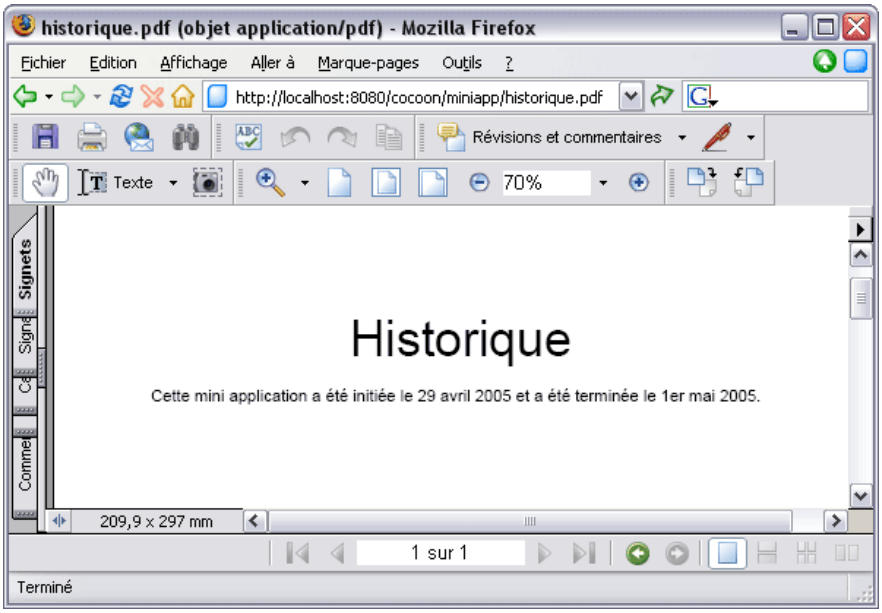

Figure 23 - La page "Historique" au format PDF

Schématiquement, voici comment se déroule le traitement de la requête `historique.pdf` :

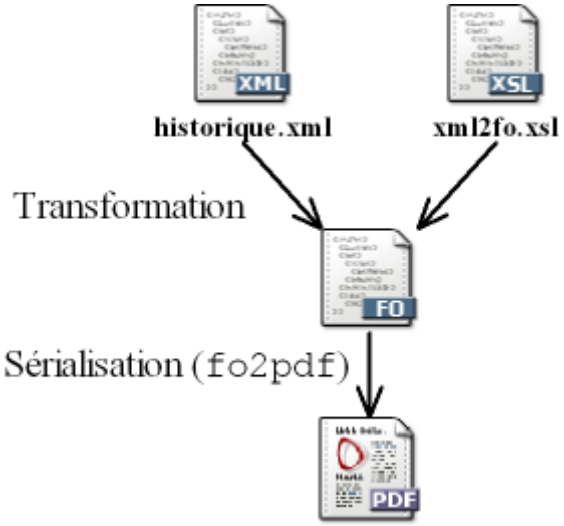

Figure 24 - Traitement de la requête "historique.pdf"

5. Affichage d'une page XML

Pour afficher une page au format XML, il suffit de ne pas appliquer de transformation(s) au flux généré et de sérialiser en XML :

```
<map:match pattern="*.xml">
  <map:generate src="content/{1}.xml"/>
  <map:serialize type="xml"/>
</map:match>
```

Une meilleure solution est d'utiliser un reader, puisque nous ne "touchons" pas au fichier de départ (pas de transformation(s)) :

```
<map:match pattern="*.xml">
  <map:read src="content/{1}.xml" mime-type="text/xml"/>
</map:match>
```

Le résultat obtenu est le suivant :

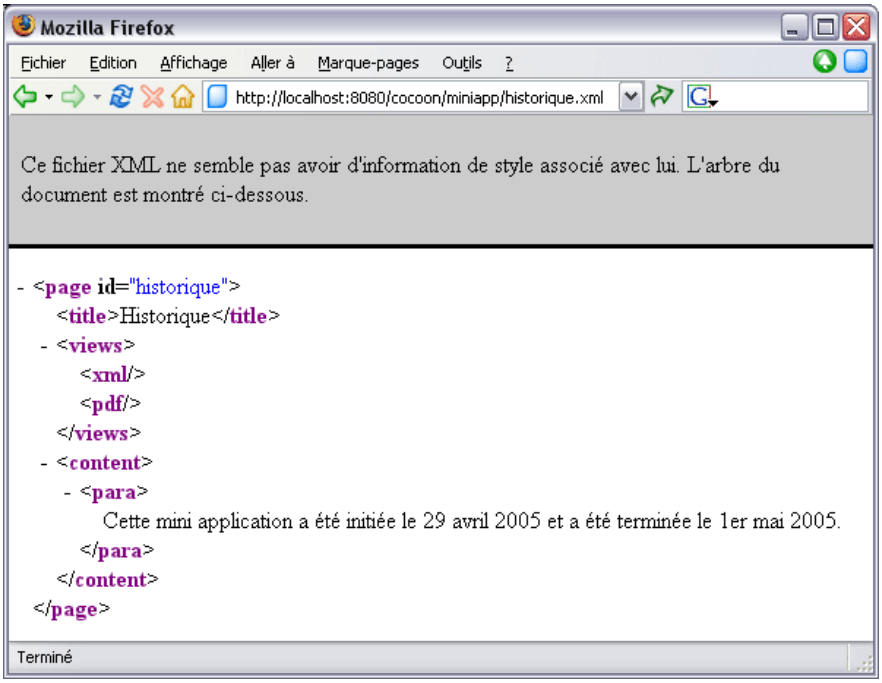

Figure 25 - La page "Historique" au format XML

6. Internationalisation (I18N)

L'internationalisation (i18n) est un concept permettant d'afficher une page en plusieurs langues, grâce à l'utilisation de catalogues.

Nous voudrions que le contenu de la page d'accueil s'affiche en anglais, en espagnol et en italien, par un simple click sur le drapeau du pays correspondant.

Le principe de l'internationalisation avec Cocoon est assez simple : nous devons appliquer des clés sur les parties (paragraphes, titres, etc.) que nous aimerions traduire. A coté de cela, nous créons un catalogue pour chaque langue qui associe une traduction à chaque clé.

Notre fichier `accueil.xml` se présente maintenant de cette façon :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!------------------------>
<!----- accueil.xml ----->
<!------------------------>
<page id="accueil" xmlns:i18n="http://apache.org/cocoon/i18n/2.1">
 <title><i18n:text i18n:key="accueil_title">Bienvenue !</i18n:text></title>
 <views>
  <xml/><pdf/>
 </views>
 <content>
  <para>
   <i18n:text i18n:key="accueil_para1">
    Bienvenue sur cette application réalisée avec Cocoon.
   </i18n:text>
  </para>
```

```
  <!-- drapeaux pour le choix de la langue -->
  <para>
   <link href="accueil-en.html"><flag id="en"/></link>
   <link href="accueil-es.html"><flag id="es"/></link>
   <link href="accueil.html"><flag id="fr"/></link>
   <link href="accueil-it.html"><flag id="it"/></link>
  </para>
 </content>
</page>
```

Une partie à traduire se présente de la façon suivante : `<i18n:text i18n:key="identifiant">Texte par défaut</i18n:text>`. La clé est la valeur de l'attribut `i18n:key`. Notons que l'utilisation de cet attribut n'est pas obligatoire, auquel cas la clé est le contenu de `<i18n:text/>` (ici ce serait "Texte par défaut").

Un catalogue est un fichier XML contenant une traduction pour chaque clé.

Voici par exemple le catalogue espagnol `messages_es.xml` (ce nom n'est pas choisi au hasard, nous verrons pourquoi un peu plus tard) :

```
<?xml version="1.0" ?>
<!--------------------------->
<!----- messages_es.xml ----->
<!--------------------------->
<catalogue xml:lang="locale">
  <message key="accueil_title">Bienvenido !</message>
  <message key="accueil_para1">Bienvenido en este sitio creado con
Cocoon.</message>
  <message key="menu_accueil">Principal</message>
  <message key="menu_historique">Historia</message>
  <message key="menu_musique">Música</message>
  <message key="menu_stats">Estadística</message>
</catalogue>
```

Notons que nous avons également traduit les éléments du menu.

A présent, pour traduire notre page d'accueil, nous devons appliquer le transformer d'internationalisation.

Dans un premier temps, nous devons déclarer le composant dans le sitemap :

```
<map:components>

<map:transformers default="xslt">
 <map:transformer name="i18n"
src="org.apache.cocoon.transformation.I18nTransformer">
     <catalogues default="messages">
       <catalogue id="messages" name="messages" location="catalogues"/>
     </catalogues
     <untranslated-text>texte non traduit</untranslated-text>
     <cache-at-startup>true</cache-at-startup>
 </map:transformer>
</map:transformers>

<!-- ... -->

</map:components>
```

Dans l'élément `<map:transformers/>`, nous devons déterminer le transformer par défaut (`xslt`). A l'intérieur de cet élément, nous déclarons le transformer d'internationalisation qui porte le nom `i18n` et qui est associé à la classe Java `org.apache.cocoon.transformation.I18nTransformer`. Nous déclarons ensuite le groupe de catalogues grâce à l'élément `<catalogue/>`. L'attribut `id` est l'identifiant du groupe, `messages` est le préfixe des fichiers du groupe et `location` est l'emplacement des fichiers. Il est tout à fait possible de déclarer plusieurs groupes de catalogues.

La dernière étape est de créer le pipeline qui va permettre d'utiliser le transformer d'internationalisation précédemment déclaré :

```
<map:match pattern="accueil-*.html">
  <map:aggregate element="document">
    <map:part src="content/menu.xml"/>
    <map:part src="content/accueil.xml"/>
  </map:aggregate>
  <map:transform type="i18n">
    <map:parameter name="locale" value="{1}"/>
  </map:transform>
  <map:transform src="stylesheets/page.xsl"/>
  <map:serialize type="html"/>
</map:match>
```

Supposons que notre requête est `accueil-it.html`. Après l'agrégation du menu et de la page d'accueil, nous appliquons le transformer d'internationalisation. Celui-ci va chercher les traductions à l'emplacement défini par la valeur de l'attribut `location` de `<catalogue/>`, dans le fichier dont le nom est défini par la valeur de l'attribut `name` de `<catalogue/>` + le symbole underscore + la valeur de l'attribut `value` de `<parameter/>` + l'extension du fichier. Dans notre cas, cela va donner : `messages` + _ + `it` + `.xml` c'est-à-dire `messages_it.xml`.

Voici le résultat que nous allons obtenir :

Figure 26 - La page d'accueil en italien

Si nous faisons une requête sur une langue qui n'existe pas, par exemple avec l'URI `accueil-pl.html`, nous n'allons pas avoir d'erreur, la page s'affichera simplement avec les valeurs par défaut, donc en français.

Schématiquement, voici comment se déroule le traitement de la requête `accueil-it.html` :

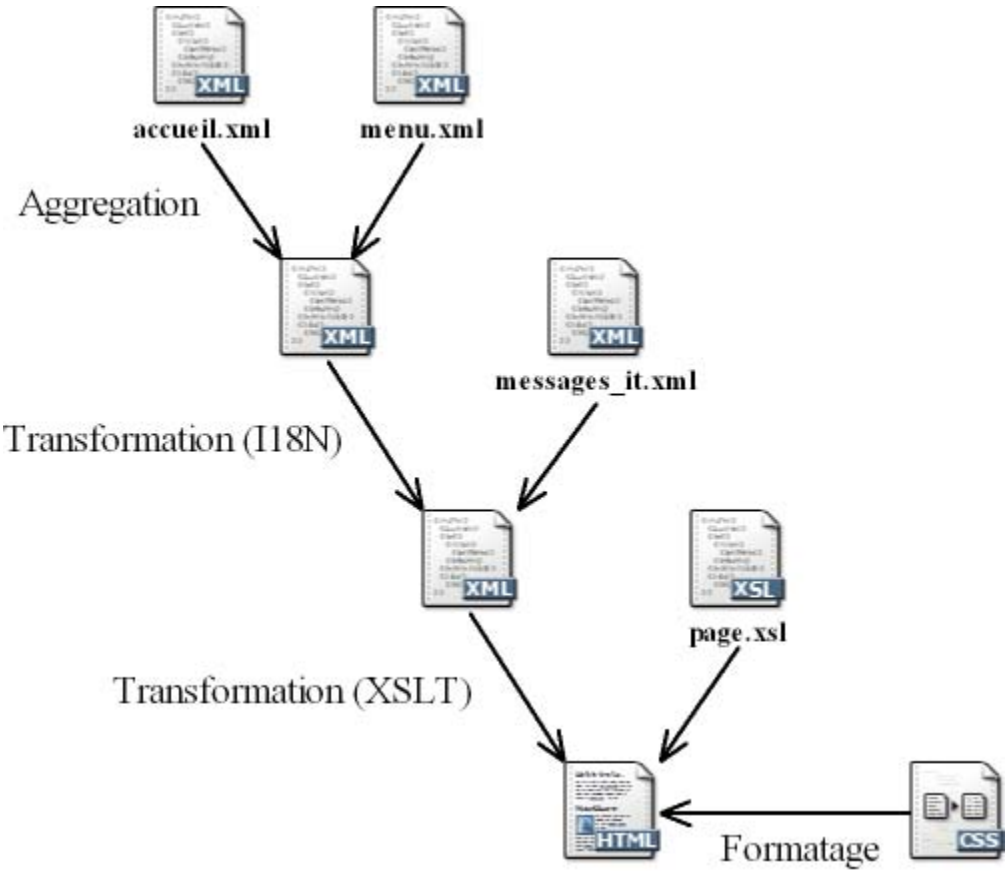

Figure 27 - Traitement de la requête "accueil-it.html"

7. Répertoire musical

Une section du site liste tous les fichiers MP3 d'un répertoire et en donne les caractéristiques. Nous allons réaliser cela avec le *Mp3directory Generator*. Le dossier que nous allons lister est `/musique`.

Pour atteindre la page "Musique", l'URI est `musique/liste.html`. Dans le sitemap, nous allons matcher avec le pipeline suivant :

```
<map:match pattern="musique/*">
  <map:mount src="musique/sitemap.xmap" uri-prefix="musique"/>
</map:match>
```

Cela aura pour effet de monter le sous-sitemap situé à l'emplacement `musique/sitemap.xmap` en supprimant le préfixe `musique/` de la requête.

Voici le contenu de ce sitemap :

```
<!-- SITEMAP MUSIQUE -->

<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">

<!-- =========================== Pipelines ================================== -->

 <map:pipelines>

   <map:pipeline>

    <map:match pattern="">
     <map:redirect-to uri="liste.html"/>
    </map:match>

    <map:match pattern="*.mp3">
     <map:read src="{1}.mp3" mime-type="audio/x-mp3"/>
    </map:match>

    <map:match pattern="liste.html">
     <map:aggregate element="document">
        <map:part src="../content/menu.xml"/>
        <map:part src="cocoon:/liste.xml"/>
     </map:aggregate>
     <map:transform src="../stylesheets/mp3dir.xsl"/>
     <map:serialize type="html"/>
    </map:match>

    <map:match pattern="liste.xml">
     <map:generate type="mp3directory" src="."/>
     <map:serialize type="xml"/>
    </map:match>

   </map:pipeline>

 </map:pipelines>

</map:sitemap>
```

Nous allons matcher avec le troisième pipeline :

```
<map:match pattern="liste.html">
  <map:aggregate element="document">
    <map:part src="../content/menu.xml"/>
    <map:part src="cocoon:/liste.xml"/>
  </map:aggregate>
  <map:transform src="../stylesheets/mp3dir.xsl"/>
  <map:serialize type="html"/>
</map:match>
```

Il s'agit d'un très bon exemple d'une notion dont nous n'avions pas encore parlé : les pipelines imbriqués.

Nous allons générer un flux qui sera l'agrégation du menu avec le résultat du pipeline qui matche la requête `liste.xml`. Le protocole `cocoon:/` signifie que l'on va rechercher un pipeline qui matche dans le sitemap courant. Le protocole `cocoon://` existe aussi, il signifie que l'on va rechercher un pipeline qui matche dans le sitemap racine.

Voici le sous-pipeline qui va être exécuté par la requête `cocoon:/liste.xml` :

```
<map:match pattern="liste.xml">
  <map:generate type="mp3directory" src="."/>
  <map:serialize type="xml"/>
</map:match>
```

Et voici le résultat qu'il retourne :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<dir:directory name="." lastModified="1114862636187" date="30/04/05 14:03" size="0"
     sort="name" reverse="false" requested="true"
     xmlns:dir="http://apache.org/cocoon/directory/2.0">
 <dir:file name="Bobby Fuller Four - I Fought the Law.mp3"
      lastModified="1102674226000" date="10/12/04 11:23" size="2237753"
      frequency="44.1" mode="Joint stereo" bitrate="128"/>
 <dir:file name="Canned Heat - On the Road Again.mp3" lastModified="1102674212000"
      date="10/12/04 11:23" size="3290593" frequency="44.1" mode="Joint stereo"
      bitrate="128"/>
 <dir:file name="Count Five - Psychotic Reaction.mp3" lastModified="1102674204000"
      date="10/12/04 11:23" size="2935327" frequency="44.1" mode="Joint stereo"
      bitrate="128"/>
 <!-- ... -->
</dir:directory>
```

Après l'agrégation du résultat ci-dessus et de `menu.xml`, nous allons utiliser la feuille de style XSLT `mp3dir.xsl` qui est une adaptation de la feuille de style `page.xsl` :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!---------------------->
<!----- mp3dir.xsl ----->
<!---------------------->
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:dir=http://apache.org/cocoon/directory/2.0
   exclude-result-prefixes="dir">

<xsl:import href="page.xsl"/> <!-- import des templates de page.xsl -->

<xsl:template match="/">
  <html>
   <head>
    <link rel="stylesheet" type="text/css" href="../css/default.css"/>
    <title>MiniApp | Musique</title>
   </head>
   <body>
    <h1>Musique</h1>
    <div id="views">
     <span class="view">
      <a href="liste.xml">
       <img src="../images/xml.gif" alt="Version XML"/>
      </a>
     </span>
    </div>
    <xsl:apply-templates select="//menu"/>
    <table id="mp3table">
     <tr>
      <th>Artiste - Titre</th>
      <th>Dernière modification</th>
      <th>Taille (o)</th>
      <th>Fréquence</th>
      <th>Mode</th>
```

```
      <th>Bitrate</th>
     </tr>
     <xsl:apply-templates select="//dir:file"/>
    </table>
   </body>
  </html>
</xsl:template>

<!-- nous affichons tous les fichiers sauf sitemap.xmap -->
<xsl:template match="dir:file[@name!='sitemap.xmap']">
 <tr>
  <td>
   <a>
   <xsl:attribute name="href">
    <xsl:value-of select="@name"/>
   </xsl:attribute>
   <xsl:value-of select="@name"/>
   </a>
  </td>
  <td><xsl:value-of select="@date"/></td>
  <td><xsl:value-of select="@size"/></td>
  <td><xsl:value-of select="@frequency"/> Khz</td>
  <td><xsl:value-of select="@mode"/></td>
  <td><xsl:value-of select="@bitrate"/> Kbps</td>
 </tr>
</xsl:template>

</xsl:stylesheet>
```

Pour terminer, nous sérialisons au format HTML.

Voici le résultat à l'écran :

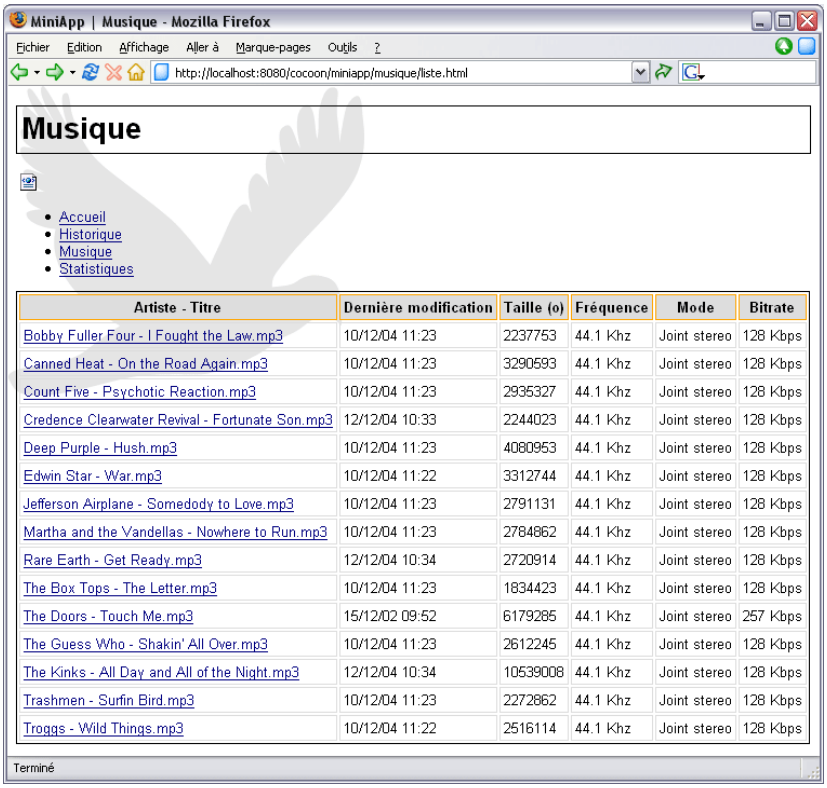

Figure 28 - Aperçu de la page "Musique"

8. Statistiques

Dans cette partie, nous allons voir comment transformer dynamiquement des données numériques en graphiques élaborés. Pour faire cela, nous allons utiliser SVG.

SVG est un format basé sur XML permettant de réaliser des graphiques vectoriels en deux dimensions. Les coordonnées, dimensions et structures des objets vectoriels sont indiquées sous forme numérique dans le document XML. On le présente comme un concurrent sérieux à Flash de Macromedia.

Voici ci-dessous un exemple de code source SVG. Nous voulons dessiner un carré vert muni d'un bord rouge.

```
<?xml version="1.0" ?>
<!---------------------->
<!----- carre.svg ----->
<!---------------------->
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/PR-SVG-20010719/DTD/svg10.dtd">
<svg width="300" height="300" xmlns="http://www.w3.org/2000/svg">
 <title>Exemple de SVG</title>
 <desc>Carré vert bordé de rouge</desc>
 <rect x="50" y="50" width="200" height="200"
       style="fill:#1AB11A;stroke:#FE0000;stroke-width:3"/>
</svg>
```

Chaque page SVG comporte la racine `<svg/>`. On y indique la largeur (`width`) et la hauteur (`height`) de la future image. On peut donner un titre (`<title/>`) à notre image et une description (`<desc/>`). Ensuite, nous dessinons le rectangle en donnant les coordonnées de son bord supérieur gauche (`x` et `y`) et en donnant sa largeur et sa hauteur. L'attribut `style` nous permet de définir la couleur de remplissage (`fill`), la couleur du contour (`stroke`) et la largeur de ce contour (`stroke-width`).

Voici le résultat que nous obtenons dans le navigateur :

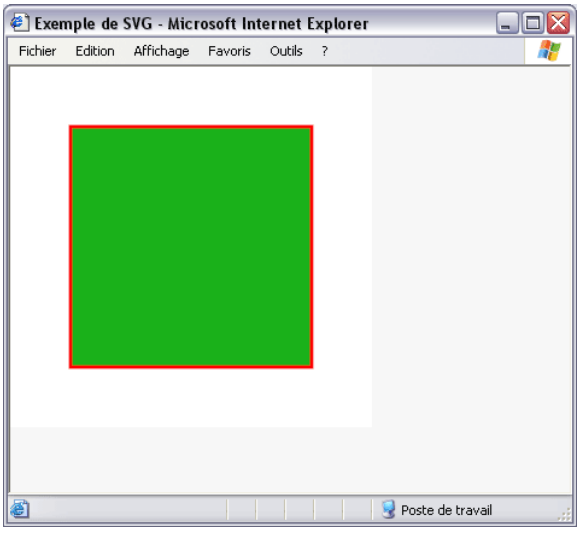

Figure 29 - Carré vert muni d'un bord rouge en SVG

Pour pouvoir afficher des graphiques SVG dans le navigateur, celui-ci doit être muni d'un plugin, comme *SVG Viewer* d'Adobe (http://www.adobe.com/svg).

Pour revenir à notre exemple, nous voudrions créer un histogramme en SVG à partir de données numériques contenues dans un fichier XML (`datas.xml`). On suppose que ce fichier contient le nombre de visites par mois sur notre site et qu'il est rangé par ordre chronologique :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!--------------------->
<!----- datas.xml ----->
<!--------------------->
<datas>
 <set id="1">
  <month>12</month><year>2004</year><visits>132</visits>
 </set>
 <set id="2">
  <month>01</month><year>2005</year><visits>147</visits>
 </set>
 <set id="3">
  <month>02</month><year>2005</year><visits>201</visits>
 </set>
 <set id="4">
  <month>03</month><year>2005</year><visits>185</visits>
 </set>
 <set id="5">
  <month>04</month><year>2005</year><visits>191</visits>
 </set>
</datas>
```

Les données d'un mois sont définies par l'élément `<set/>`.

Nous devons transformer ce fichier XML en SVG grâce à une feuille de style XSLT. Ensuite, nous sérialisons au format SVG pour l'affichage dans le navigateur.

Voici comment cela se passe dans le sitemap :

```
<map:match pattern="stats.svg">
  <map:generate src="content/datas.xml"/>
  <map:transform src="stylesheets/dat2svg.xsl"/>
  <map:serialize type="svgxml"/>
</map:match>
```

Nous faisons une transformation XSLT de `datas.xml` avec la feuille de style `dat2svg.xsl`. Le serializer pour le format SVG est `svgxml`.

Comme nous l'avons vu, le navigateur a besoin d'un plugin pour afficher le SVG. Pour contourner ce problème, Cocoon propose de transformer le SVG en JPG ou en PNG à la sérialisation. Il suffit de changer le serializer en `svg2jpeg` pour obtenir une image JPG et en `svg2png` pour obtenir une image PNG.

Le contenu de la feuille de style `dat2svg.xsl` est le suivant :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!----------------------->
<!----- dat2svg.xsl ----->
<!----------------------->
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
 <svg width="300" height="200">
  <title>Visites sur le site</title>
  <xsl:apply-templates/>
 </svg>
</xsl:template>

<xsl:template match="set">

<xsl:variable name="id"><xsl:value-of select="@id"/></xsl:variable>

<text x="5"> <!-- LEGENDE POUR CHAQUE BATONNET (mm-yyyy) -->
 <!-- calcul de la position en y (y) à partir de la valeur de "id" -->
 <xsl:attribute name="y">
  <xsl:copy-of select="($id * 24) + 15" />
 </xsl:attribute>
 <xsl:value-of select="month" />-<xsl:value-of select="year" />
</text>

<!-- BATONNET DE L'HISTOGRAMME -->
<!-- choix de la largeur des rectangles (height), de la position en x (x) et du
style (style) -->
<rect height="20" x="60" rx="5" ry="5"
      style="fill:#D4F4FF;stroke-width:1;stroke:#000000">
 <!-- calcul de la position en y (y) à partir de la valeur de "id" -->
 <xsl:attribute name="y">
  <xsl:copy-of select="$id * 24" />
 </xsl:attribute>
 <!-- calcul de la largeur (width) en fonction du nombre de visites -->
 <xsl:attribute name="width">
   <xsl:value-of select="visits" />
 </xsl:attribute>
</rect>

<text x="65"> <!-- VALEUR DU BATONNET -->
 <!-- calcul de la position en y (y) à partir de la valeur de "id" -->
 <xsl:attribute name="y">
  <xsl:copy-of select="($id * 24) + 15" />
 </xsl:attribute>
 <xsl:value-of select="visits" />
</text>

</xsl:template>

</xsl:stylesheet>
```

Pour chaque `<set/>`, nous créons trois choses :

- Un bâtonnet de l'histogramme
- La légende de ce bâtonnet
- Une valeur sur le bâtonnet

Le bâtonnet de l'histogramme est un rectangle de hauteur fixe et dont la largeur est proportionnelle au nombre de visites pour le mois concerné.

La légende du bâtonnet sert à identifier le mois concerné. Elle est de la forme *mm-yyyy*, par exemple "03-2005". On la place en face (à gauche) du bâtonnet.

La valeur du bâtonnet correspond au nombre de visites sur le mois. On la place sur le bâtonnet lui-même.

Pour ne pas que tous les rectangles s'affichent au même endroit sur le graphique, nous avons du gérer des décalages. Nous avons réalisé cela grâce à la valeur de l'attribut `id` de l'élément `<set/>`.

Le résultat que nous obtenons est celui-ci :

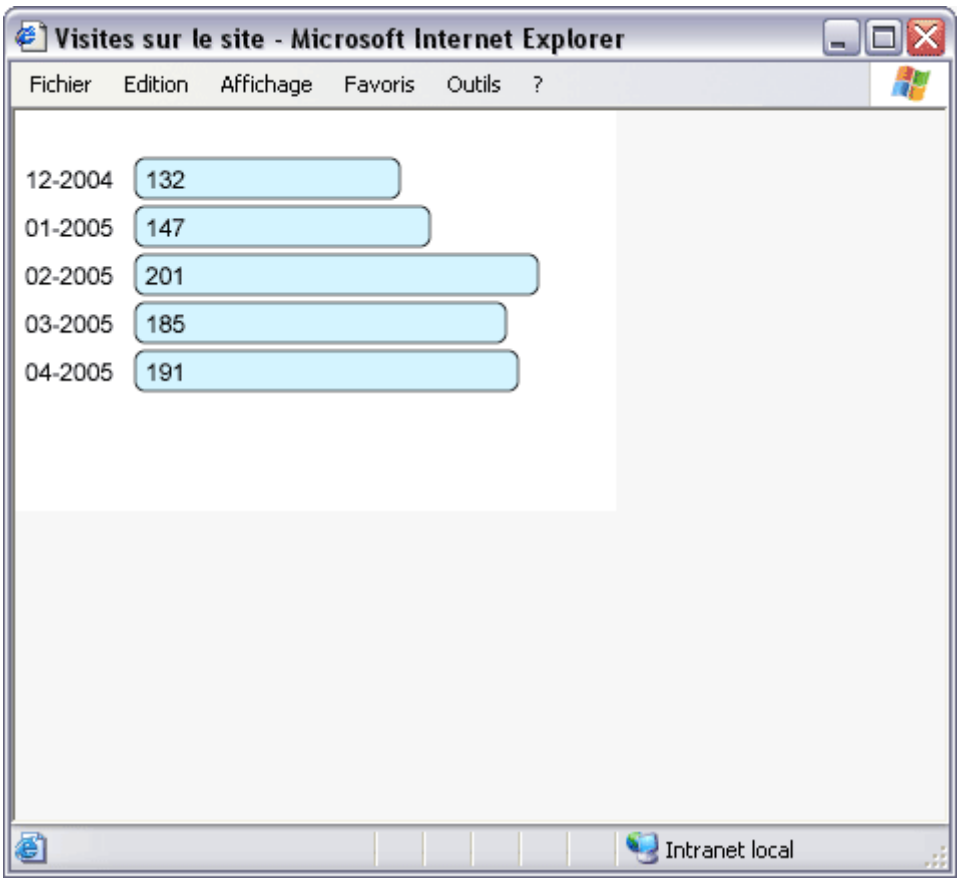

Figure 30 - Aperçu des visites sur le site en SVG

Chapitre 6. Politique adoptée sur le site d'eTransparence

1. Mise en situation

Avant de se lancer dans l'implémentation proprement dite, nous nous sommes imposés une politique qu'il a fallu tenir à l'esprit pendant tout le développement de l'application. En respectant cette politique jusqu'au bout, nous étions sûr d'obtenir un site d'un très haut niveau d'accessibilité et d'efficacité.

2. Respect des normes

Le site doit respecter les standards définis par le W3C, et en particulier les normes du XHTML (1.0 Strict) et du CSS (2.0).

Le fait de respecter les recommandations du W3C assure au site une compatibilité certaine sur n'importe quel navigateur moderne sur n'importe quel système d'exploitation, à condition que le navigateur respecte lui-même les standards, ce qui n'est malheureusement pas toujours le cas...

Pour être certain et pour pouvoir vérifier qu'un site respecte les normes, le W3C a mis en place sur son site Web, une quantité d'outils pour vérifier la validité des langages dont cet organisme assure la gestion.

Sur le site d'eTransparence, il est possible de valider le XHTML et le CSS en cliquant sur les boutons suivants :

Figure 31 - Boutons de validation XHTML et CSS

Comme le site est valide XHTML, voici le message que vous allez apercevoir :

Figure 32 - Résultat du validateur XHTML

Même chose pour la validation du CSS :

Figure 33 - Résultat du validateur CSS

Validateurs :

- (X)HTML : http://validator.w3.org/
- CSS : http://jigsaw.w3.org/css-validator/

3. Design "tableless"

Le site doit adopter le design "tableless" c'est-à-dire sans mise en forme par tableaux HTML. Les tableaux en HTML sont définis par l'élément `<table/>`, les rangées par l'élément `<tr/>` et les cellules par l'élément `<td/>`. Au lieu d'utiliser la mise en page par tableaux, nous allons utiliser la puissance du langage de formatage CSS et sa fonctionnalité permettant le positionnement des éléments.

3.1. Introduction

Originellement prévus dans le but de mettre en forme des données, les tableaux ont rapidement été détournés par les auteurs Web pour combler un vide qui faisait cruellement défaut au HTML à l'époque[3] : la mise en page.

En effet, qui n'a jamais été tenté d'utiliser des tableaux pour aligner un menu sur la gauche ou un titre au dessus du texte?

Bien entendu, l'utilisation de tableaux est valide selon les normes établies mais hélas, les standards ne sont pas entièrement respectés. En effet, respecter les standards, ce n'est pas que respecter la syntaxe du balisage, c'est aussi respecter la sémantique des balises !

A cette époque, le CSS n'existait pas et encore moins la séparation du contenu et de la présentation. Les développeurs pensaient avoir trouvé la parade avec les tableaux[4], et ont donc commencé à utiliser massivement ces derniers pour du positionnement et autres mises en page complexes.

[3] Au milieu des années 90.
[4] Il est important de noter qu'il ne faut pas bannir les tableaux pour autant. Ils remplissent parfaitement leur rôle quand il s'agit de structurer des données.

3.2. Positionnement CSS

De nos jours, avec l'apparition du positionnement CSS, tout est plus devenu plus simple. Pour réaliser une mise en page réussie, il suffit, d'une part, de créer des blocs[5] dans le fichier HTML et, d'autre part, de créer un fichier CSS qui permet de positionner ces blocs où l'on veut. De cette manière, on pourrait imaginer avoir plusieurs fichiers CSS qui feraient office de différents habillages pour l'application Web.

Dans cet ordre d'idée, le site http://www.csszengarden.com propose à n'importe qui d'appliquer sa propre feuille de style CSS à un document HTML de base et commun à tous. Les résultats sont très différents, alors que le fichier sur lequel les participants se basent est identique.

Voici un exemple de mise en page par tableaux :

```
<!---------------------------->
<!----- sample-dirty.htm ----->
<!---------------------------->
<html>
 <body>
  <table width="800" border="1">
   <tr height="80"><td colspan="3" width="800">Titre</td></tr>
   <tr height="500">
    <td width="100">Menu</td>
    <td width="600">Contenu</td>
    <td width="100">Colonne de droite</td>
   </tr>
   <tr height="20"><td colspan="3" width="800">Bas</td></tr>
  </table>
 </body>
</html>
```

Le titre, le menu, la colonne de droite et le bas de la page sont chacun contenus dans une cellule du tableau. Cellules qu'il a fallu fusionner (avec l'attribut `colspan`) et dont il a fallu spécifier la taille (attributs `height` et `width`) pour arriver à un résultat valable.

Et voici un exemple de fichier HTML "tableless" accompagné de sa feuille de style CSS :

```
<!---------------------------->
<!----- sample-clean.htm ----->
<!---------------------------->
<html>
 <body>
  <div id="title">Titre</div>
  <div id="menu">Menu</div>
  <div id="content">Contenu</div>
  <div id="right-col">Colonne de droite</div>
  <div id="bottom">Bas</div>
 </body>
</html>
```

[5] Les principaux éléments qui peuvent faire office de blocs sont : div, span, h1, h2, h3, h4, h5, h6, p, ul, ol, li, dl, dd, blockquote, pre et address.

On remarque qu'aucune information de formatage ne vient polluer notre fichier HTML. Seuls des blocs sont créés. Cette page est également moins volumineuse et beaucoup plus lisible que la précédente. Les blocs seront positionnés avec la feuille de style CSS ci-dessous :

```
<!---------------------->
<!----- sample.css ----->
<!---------------------->

div#title <!-- positionnement du titre -->
{
 border:1px solid #000000;
 position:absolute; left:200px; top:5px;
 width:800px; height:80px;
}
div#menu <!-- positionnement du menu -->
{
 border:1px solid #000000;
 position:absolute; left:200px; top:85px;
 width:100px; height:500px;
}
div#content <!-- positionnement du bloc de contenu -->
{
 border:1px solid #000000;
 position:absolute; left:300px; top:85px;
 width:600px; height:500px;
}
div#right-col <!-- positionnement de la colonne de droite -->
{
 border:1px solid #000000;
 position:absolute; left:900px; top:85px;
 width:100px; height:500px;
}
div#bottom <!-- positionnement du bas de page -->
{
 border:1px solid #000000;
 position:absolute; left:200px; top:585px;
 width:800px; height:20px;
}
```

3.3. Problèmes des tableaux

En résumé, les problèmes des tableaux sont les suivants :

- Un tableau est lourd à charger pour le navigateur, car il se fonde sur des règles pour calculer la hauteur des lignes et des cellules.
- Les tableaux nuisent à l'impression. Une page qui comporte une mise en page par tableaux risque de ne plus rien donner une fois imprimée.
- La mise en page par tableaux vaudra un coût de production et de maintenance très haut.
- Les tableaux alourdissent les pages, en terme de taille de fichiers. Les pages avec une structuration par positionnement CSS ont une taille beaucoup plus petite. Dans certains cas, le rapport peut descendre jusque 1/2.

4. Compatibilité

Le site doit être compatible avec tous les navigateurs modernes.

Étant donné que le site est conforme aux normes du W3C, ce souci de compatibilité ne devrait en principe pas se poser. Malheureusement, certains navigateurs sont très capricieux, c'est le cas du navigateur de Microsoft, Internet Explorer. Ce dernier, qui s'approprie 88,42% des parts de marché des navigateurs Web (au 12 décembre 2004), est loin d'être le meilleur outil pour naviguer sur Internet. Il comporte en effet de graves lacunes, notamment en ce qui concerne l'affichage des images PNG et la gestion du CSS 2.0. Ces technologies sont pourtant des normes du W3C depuis un certain temps.

Pour être correctement vu sur ce navigateur, le site a du subir des artifices et utiliser des moyens détournés. Il a même fallu sacrifier une partie des fonctionnalités possibles du CSS, non gérée par Explorer.

Une communauté de plus en plus importante d'internautes adopte l'utilisation de navigateurs alternatifs, Mozilla Firefox[6] en tête. Ce dernier à l'avantage de respecter à la lettre et d'implémenter tous les standards du Web. De plus, il propose une pléthore de modules et d'outils, tous plus utiles les uns que les autres.

Le site a été testé sur Internet Explorer 6.0, Firefox 1.0, Opera 7.54, K-Meleon 0.9 et Netscape 8.0 sous Windows XP SP2, Internet Explorer 5.0 et Safari sous Mac OS X et Konqueror 3.3 sous Linux. Les résultats sont excellents, il n y a aucun problème à signaler sur aucun de ces navigateurs.

Le site est également visible avec n'importe quelle configuration d'écran, mais il est conseillé de disposer au minimum d'un affichage de 1024*768 pour une visite dans de bonnes conditions.

5. Référencement

Le site doit être "très bien référencé" sur les moteurs de recherche et en particulier sur Google, leader incontestable du marché (+ de 75% des parts de marché au 1er mars 2005[7]). Par "très bien référencé", on entend qu'il ne suffit pas que l'internaute tape "eTransparence" dans un moteur de recherche pour trouver la site, il faut que le site soit "trouvable" en utilisant des mots clés plus abstraits comme par exemple "informatique Belgique".

Le référencement de sites Web est quelque chose de très complexe qui pourrait certainement faire un bon sujet de mémoire à lui seul pour en explorer tous les aspects. Dans le cadre de ce mémoire, nous allons simplement essayer d'optimiser le site pour qu'il soit le plus "visible" possible sur la toile.

Pour mettre toutes les chances de notre coté afin d'être bien référencé, nous avons suivi les dix conseils de base prônés par http://www.webrankinfo.com, une des ressources les plus complètes sur le référencement.

[6] La version 1.0 de Firefox est sortie le 9 novembre 2004.
[7] Source : Web Rank Info (http://www.webrankinfo.com/actualites/200503-barometre.htm)

1. Contenu

   Le site doit bénéficier d'un contenu utile et original, pour intéresser le visiteur. En outre, au plus le site possède de pages, au mieux il sera référencé.

   Dans cette optique, il a été décidé que le site d'eTransparence comporterait, d'une part, des pages commerciales qui présentent les solutions et les produits proposés par la société, et d'autre part, des articles purement scientifiques qui ont pour but d'expliquer certaines technologies informatiques. De cette manière, le contenu devient plus intéressant et le nombre de pages augmente, favorisant le référencement.

2. Nombreux liens externes

   Il est important d'avoir un grand nombre de liens externes pointant vers son site, surtout pour Google. En effet, ce dernier a mis au point un concept nommé le PageRank qui indique le degré de sensibilité d'un site aux moteurs de recherche, sur une échelle de 0 à 10. Au plus le site a un PageRank élevé, au plus le site est "trouvable" sur Google. Le défi pour les webmasters est donc de maximiser ce PageRank. Et le meilleur facteur faisant augmenter cette valeur[8] est le nombre de liens externes.

   Pour eTransparence, il a été demandé à des partenaires de faire de l'échange de lien, un lien vers eTransparence est laissé sur chaque site Web créé par la société (avec la mention "Réalisé par eTransparence") et un lien vers eTransparence est laissé dans chaque signature d'e-mails et de messages sur les forums.

3. Liens externes de qualité

   Il ne suffit pas d'avoir des liens externes vers son site, ces liens doivent également être de bonne qualité. Ceux-ci doivent se trouver sur des pages ayant elles-mêmes un bon PageRank, et de préférence, sur des sites traitant du même sujet.

4. Bons titres

   Les titres doivent être bien choisis et refléter le contenu réel de la page.

5. S'inscrire dans les grands annuaires

   L'inscription dans de grands annuaires comme Yahoo! et DMOZ est très importante car elle fait augmenter considérablement le PageRank.

6. Pas de frames, et une bonne adresse de site

   Il est important d'avoir un nom de domaine payant du genre http://www.etransparence.be et non pas un nom de domaine d'hébergeur gratuit du type http://membres.lycos.fr/etransparence. De plus, les frames ne sont pas à conseiller dans les pages car mal gérées par les moteurs de recherche.

[8] Il est possible de connaître le PageRank de n'importe quelle page Web en téléchargeant la barre d'outil Google sur http://toolbar.google.com/.

Dans le cas d'eTransparence, ces problèmes ne se posent pas car, premièrement, le nom de domaine est "bon", et deuxièmement, le site ne comporte pas de frames car ces dernières ont été dépréciées et ne font pas partie de la recommandation XHTML 1.0 Strict.

7. Un site toujours disponible

   Les robots d'indexation des moteurs de recherche risquent de rayer un site de leur liste s'ils se retrouvent sur une page 404[9] lors de leurs visites. Il faut donc posséder un hébergement fiable, ne subissant pas de coupures.

   eTransparence a adopté une solution professionnelle pour son hébergement, donc il ne devrait pas y avoir de problèmes de ce coté là.

8. Interconnexion de vos pages

   Les pages du site doivent posséder beaucoup de liens entre-elles, en utilisant des mots ciblés au lieu de liens du style "Cliquez ici". Il est important aussi de disposer d'une page qui contient un plan du site, de manière à ce que les robots sachent visiter le site en entier en parcourant cette page.

9. Pas de site sans texte

   Le site ne doit pas être entièrement en Flash ou en image, car les robots ne savent pas lire ces formats binaires. Il est également important de remplir l'attribut `alt` (texte alternatif) des balises `<img/>` car il est pris en compte par les robots.

   Dans la recommandation XHTML 1.0, l'attribut `alt` est obligatoire sur les images. De ce fait, chaque image du site d'eTransparence comporte l'attribut `alt`.

10. Mises à jour régulières du site

    Google tout comme n'importe quel internaute adore les sites souvent mis à jour. Les robots reviendront plus souvent visiter le site si celui-ci est mis à jour régulièrement.

Notons que la très connue technique de référencement par balises `<meta/>` qui consiste à placer une description du site, des mots clés, etc. dans des balises `<meta/>` se situant entre `<head>` et `</head>` n'est plus prise en compte du tout par Google. C'est néanmoins toujours le cas pour des moteurs de recherche comme Yahoo ou Altavista.

[9] 404 est le code d'erreur lorsqu'une page est introuvable.

Chapitre 7. Analyse fonctionnelle du site d'eTransparence

1. Mise en situation

Cette partie explique les différentes fonctionnalités que possédera le site d'eTransparence, une fois terminé. Nous ne rentrons pas dans les détails techniques de l'implémentation mais tentons de décrire à quoi ressemblera et ce que permettra le site d'un point de vue externe. Cette analyse a été réalisée à partir du cahier des charges fourni par eTransparence.

2. Tâches à accomplir

2.1. Partie commerciale

Le site doit décrire les activités commerciales de la société eTransparence sous la forme de petits textes de présentation. Ces textes doivent être conviviaux ce qui implique qu'il faut utiliser des images et formater le texte (utilisation du gras, de l'italique, du souligné, des couleurs) pour mettre en valeur certaines idées.

Les activités commerciales d'eTransparence sont :

- Les services qui se décomposent de la façon suivante :
  - Des solutions réseaux et télécommunications, principalement dans le sans-fil
  - L'installation et la configuration de serveurs en tous genres basés sur des solutions Linux
  - L'installation et la configuration de matériel de téléphonie
- Le service technique qui établit des diagnostics, installe et répare tout matériel informatique
- La réalisation de sites Web (Internet et Intranet), de logiciels et de workflows
- La vente de matériel informatique (voir section suivante)

Un menu de navigation doit être toujours visible sur la gauche de la page. Celui-ci permet d'accéder rapidement aux autres pages du site.

De plus, les pages de cette partie doivent pouvoir être imprimée (via une version imprimable) et téléchargées au format PDF, pour une impression ou une lecture hors ligne.

En résumé, une page de présentation des activités d'eTransparence ressemble à ceci :

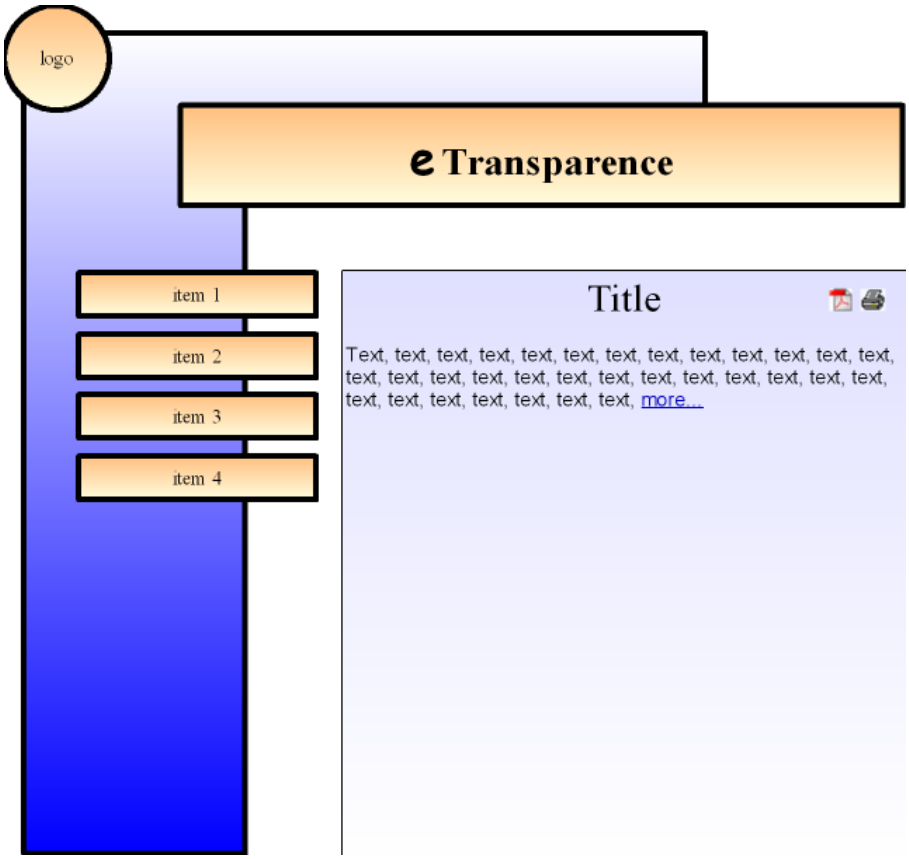

Figure 34 - Aperçu d'une page du site d'eTransparence

2.2. Vente de matériel

Le site doit comporter une vitrine virtuelle pour le matériel vendu par eTransparence. Il s'agit principalement d'ordinateurs de bureau, d'ordinateurs portables et de réseaux sans-fil (c'est-à-dire des routeurs sans-fil, des cartes WiFi, etc.).

L'administrateur du site doit pouvoir ajouter et supprimer de manière simple les nouveaux articles et les nouvelles promotions que propose eTransparence.

Cette partie différencie la vente de matériel pour les entreprises de la vente de matériel pour les particuliers. Les prix de la partie entreprise sont exprimés sans la taxe sur la valeur ajoutée (HTVA) alors que les prix de la partie particulier incluent toutes les taxes (TTC). Il s'agit là de la seule et unique différence entre ces deux parties.

Le matériel à vendre se présente sous la forme d'une liste composée du nom de l'article et de son prix. Un click sur un élément de la liste permet d'avoir une description complète de l'article (caractéristiques, configuration logicielle et matérielle, garanties, etc.).

En résumé, la partie s'occupant de la vente de matériel se présente de cette façon :

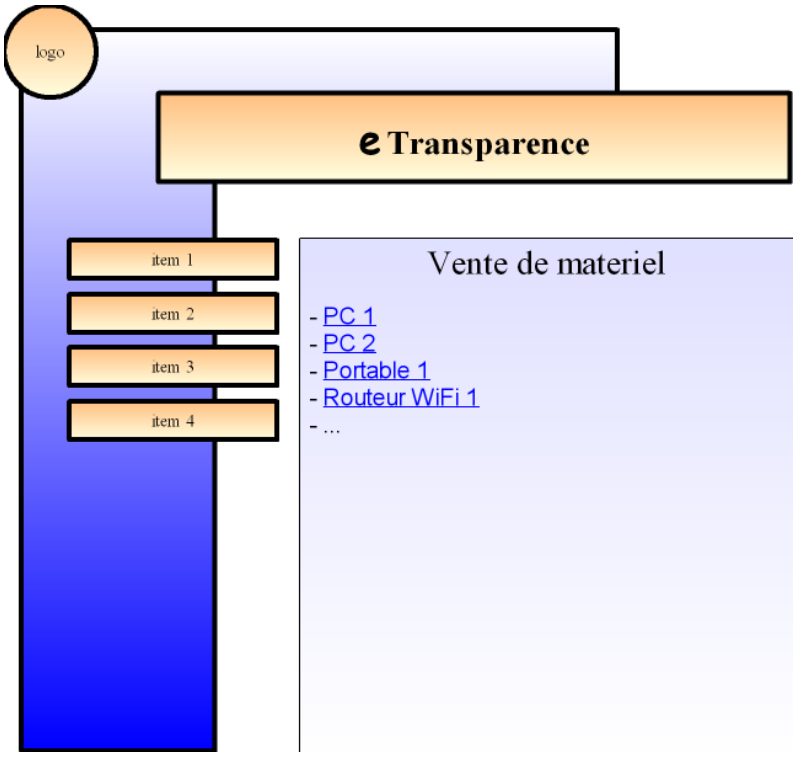

Figure 35 - Aperçu de la page "Vente de matériel"

Si nous cliquons sur "Portable 1" par exemple, nous obtenons :

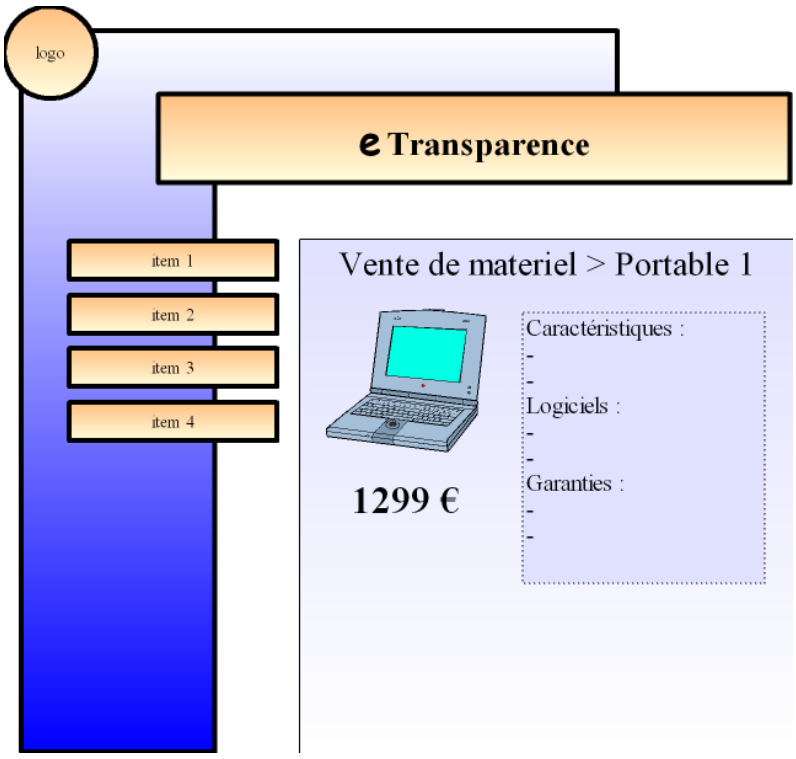

Figure 36 - Aperçu d'un article à vendre

2.3. Partie scientifique

La partie scientifique doit être consacrée à définir et expliquer certains termes et certaines technologies mentionnées dans les pages de la partie commerciale. Les pages de cette partie se présentent exactement comme celles de la partie commerciale, c'est à dire sous la forme d'un texte continu et formaté. De plus, les pages doivent également pouvoir être imprimées et téléchargées au format PDF.

Les textes de cette partie peuvent être importés de sites scientifiques, comme par exemple : Comment ça marche? (http://www.commentcamarche.net) ou Wikipédia (http://fr.wikipedia.org). Cette procédure d'import doit se réaliser de manière automatique (pas de copier-coller).

La maintenance de cette partie doit être très simple et demander très peu de manipulations de la part de l'administrateur du site.

2.4. Système de news

Le site doit bénéficier d'un système de news mis à jour régulièrement. En réalité, le site va importer des news intéressantes de sites extérieurs grâce à la technologie RSS, ce qui permettra de réduire le temps de maintenance.

Sur le site, la première page des news permet de choisir le fil RSS que l'on veut consulter parmi une liste. Il est important que cette dernière soit facilement modifiable par l'administrateur du site. Ensuite, une fois le fil RSS choisi, apparaît la liste des dernières nouvelles de celui-ci.

2.5. Formulaire de contact

Les visiteurs doivent pouvoir envoyer un message via un formulaire de contact s'ils ont une question à poser ou une suggestion à faire. Pour s'identifier et permettre une réponse de la part d'eTransparence, on demandera au visiteur d'entrer son adresse e-mail en plus de son message.

Le formulaire se présente de cette façon :

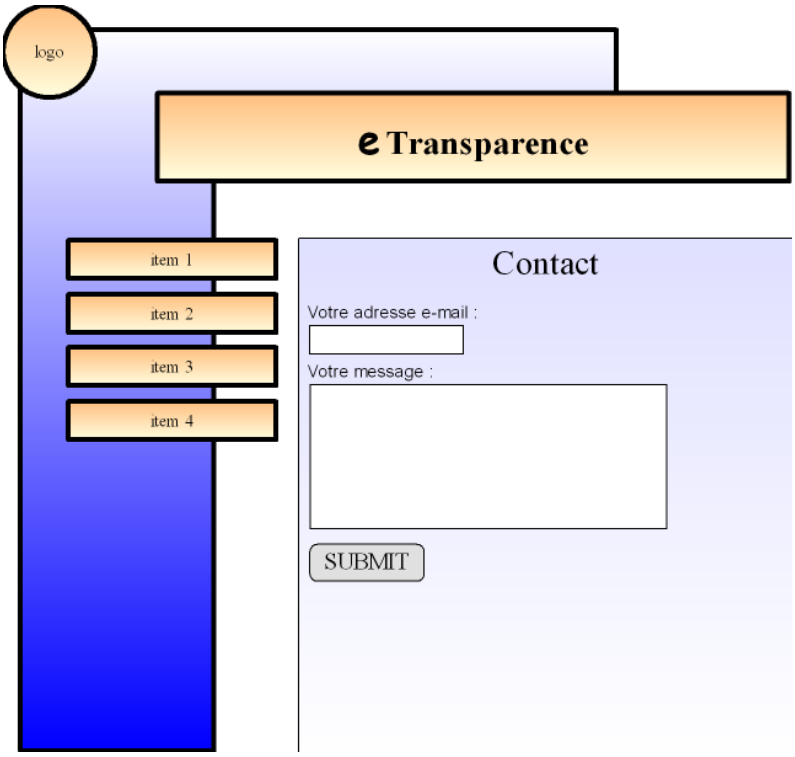

Figure 37 - Aperçu de la page "Contact"

Les messages doivent être envoyés par e-mail à l'adresse info@etransparence.be. Il faut donc élaborer un mécanisme permettant d'envoyer un e-mail lors de la soumission d'un formulaire.

2.6. Moteur de recherche interne

Le site doit être muni d'un moteur de recherche qui doit porter uniquement sur les pages internes au site d'eTransparence, c'est-à-dire au domaine www.etransparence.be.

Ce moteur de recherche ne doit pas nécessairement être créé de toute pièce. L'utilisation des possibilités offertes par des moteurs de recherche de très haut niveau comme Google peut être envisagée.

Pour consulter le moteur de recherche, le visiteur est invité à entrer un ou plusieurs mots-clés dans un formulaire. Celui-ci doit être visible sur chaque page pour une meilleure accessibilité.

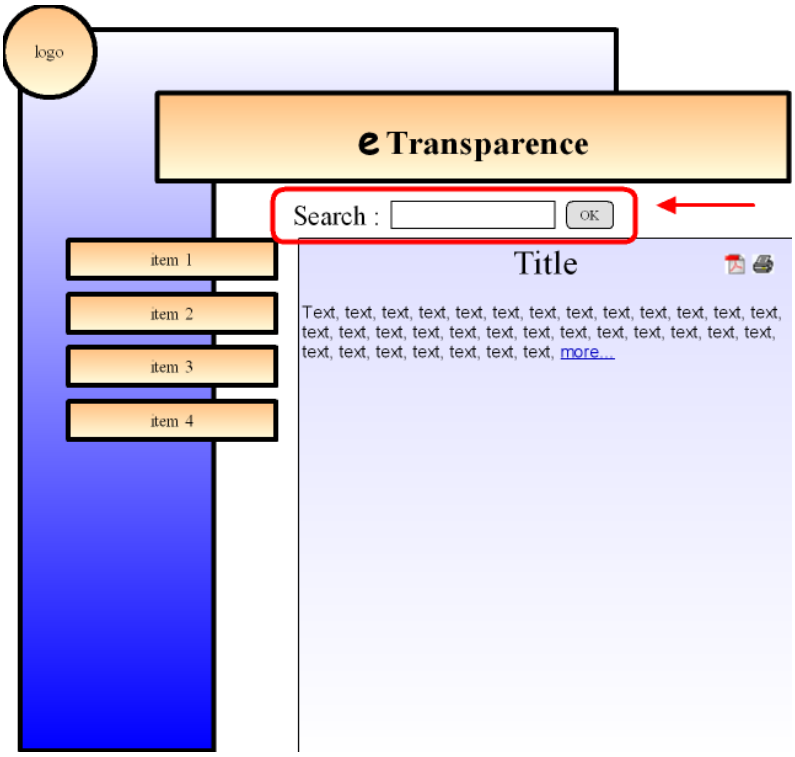

Figure 38 - Aperçu du moteur de recherche interne

2.7. Espace administrateur

Une interface d'administration doit être prévue pour gérer les aspects dynamiques du site.

Quelques possibilités de l'espace administrateur sont :

- ajout/modification/suppression d'articles à la rubrique "vente de matériel"
- ajout/suppression de fils de news
- ajout/modification/suppression de nouveaux contenus à la partie scientifique
- ...

Cet espace doit être protégé par login et mot de passe pour que n'importe quel intrus ne puisse y accéder.

2.8. Espace client

Les clients doivent pouvoir se connecter à un espace personnel protégé par login et mot de passe. Ils recevront ces identifiants sur demande ou automatiquement lors de leur premier achat chez eTransparence. Cet espace leur permettra entre autres de voir l'état de leurs commandes et achats.

2.9. Version PDA

Une version "light" du site doit être créée pour être visible sur un agenda électronique de type PDA (ou Pocket PC). En effet, ces appareils disposent d'une résolution d'écran beaucoup plus petite qu'un écran d'ordinateur normal. Il est donc obligatoire de laisser tomber certains détails graphiques superflus, qui s'avèrent très agréables sur un écran d'ordinateur normal, mais tout à

fait nuisibles à la lecture sur l'écran d'un agenda électronique. La différence entre les deux versions sera donc uniquement graphique, le contenu restant le même.

Cette version PDA doit être disponible via l'URL http://pda.etransparence.be.

2.10. Version WAP

Le site doit être disponible sur le WAP via l'URL http://wap.etransparence.be. Étant donné que l'écran d'un téléphone portable ne se prête pas du tout à la lecture de longs textes, cette version sera un résumé du site, allant à l'essentiel.

Chapitre 8. Implémentation du site d'eTransparence

1. Structure générale du site

1.1. Arborescence de fichiers

Pour mieux comprendre comment sont organisés tous les fichiers dont on parlera plus loin, en voici l'arborescence complète :

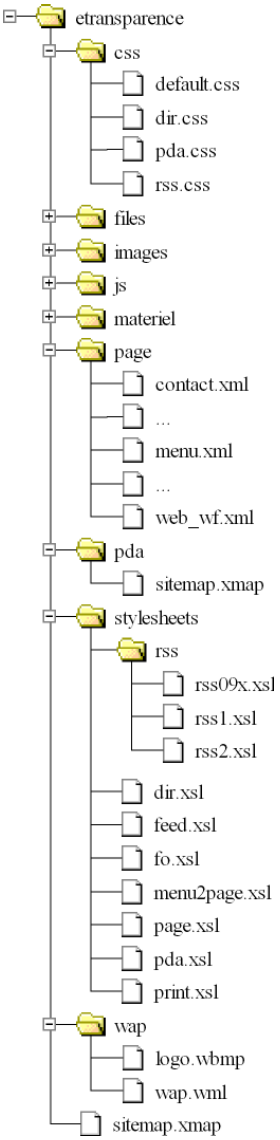

Figure 39 - Arborescence de fichiers du site d'eTransparence

Le répertoire `css` contient les feuilles de style CSS, `files` contient les fichiers de la zone téléchargement, `images` contient les images, `js` contient des scripts en JavaScript, utiles pour certaines fonctionnalités, `materiel` contient les fichiers XML de la partie vente, `page` contient toutes les pages au format XML, `stylesheets` contient les feuilles de style XSLT et `wap` contient le fichier WML de la version WAP du site.

1.2. Structure d'une page

La structure du site d'eTransparence est toujours la même pour chaque page. Elle se compose du logo de la société en haut, du menu de navigation sur la gauche, du contenu de la page au milieu et d'un cadre avec des liens sur la droite.

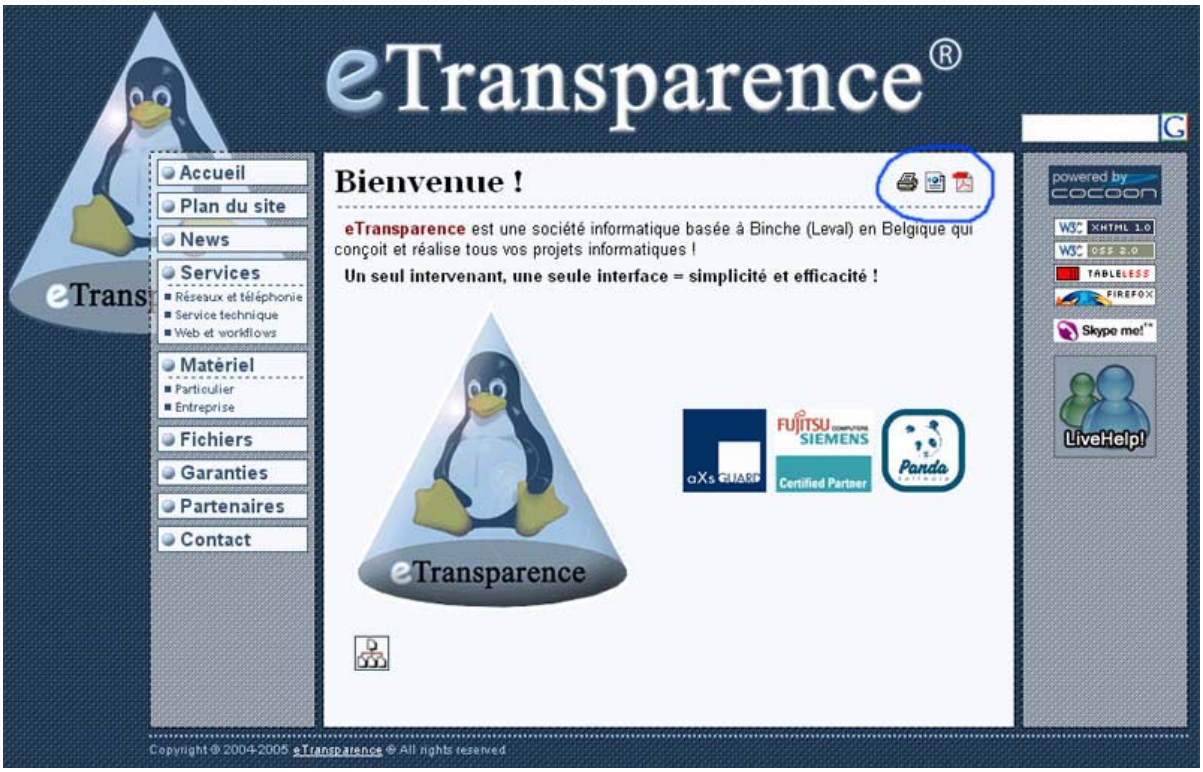

Figure 40 - Copie d'écran du site d'eTransparence

1.2.1. Menu de navigation

Le menu est situé sur la gauche de chaque page. Il est généré à partir du fichier `menu.xml` dont la structure est la suivante :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!-------------------->
<!----- menu.xml ----->
<!-------------------->
<menu>
 <menuitem>
  <label/>
  <link/>
  <submenu>
   <submenuitem>
     <label/>
     <link/>
```

```
     <subsubmenu>
      <!-- ... -->
     </subsubmenu>
   </submenuitem>
 </menuitem>
 <menuitem>
  <!-- ... -->
 </menuitem>
 <!-- ... -->
</menu>
```

Le nombre de sous-niveaux peut être infini mais il faut savoir que seuls les deux premiers niveaux sont significatifs et apparaîtront sur la page. En effet, la feuille de style XSLT ne tient compte que des éléments `<menu/>`, `<menuitem/>`, `<submenu/>` et `<submenuitem/>`. Voici la portion de la feuille de style (`page.xsl`) qui s'occupe de transformer le menu :

```
<xsl:template match="menuitem[not(submenu)]">
 <div class="menuitem">
  <img src="images/puce.gif" alt="eTransparence"/> <a class="menu">
  <xsl:attribute name="href">
   <xsl:value-of select="link"/>
  </xsl:attribute>
  <xsl:value-of select="label"/>
  </a>
 </div>
</xsl:template>

<xsl:template match="menuitem">
 <div class="menuitem">
  <img src="images/puce.gif" alt="eTransparence"/> <xsl:value-of
select="label"/>
  <xsl:apply-templates select="submenu"/>
 </div>
</xsl:template>

<xsl:template match="submenu">
 <div class="submenu">
  <xsl:apply-templates/>
 </div>
</xsl:template>

<xsl:template match="submenuitem">
 <div class="submenuitem">
  <img src="images/carre.gif" alt="eTransparence"/> <a class="menu">
  <xsl:attribute name="href">
   <xsl:value-of select="link"/>
  </xsl:attribute>
  <xsl:value-of select="label"/>
  </a>
 </div>
</xsl:template>
```

On note que le traitement est différent selon qu'un élément du menu possède un sous-menu ou pas. La différentiation est faite grâce à la requête XPath : `menuitem[not(submenu)]`. On matche cette expression si l'élément du menu ne possède pas de sous-menu.

Les blocs `<div/>` des éléments du menu et des sous-menus sont mis en forme et positionnés ensuite grâce à une feuille de style CSS.

1.2.2. Contenu de la page

Comme nous l'avons vu, le menu se trouve dans un fichier à part, séparé du contenu des pages.

La structure d'une page est exactement la même que dans l'exemple récapitulatif (voir **Chapitre 6, Section 2, "Répertoires et canevas de fichiers"**). L'exemple ci-dessous présente le code source de la page d'accueil (`home.xml`) qui correspond à la copie d'écran de la figure 40 :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!-------------------->
<!----- home.xml ----->
<!-------------------->
<page id="home">
 <title>Bienvenue !</title>
  <actions>
   <print/><xml/><pdf/>
  </actions>
  <content>
   <para>
    <et>eTransparence</et> est une société informatique basée à Binche (Leval)
    en Belgique qui conçoit et réalise tous vos projets informatiques !
   </para>
   <para>
    <b>Un seul intervenant, une seule interface = simplicité et efficacité !</b>
   </para>
   <para>
    <img src="images/homelogo.gif" alt="Bienvenue sur le site d'eTransparence
    | Consultance informatique en Belgique"/>
   </para>
   <para>
    <img href="sitemap" src="images/map.gif" alt="Sitemap | Plan du site"/>
   </para>
   <partners/>
  </content>
</page>
```

Dans le sitemap, nous utilisons un aggregator pour concaténer cette page avec le menu avant la transformation par la feuille de style XSLT. Ensuite, nous sérialisons au format XHTML.

L'URI pour arriver à la page d'accueil est `home` :

```
<map:match pattern="*">
  <map:aggregate element="document">
    <map:part src="page/menu.xml"/>
    <map:part src="page/{1}.xml"/>
  </map:aggregate>
  <map:transform src="stylesheets/page.xsl"/>
  <map:serialize type="xhtml"/>
</map:match>
```

Une feuille de style CSS (définie dans la feuille de style XSLT) s'occupe ensuite de positionner et d'habiller les éléments.

1.2.3. Cadre de droite

Le cadre de droite ne se trouve pas dans un fichier séparé comme le menu, il est défini en dur dans la feuille de style XSLT. Ce cadre contient des liens divers, notamment vers les validateurs XHTML et CSS (voir **Chapitre 7, Section 2, "Respect des normes"**), ce qui permet de valider la page courante à tout moment.

Un click sur le lien "Skype Me" permet de se retrouver en ligne avec un des consultants d'eTransparence. Cela implique qu'on ait installé le logiciel Skype (http://www.skype.com) sur son ordinateur. Skype permet de téléphoner avec son PC grâce au protocole peer-to-peer (P2P).

1.2.4. PDF, XML et version imprimable

Chaque page est disponible dans quatre formats différents. Outre le format XHTML de base, il est également possible de voir le code source XML de la page, de la télécharger au format PDF (pour la lire hors ligne ou l'imprimer) et d'en avoir une version imprimable. Ces différentes versions sont accessibles en cliquant sur les boutons qui sont entourés de bleu sur la figure 40.

La version imprimable est obtenue en ajoutant l'extension `.print` au nom de la page. Par exemple, pour obtenir la version imprimable de la page `home`, l'URI est `home.print`. En pratique, deux choses changent par rapport à la version XHTML de base. Premièrement, on fait appel à une autre feuille de style XSLT simplifiée (`print.xsl`). Deuxièmement, le menu n'a plus besoin d'être agrégé au contenu de la page. Cela nous donne :

```
<map:match pattern="*.print">
  <map:generate src="page/{1}.xml"/>
  <map:transform src="stylesheets/print.xsl"/>
  <map:serialize type="xhtml"/>
</map:match>
```

Un événement Javascript agissant sur la balise `<body/>` permet d'afficher l'invite d'impression lors du chargement de la page :

```
<body onload="window.print()">
```

Cela nous permet d'obtenir le résultat suivant :

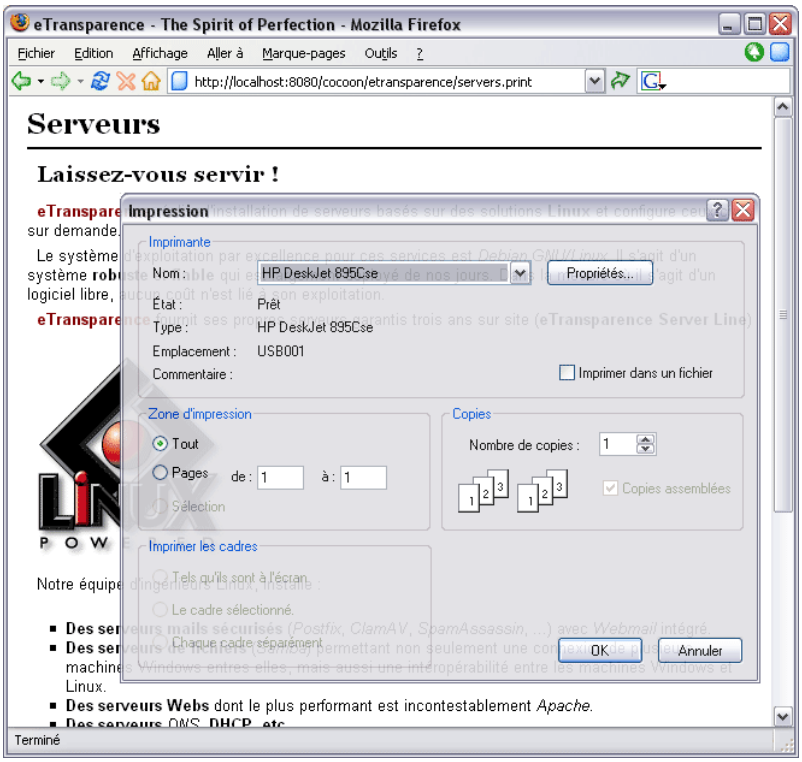

Figure 41 - Version imprimable d'une page

Les versions PDF et XML sont exactement (ou à peu de chose près) les mêmes que dans l'exemple récapitulatif. Voici par exemple la version PDF de la page "Web & Workflows" :

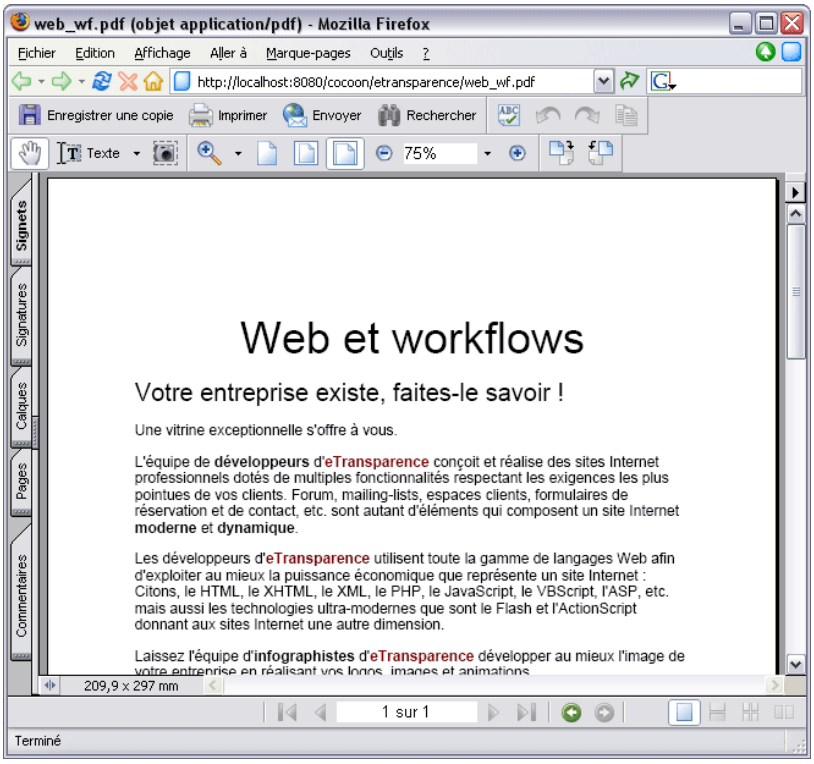

Figure 42 - Version PDF d'une page

La feuille de style XSLT (`fo.xsl`) que nous avons utilisé pour transformer le XML en XSL-FO (pour ensuite sérialiser en PDF) est semblable à celle de l'exemple récapitulatif.

Schématiquement, voici comment est déclinée la page d'accueil dans les quatre versions différentes :

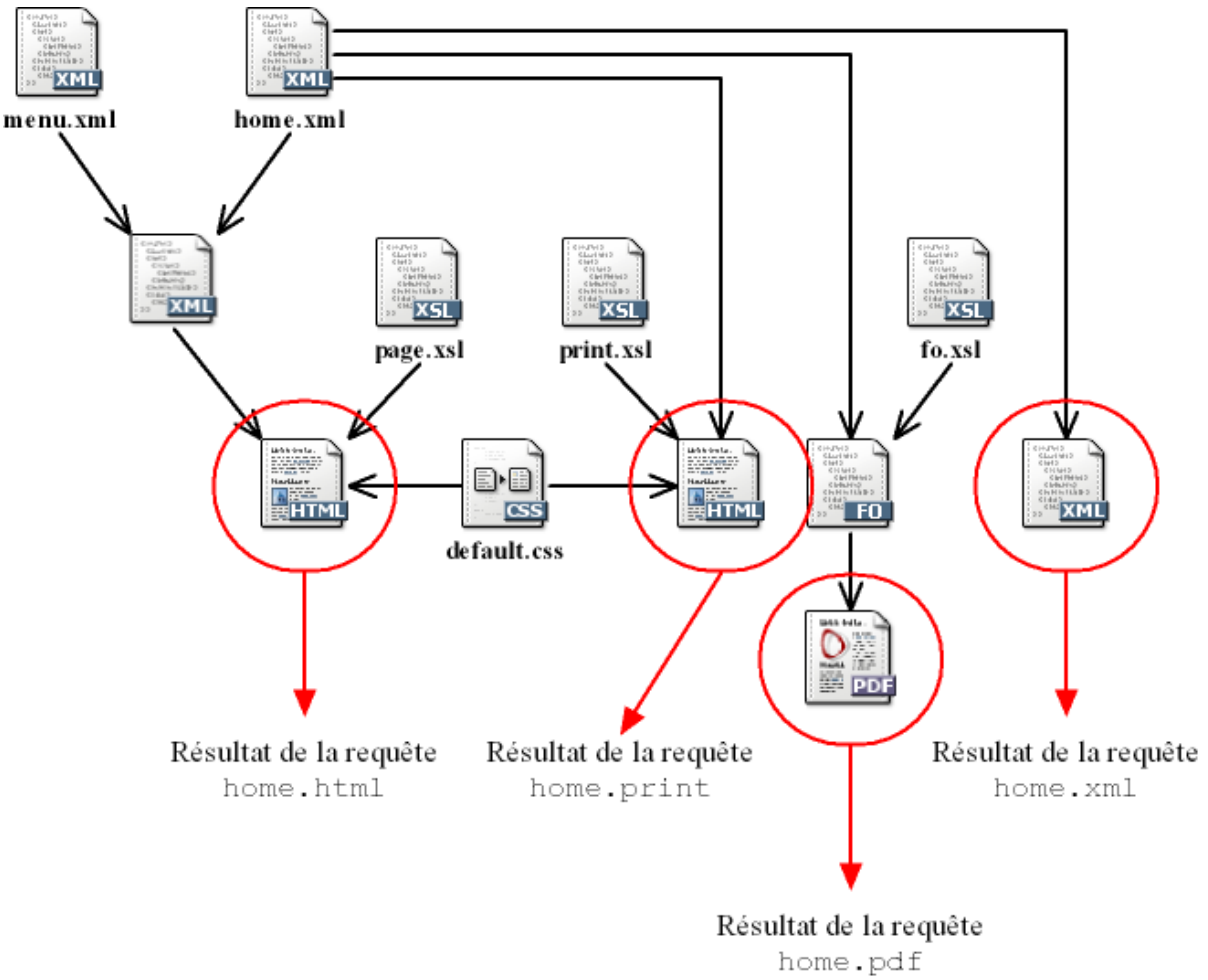

Figure 43 - Construction des quatre formats de la page d'accueil

1.3. Plan du site

Comme nous l'avons vu, disposer d'un plan sur son site est très favorable au référencement, car cela permet aux robots des moteurs de recherche de visiter le site entièrement grâce au parcours de cette page.

Par facilité de maintenance, cette page est créée à partir de `menu.xml`. En effet, si on disposait d'une page à part pour le plan du site (par exemple `plan.xml`), l'ajout d'un nouvel article sur le site impliquerait la mise à jour de `menu.xml` mais aussi de `plan.xml` ce qui n'est pas très efficace.

Contrairement au menu qui n'affichait que deux niveaux du fichier `menu.xml`, le plan du site en affiche trois. Comme le fichier `menu.xml` ne servira plus à rien d'autre, on peut donc affirmer avec certitude qu'il est inutile de descendre à des niveaux inférieurs à trois dans ce fichier, car les informations qu'ils contiendraient ne seront jamais significatives.

L'URI pour accéder au plan du site est `plansite`. Dans le sitemap, nous allons devoir traiter la transformation du menu en une page "normale" avant de pouvoir réaliser l'agrégation, la transformation et la sérialisation. Nous allons utiliser pour cela le concept de pipelines imbriqués :

```
<!-- pipeline qui est exécuté par la requête cocoon:/map -->
<map:match pattern="map">
  <map:generate src="page/menu.xml"/>
  <map:transform src="stylesheets/menu2page.xsl"/>
  <map:serialize type="xml"/>
</map:match>

<!-- pipeline principal -->
<map:match pattern="plansite">
  <map:aggregate element="document">
    <map:part src="page/menu.xml"/>
    <map:part src="cocoon:/map"/>
  </map:aggregate>
  <map:transform src="stylesheets/page.xsl"/>
  <map:serialize type="xhtml"/>
</map:match>
```

Le traitement est fort semblable à celui d'une page ordinaire. La différence est que nous agrégeons `menu.xml` avec le résultat du pipeline qui matche avec la requête `map` se trouvant dans le même sitemap (car le protocole est `cocoon:/`). Ce pipeline transforme `menu.xml` en une page normale en XML grâce à la feuille de style `menu2page.xsl`. Le résultat de la requête `map` donnera un résultat du type :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<page id="sitemap">
  <title>Plan du site</title>
  <content>
    <ul>
     <li><link type="internal" href="home">Accueil</link></li>
     <li><link type="internal" href="sitemap">Plan du site</link></li>
     <li><link type="internal" href="news">News</link></li>
     <li>Services
       <ul>
        <li><link type="internal" href="net_tel">Réseaux et téléphonie</link>
          <ul>
           <li><link type="internal" href="telecom">Réseaux et télécoms</link></li>
           <li><link type="internal" href="servers">Serveurs</link></li>
           <li><link type="internal" href="tel">Téléphonie</link></li>
          </ul>
        </li>
        <li><link type="internal" href="tech">Service technique</link></li>
        <li><link type="internal" href="web_wf">Web et workflows</link></li>
       </ul>
     </li>
     <li>Matériel
       <ul>
        <li><link type="internal" href="particulier">Particulier</link></li>
        <li><link type="internal" href="entreprise">Entreprise</link></li>
       </ul>
     </li>
     <li><link type="internal" href="files">Fichiers</link></li>
     <li><link type="internal" href="garanties">Garanties</link></li>
     <li><link type="internal" href="partenaires">Partenaires</link></li>
     <li><link type="internal" href="contact">Contact</link></li>
    </ul>
  </content>
</page>
```

Après agrégation, transformation et sérialisation, nous obtenons le résultat final :

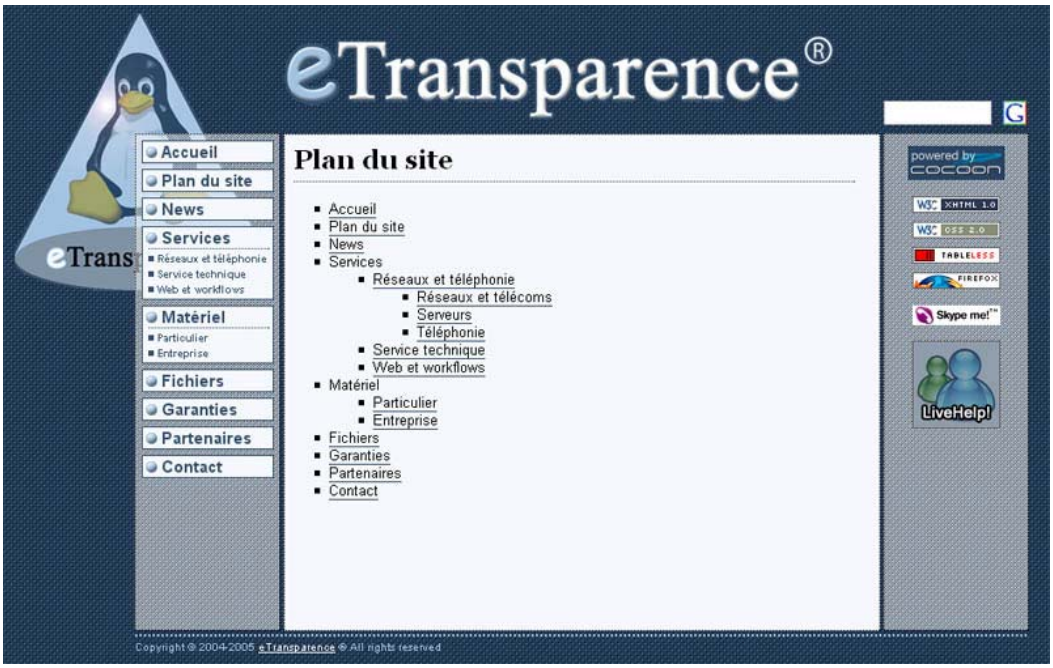

Figure 44 - Plan du site d'eTransparence

Le traitement menant à ce résultat est schématisé ci-dessous :

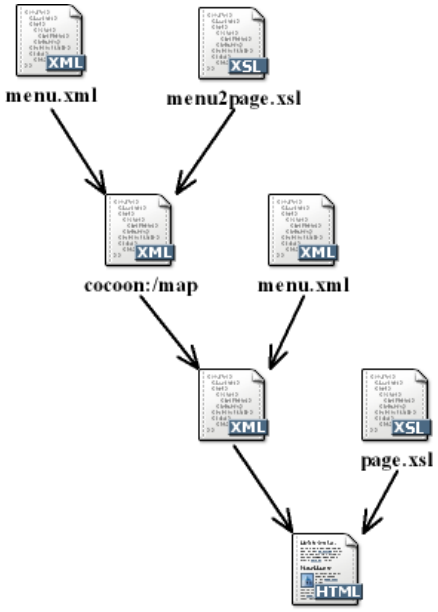

Figure 45 - Construction du plan du site

2. Fonctionnalités

2.1. Système de news (RSS)

Nous avons utilisé la technologie RSS pour importer les news de sites intéressants sur le site d'eTransparence.

Nous avons pour cela défini un fichier XML dans lequel sont enregistrées les URL de fichiers RSS sur le Web. En plus de l'URL, nous définissons également le titre, la langue et la version (0.9x, 1.0 ou 2.0) du fichier RSS. Celle-ci doit être obligatoirement connue car elle permet de déterminer la feuille de style XSLT que nous devons appliquer, car nous avons vu que chacune des trois versions possède sa propre feuille de style. Vous pouvez trouver celles-ci en annexe iii.

Ce fichier se présente de la façon suivante :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!-------------------->
<!----- news.xml ----->
<!-------------------->
<page id="news">
 <title>News</title>
 <content>
  <feeds>
   <feed version="2.0" lang="fr">
    <label>Matbe.com : Articles</label>
    <url>www.matbe.com/export/rss_articles.xml</url>
   </feed>
   <feed version="0.9x" lang="fr">
    <label>Web Rank Info</label>
    <url>www.webrankinfo.com/rss.php</url>
   </feed>
   <feed version="1.0" lang="en">
    <label>XML.com</label>
    <url>www.oreillynet.com/meerkat/?_fl=rss10&amp;t=ALL&amp;c=47</url>
   </feed>
   <feed version="0.9x" lang="fr">
    <label>ZDNet News</label>
    <url>www.zdnet.fr/feeds/rss/actualites</url>
   </feed>
  </feeds>
 </content>
</page>
```

Chaque fil RSS est identifié par une version (attribut `version`), une langue (attribut `lang`), un nom (élément `<label/>`) et le lien vers le fichier distant (élément `<url/>`).

Cette page est affichée grâce à la feuille de style `feed.xsl` :

```
<?xml version="1.0"?>
<!-------------------->
<!----- feed.xsl ----->
<!-------------------->
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/1999/xhtml">
```

```
<xsl:include href="page.xsl"/>

<!-- création de la table contenant tous les fils -->
<xsl:template match="feeds">
 <table id="feeds">
  <xsl:apply-templates/>
 </table>
</xsl:template>

<!-- affichage d'une rangée de la table
     contenant les informations d'un fil -->
<xsl:template match="feed">
 <tr class="feed">
  <td class="flag-feed">
   <img class="noborder" alt="Lang">
   <xsl:attribute name="src">
    ./images/<xsl:value-of select="@lang"/>.gif
   </xsl:attribute>
   </img>
  </td>
  <td class="feed">
  <a class="ternal" rel="external">
  <xsl:attribute name="href">
   ./news/<xsl:value-of select="@version"/>/<xsl:value-of select="url"/>
  </xsl:attribute>
   <xsl:value-of select="label"/>
  </a>
  </td>
 </tr>
</xsl:template>

</xsl:stylesheet>
```

Cela va nous créer une table où chaque rangée est composée du drapeau de la langue et le nom du fil RSS avec un lien vers celui-ci de la forme `news/[VERSION]/[URL]`. Pour le fil RSS de "Web Rank Info" par exemple, nous obtenons le lien `news/0.9x/www.webrankinfo.com/rss.php`.

La figure ci-dessous nous donne un aperçu de la page "News" :

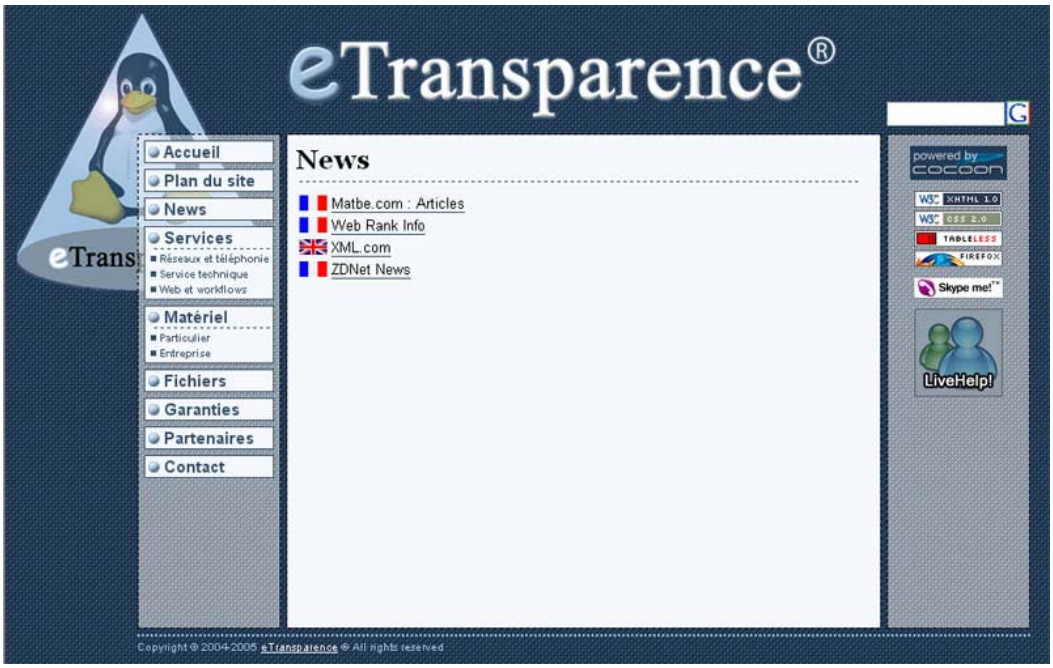

Figure 46 - Aperçu de la page "News"

Dans le sitemap, la requête `news/[VERSION]/[URL]` va matcher avec le pipeline suivant :

```
<map:match pattern="news/*/**">
  <map:generate src="http://{2}"/>
  <map:select type="parameter">
    <map:parameter name="parameter-selector-test" value="{1}"/>
    <map:when test="0.9x">
      <map:transform src="stylesheets/rss/rss09x.xsl"/>
    </map:when>
    <map:when test="1.0">
      <map:transform src="stylesheets/rss/rss1.xsl"/>
    </map:when>
    <map:when test="2.0">
      <map:transform src="stylesheets/rss/rss2.xsl"/>
    </map:when>
  </map:select>
  <map:serialize type="xhtml"/>
</map:match>
```

Dans un premier temps, nous générons le fichier RSS distant. Ensuite, nous utilisons un selector qui va appliquer une feuille de style différente en fonction de la version du fichier RSS. Pour terminer, nous sérialisons au format XHTML.

Si nous cliquons sur "Web Rank Info" par exemple, nous obtenons le résultat suivant :

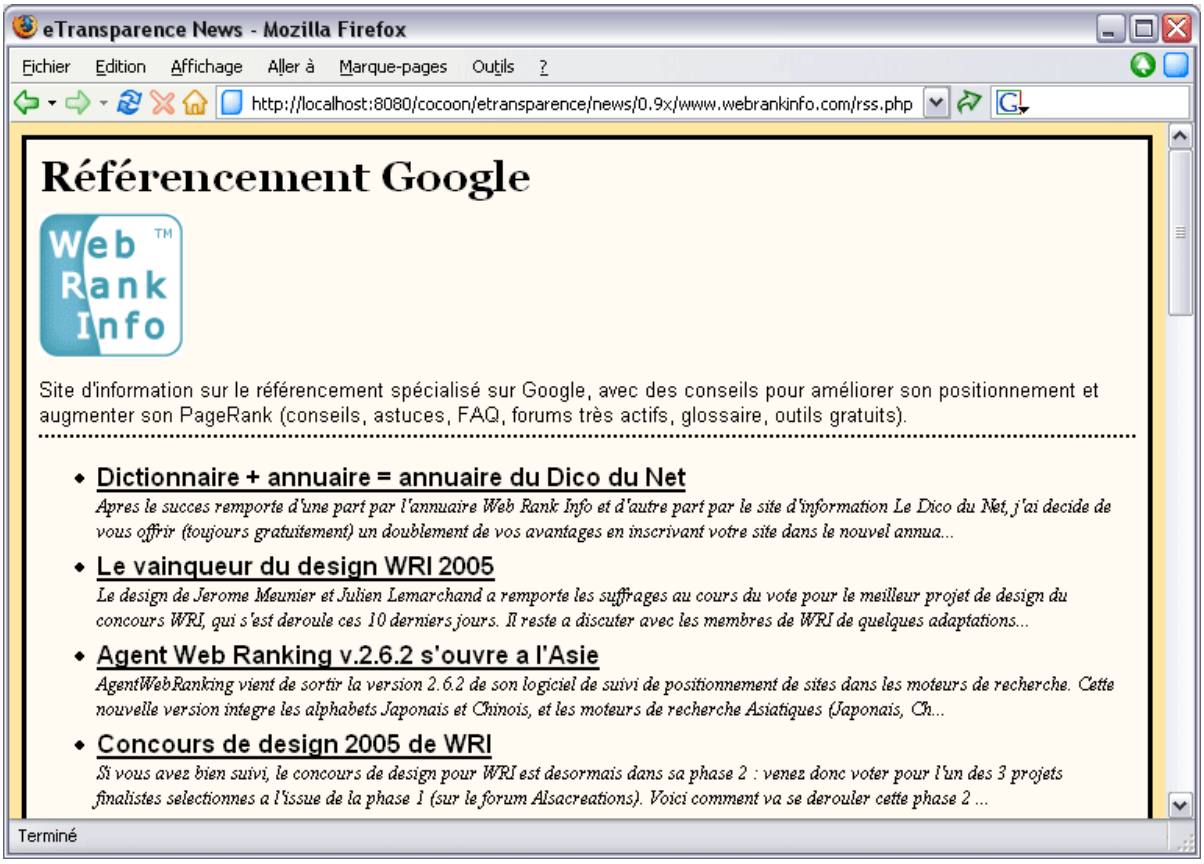

Figure 47 - Fil RSS sur le site d'eTransparence

Le travail de mise à jour des fils RSS que nous souhaitons inclure sur le site est très simple. Il suffit de modifier le fichier `news.xml` et de rajouter des éléments `<feed/>`.

Quelques améliorations peuvent être apportées au système actuel. D'une part, il serait plus simple pour l'administrateur de remplir un formulaire d'ajout au lieu de modifier directement le fichier `news.xml`. D'autre part, on pourrait détecter automatiquement la langue, la version et le titre du fil RSS dans le fichier distant. De cette manière, le fichier `news.xml` ne contiendrait plus que des URL ce qui apporterait un gain de temps non négligeable pour la maintenance et qui permettrait d'éviter certains problèmes (par exemple, si la version entrée par l'administrateur ne correspond pas à la version réelle du fichier).

2.2. Zone téléchargements

Une page du site permet aux visiteurs de télécharger des fichiers divers : articles en PDF, vidéos, etc.

Nous réalisons cela grâce au *DirectoryGenerator* qui liste tous les fichiers se trouvant dans un répertoire donné. Le répertoire que nous avons choisi est le répertoire `files`. Cette méthode permet de simplifier considérablement l'ajout de nouveaux fichiers. En effet, il suffit de placer chaque nouveau fichier dans le répertoire `files`, et la liste se met à jour automatiquement.

Comme nous avons déjà détaillé le *Mp3directoryGenerator* dans l'exemple récapitulatif, nous ne reviendrons pas sur le fonctionnement interne du *DirectoryGenerator* ici.

Le résultat final est le suivant :

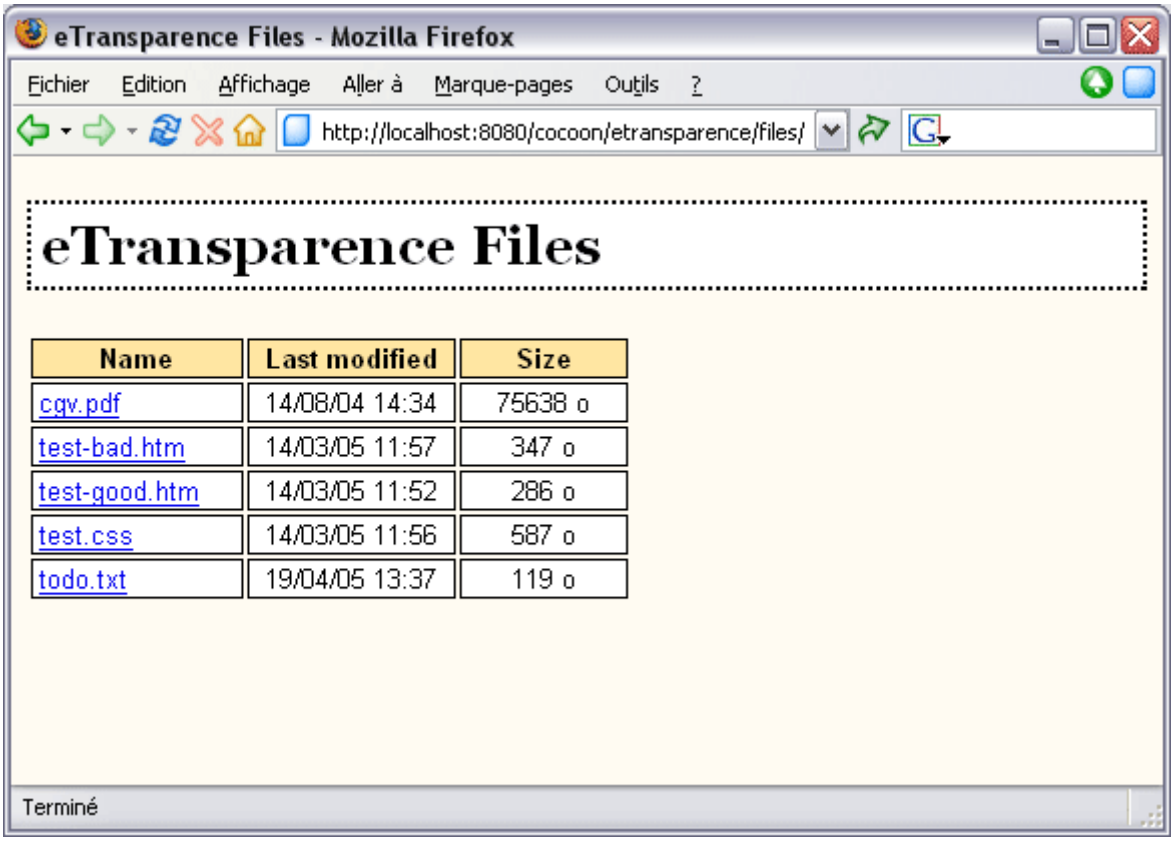

Figure 48 - Zone téléchargements

2.3. Moteur de recherche interne

Le site possède un moteur de recherche qui permet de faire des recherches internes au site.

Pour effectuer la recherche, nous faisons appel à Google. Ce choix est motivé par deux choses. Premièrement, cela nous permet de ne pas avoir à implémenter le moteur de recherche. Deuxièmement, nous bénéficions de la puissance d'indexation de Google.

Sur le site d'eTransparence, le moteur de recherche interne se présente sous la forme d'un champ en haut à droite de chaque page :

Figure 49 - Moteur de recherche interne

Lors du click sur le "G", nous sommes redirigés vers la page http://www.google.be/search?q=[KEYWORD]&sitesearch=www.etransparence.be où la valeur de q est le mot clé recherché et où la valeur de `sitesearch` est le domaine sur lequel porte la recherche. Comme le nom de domaine d'eTransparence est www.etransparence.be, nous faisons bien une recherche interne.

D'un point de vue interne, cette recherche est réalisée avec le simple formulaire HTML suivant :

```
<form action="http://www.google.be/search" method="get">
  <input type="text" name="q" size="12" maxlength="100"/>
  <input type="hidden" name="sitesearch" value="www.etransparence.be"/>
  <input type="image" src="images/g-submit.gif"/>
</form>
```

Voici un exemple de résultat avec le mot clé "serveurs" :

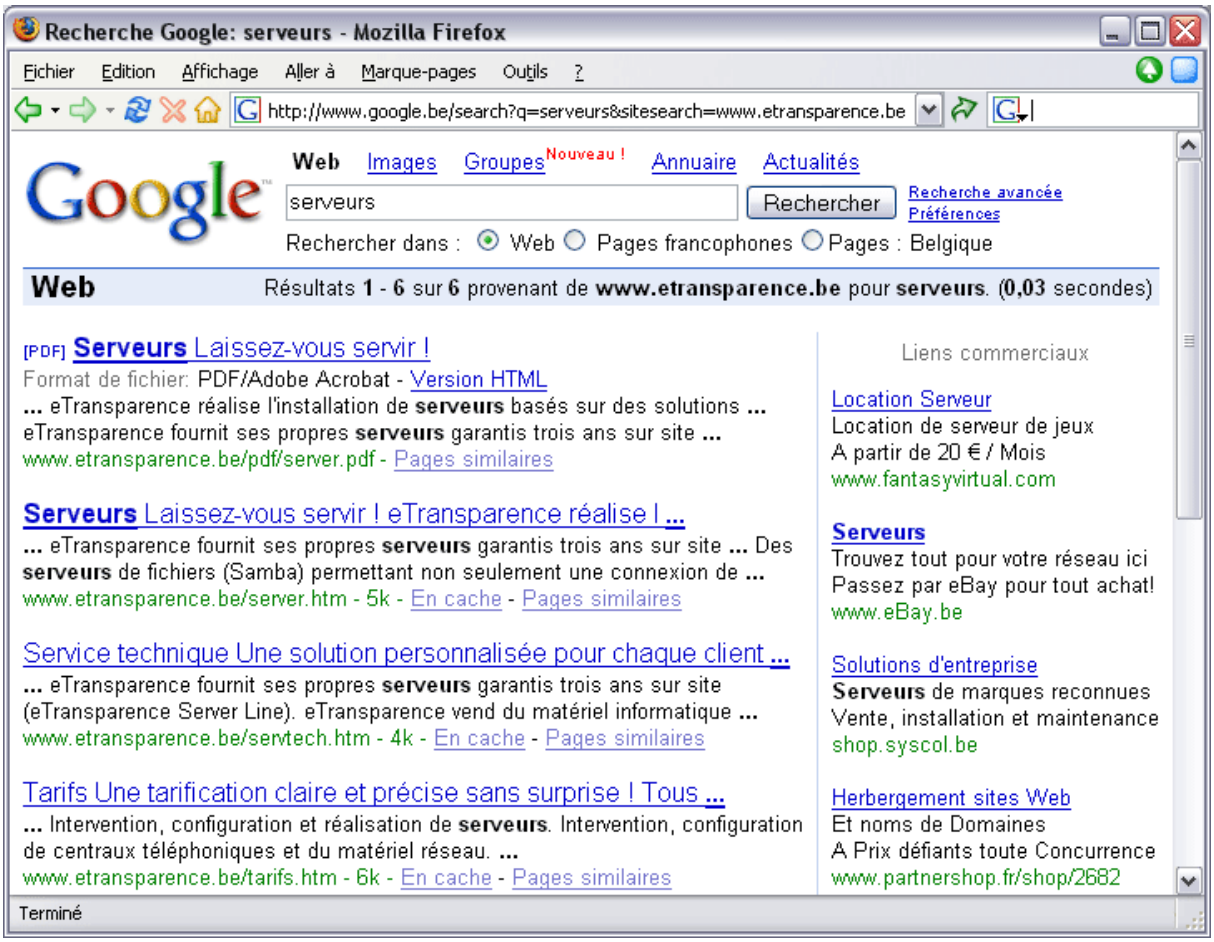

Figure 50 - Résultat d'une recherche interne

L'inconvénient de cette méthode est que nous sommes redirigés vers une page externe (Google Belgique) qui possède une charte graphique différente du site d'eTransparence ce qui pourrait troubler les visiteurs.

Une meilleure solution serait d'utiliser l'API de Google (http://www.google.com/apis/) qui est développée en Java et qui utilise SOAP pour le transit d'informations. Elle permet d'inclure

directement les résultats de Google à l'intérieur du site avec l'habillage que l'on souhaite. Pankaj Kumar [23] présente une mise en oeuvre de cette API avec Cocoon. Cette méthode est encore en phase d'étude et sera mise en oeuvre prochainement sur le site d'eTransparence.

2.4. Vente de matériel

Pour la vente de matériel, nous avons décidé de créer un fichier XML par article que vend eTransparence. Les types d'articles possibles sont les ordinateurs de bureau, les ordinateurs portables et les infrastructures sans fil. Tous ces fichiers sont placés dans le répertoire `materiel`.

Ces fichiers doivent respecter une DTD créée par nos soins. Le canevas de ces fichiers est le suivant :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE article SYSTEM "matos.dtd">
<article>
 <name>eTPC Standard Edition<!-- Nom de l'article --></name>
 <carac>
  <config-mat>
  <!-- Configuration mat&eacute;rielle (sur plusieurs lignes) -->
   <line></line>
   <line></line>
   <!-- ... -->
  </config-mat>
  <config-log>
  <!-- Configuration logicielle (sur plusieurs lignes) -->
   <line></line>
   <!-- ... -->
  </config-log>
 </carac>
 <guar>
 <!-- Garanties (sur plusieurs lignes) -->
  <line></line>
  <!-- ... -->
 </guar>
 <price type="ttc" currency="euro">
  959<!-- prix de l'article -->
 </price>
</article>
```

Le contenu de la DTD (`matos.dtd`) est le suivant :

```
<!ELEMENT article (name,carac,guar,price)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT carac (config-mat,config-log)>
<!ELEMENT config-mat (line)*>
<!ELEMENT config-log (line)*>
<!ELEMENT guar (line)*>
<!ELEMENT line (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ATTLIST price
          type (ttc|htva) #REQUIRED
          currency (euro|dollar) #IMPLIED>
```

Une fois que tous nos fichiers sont créés dans le répertoire `materiel`, nous les listons avec le *XpathdirectoryGenerator* qui fonctionne comme le *DirectoryGenerator* à la différence qu'il

permet de faire des requêtes XPath sur les fichiers. Ce generator s'applique donc uniquement aux répertoires de fichiers XML, comme c'est le cas ici.

Si nous souhaitons lister les fichiers en affichant le nom et le prix de l'article, nous utiliseront le paramètre suivant du generator :

```
<map:generate type="xpathdirectory" src="materiel">
  <map:parameter name="xpath" value="//name|//price"/>
</map:generate>
```

Cela aura pour effet d'effectuer la requête XPath contenue dans l'attribut `value` sur le fichier XML.

La sortie du générateur sera de ce type :

```
<dir:directory name="materiel" lastModified="1116698458468"
               date="21/05/05 20:00" size="0" sort="name"
               reverse="false" requested="true">
  <dir:file name="matos.dtd" lastModified="1116699155406" date="21/05/05 20:12"
size="484"/>
  <dir:file name="pc1.xml" lastModified="1116700297093" date="21/05/05 20:31"
size="774">
    <dir:xpath query="//name">
      <name>
        eTPC Standard Edition
      </name>
      <price currency="euro" type="ttc">
        959
      </price>
    </dir:xpath>
  </dir:file>
  <!-- ... -->
</dir:directory>
```

Il ne reste plus à présent qu'à appliquer une feuille de style XSLT et de sérialiser au format XHTML. L'output que nous obtenons est une liste en HTML :

```
<ul>
 <li><a href="pc1.xml.html">eTPC Standard Edition</a> (959 € TTC)</li>
 <!-- ... -->
</ul>
```

Pour afficher les caractéristiques complètes de l'article, il suffit de cliquer sur un élément de la liste. Cela aura pour effet de transformer le fichier XML en HTML grâce à l'utilisation d'une feuille de style XSLT :

```
<map:match pattern="materiel/*.html">
  <map:generate src="materiel/{1}"/>
  <map:transform src="stylesheets/materiel.xsl"/>
  <map:serialize type="xhtml"/>
</map:match>
```

2.5. Formulaire de contact

Via un formulaire de contact, les visiteurs peuvent laisser un message qui sera envoyé par mail à l'adresse info@etransparence.be de manière tout à fait invisible.

L'envoi du mail se fait grâce à l'action *Sendmail Action* dans le sitemap de Cocoon. Nous déclarons ce composant de la façon suivante :

```
<map:components>
  <map:actions>
    <map:action name="sendmail" src="org.apache.cocoon.acting.Sendmail"/>
  </map:actions>
</map:components>
```

Dans la page "Contact", nous plaçons simplement un formulaire HTML qui se compose de deux champs : un pour l'adresse électronique de la personne émettrice et le second pour le message proprement dit :

```
<form action="contact/sendmail" method="POST">
  <input type="text" name="email" size="30" value="Votre e-mail ici..."/>
  <textarea name="message" rows="10" cols="35">Votre message ici...</textarea>
</form>
```

Dans le sitemap, l'URI `contact/sendmail` (voir attribut `action` du formulaire ci-dessus) va matcher avec le pipeline suivant :

```
<map:match pattern="contact/sendmail">
  <map:act type="sendmail">
    <map:parameter name="smtp-host" value="mail.etransparence.be"/>
    <map:parameter name="smtp-user" value="etransparence"/>
    <map:parameter name="smtp-password" value=""/>
    <map:parameter name="from" value="{request-param:email}"/>
    <map:parameter name="to" value="info@etransparence.be"/>
    <map:parameter name="subject" value="Message from : {request-param:email}"/>
    <map:parameter name="body" value="{request-param:message}"/>
    <map:generate src="page/sendmail/{status}.xml"/>
    <map:transform src="stylesheets/page.xsl"/>
    <map:serialize type="xhtml"/>
  </map:act>
</map:match>
```

Ce pipeline va permettre l'envoi du mail. Nous définissons tout d'abord l'hôte (`smtp-host`), le nom d'utilisateur (`smtp-user`) et le mot de passe (`smtp-password`) du serveur SMTP. Ensuite, nous déterminons l'émetteur (`from`), le destinataire (`to`), le sujet (`subject`) et le corps (`body`) de l'e-mail. Nous aurions aussi pu ajouter le paramètre "copie conforme" (`cc`) et le paramètre "copie conforme invisible" (`bcc`). Seuls les paramètres `from` et `to` sont requis.

`{request-param:email}` et `{request-param:message}` correspondent aux valeurs que le visiteur a entrées dans le formulaire. `{status}` est le code de retour de l'envoi du mail. Il peut prendre trois valeurs : `success`, `user-error` ou `server-error`. `success` signifie que l'e-mail a été délivré avec succès. `user-error` signifie qu'il y a un problème avec au moins une des adresses spécifiées (`to`, `cc`, `bcc` ou `from`). Enfin, `server-error` signifie qu'un problème du serveur est survenu lors de l'envoi de l'e-mail.

En fonction de cette valeur de retour, nous générons un fichier XML différent. Par exemple, le fichier `success.xml` contient le message suivant :

```
Votre e-mail a été envoyé avec succès !
```

Les fichiers `user-error.xml` et `server-error.xml` contiennent des messages d'erreur.

2.6. Version WAP

Une version WAP est disponible pour les personnes disposant de ce service sur leur téléphone portable. Cette version est complètement indépendante du site original car l'écran minuscule d'un mobile ne se prête pas à la lecture de longs textes. Le site WAP est en quelque sorte un résumé bref et concis du site original.

Le langage pour le WAP est le WML, qui est basé sur XML et qui possède une syntaxe proche de HTML. On peut noter que WML est doté de son propre format d'image, appelé Wireless Bitmap (WBMP), dérivé du format BMP, mais en noir et blanc.

Le WML n'est pas maintenu par le W3C comme on aurait pu s'y attendre mais par un autre organisme nommé Wap Forum (http://www.wapforum.org).

Le WML se base sur la notion de "cartes" (*cards*). Un fichier WML se compose de plusieurs cartes interconnectées qui correspondent chacune à une page du site WAP. Un fichier WML est donc nommé à juste titre "paquet de cartes" (*deck*). Le schéma ci-dessous donne un aperçu graphique d'un fichier WML :

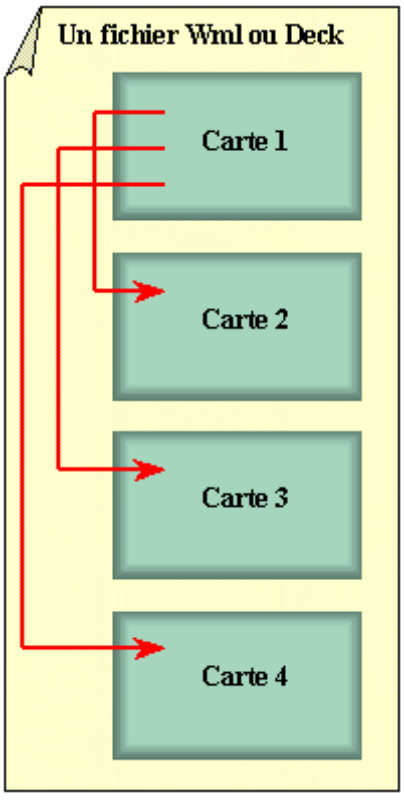

Figure 51 - Aperçu graphique d'un fichier WML

Le site WAP d'eTransparence se compose de quatre cartes. La première carte inclut le logo au format WBMP, la seconde affiche le menu permettant d'accéder directement à n'importe quelle carte, la troisième donne une présentation de la société et de ses activités et enfin, la dernière carte reprend les coordonnées de la société (adresses, n° de téléphone, etc.).

Le fichier WML du site est le suivant :

```
<?xml version="1.0"?>
<!-------------------->
<!----- wap.wml ----->
<!-------------------->
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
                     "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>

 <card id="card1" title="Bienvenue !">
  <p>
   <img alt="logo" src="logo.wbmp"/>
   <br/><a href="#card2">Continuer >>></a>
  </p>
 </card>

 <card id="card2" title="Menu">
  <p>
   <a href="#card1">Accueil</a>
   <br/><a href="#card2">Menu</a>
   <br/><a href="#card3">Présentation</a>
   <br/><a href="#card4">Coordonnées</a>
  </p>
 </card>

 <card id="card3" title="Présentation">
  <p>
   eTransparence est une société informatique basée à Binche (Leval) en
   Belgique qui conçoit et réalise tous vos projets informatiques !
  </p>
  <p>
   Nos activités vont de la vente et de l'installation de PC et infrastructures
   sans-fil à la création de logiciels et sites Web, en passant par
   l'installation de serveurs et de réseaux, aussi bien pour les particuliers
   que pour les entreprises.
  </p>
  <p>
   <a href="#card2"><<< Retour au menu</a>
  </p>
 </card>

 <card id="card4" title="Coordonnées">
  <p>
   Rue des Boulois, 218
   <br/>7134 Binche (Leval)
   <br/>Hainaut - BELGIQUE
  </p>
  <p>
   Téléphone : +32 (0) 64 21 22 28
   <br/>Fax : +32 (0) 64 21 22 83
   <br/>E-Mail : info@etransparence.be
   <br/>WAP : http://wap.etransparence.be
   <br/>WWW : http://www.etransparence.be
  </p>
  <p>
   <a href="#card2"><<< Retour au menu</a>
  </p>
 </card>

</wml>
```

Chaque carte (élément `<card/>`) a comme attributs un identifiant (`id`) et un titre (`title`). Pour faire un lien vers une carte, on utilise le caractère # suivi par l'identifiant de la carte vers

laquelle on souhaite faire le lien, par exemple `#card1`. L'interconnexion des cartes de notre site WAP est représentée sur le schéma ci-dessous :

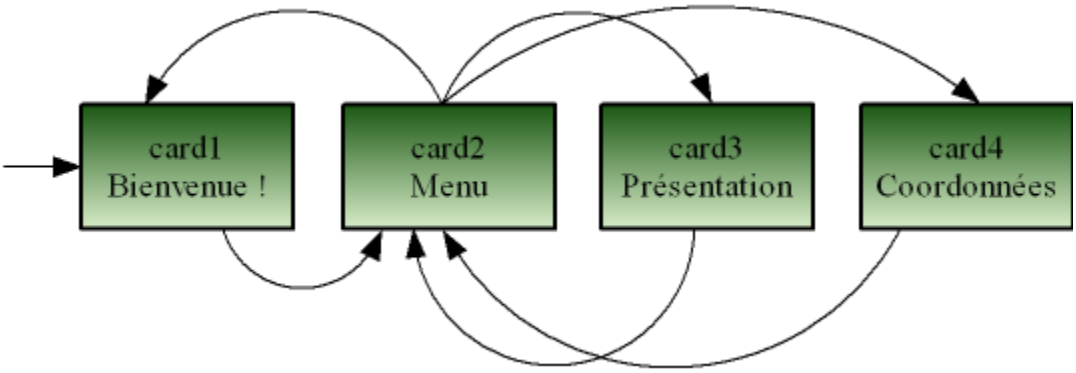

Figure 52 - Interconnexion des cartes du site WAP

Si on ne dispose pas d'un téléphone portable muni du service WAP, on peut soit utiliser un navigateur WAP, soit un émulateur WAP. Le navigateur se contente de lire les pages en WML et de les afficher alors que l'émulateur simule l'affichage d'un téléphone portable.

Le navigateur qui a été utilisé pour les tests est WinWap (http://www.winwap.com). Voici l'aperçu de la page d'accueil avec ce navigateur :

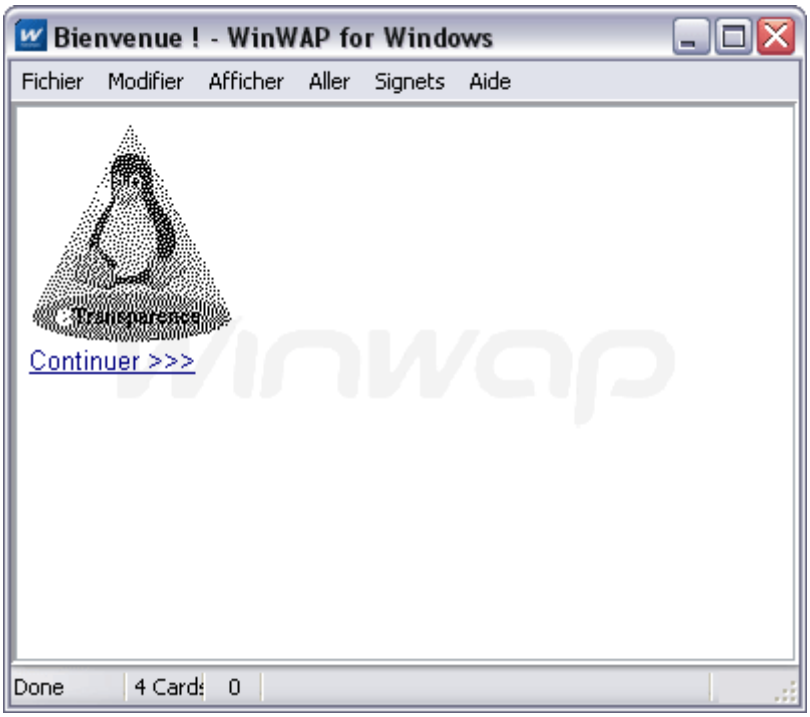

Figure 53 - Aperçu du site WAP dans un navigateur

Un émulateur donne une meilleure idée de ce que pourrait donner le site sur un téléphone portable. Nous avons utilisé l'émulateur en ligne TagTag (http://www.tagtag.com) pour nos tests. Voici l'aperçu du site entier avec cet émulateur :

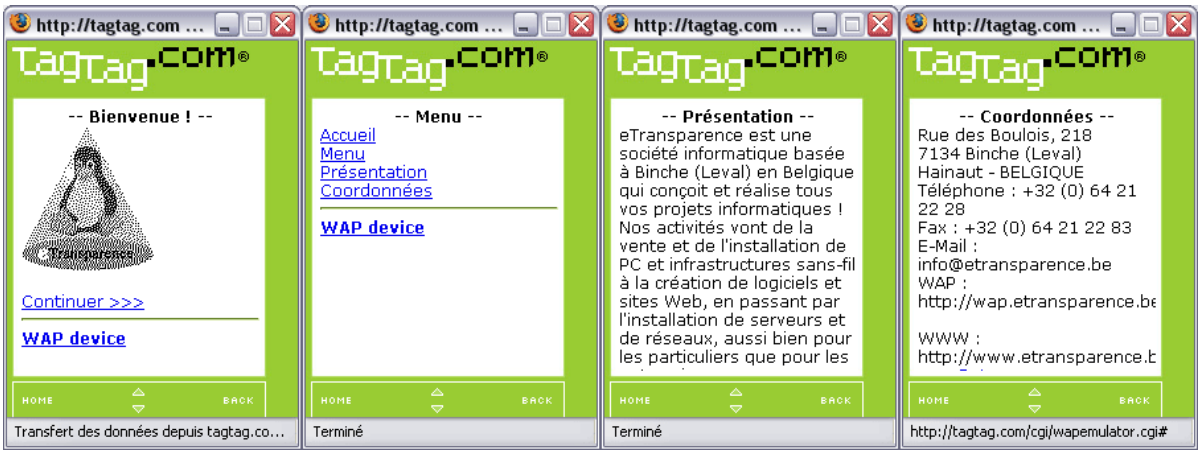

Figure 54 - Aperçu du site WAP dans un émulateur

Pour que la page WML puisse s'afficher avec Cocoon, nous avons utilisé un reader :

```
<map:match pattern="wap/">
  <map:read mime-type="text/vnd.wap.wml" src="wap/wap.wml"/>
</map:match>
```

Le site WAP sera disponible à l'URL http://wap.etransparence.be.

2.7. Version PDA

Le site possède une version pour les agendas électroniques (que l'on appelle aussi PDA ou Pocket PC). Ce type d'appareils est capable de lire des pages en HTML, nous n'avons donc pas été contraints d'utiliser un langage spécial comme c'était le cas pour la version WAP. La différence entre la version PDA et le site original est seulement graphique, le contenu reste le même. Cette version a été simplifiée graphiquement et réduite en taille pour être mieux lisible sur le petit écran de l'appareil.

Pour élaborer cette version, nous avons tout simplement utilisé une feuille de style XSLT différente pour afficher les pages.

N'ayant pas de PDA sous la main, nous avons utilisé un émulateur grandeur nature. Celui-ci se nomme *Calista PDA Emulator*.

Voici ci-dessous un aperçu du site original avec l'émulateur :

Figure 55 - Aperçu du site dans un émulateur PDA (avant)

Nous remarquons directement qu'il y a un problème. La navigation n'est pas ergonomique, il faut faire défiler les scrollbars vers le bas et vers la gauche pour pouvoir voir la page dans sa totalité.

Dans la version PDA, le menu n'est plus inclus sur chaque page. Le visiteur est invité à passer par le plan du site pour naviguer de page en page. Un lien vers le plan du site est donc disponible sur chaque page. L'interface graphique est volontairement dépouillée pour plus de lisibilité. Le résultat que nous obtenons est conforme aux attentes :

Figure 56 - Aperçu du site dans un émulateur PDA (après)

Dans Cocoon, nous utilisons le pipeline ci-dessous :

```
<map:match pattern="pda/*">
  <map:generate src="page/{1}.xml"/>
  <map:transform src="stylesheets/pda.xsl"/>
  <map:serialize type="xhtml"/>
</map:match>
```

La version PDA sera disponible sur http://pda.etransparence.be.

3. Travaux futurs

Étant donné la quantité importante de travail qui a été demandée, certaines parties sont encore en chantier, pour diverses raisons.

L'espace administrateur et l'espace client n'ont pas encore été créés car ils n'ont pas encore lieu d'exister à ce stade. La partie scientifique ne fait pas encore partie du site car elle n'est pas encore au point et doit être nettement améliorée.

3.1. Espace administrateur

L'espace administrateur sera une boîte à outils destinée à simplifier la maintenance et l'administration du site. Il sera par exemple possible d'ajouter un fil RSS via un formulaire, de modifier les pages grâce à une interface WYSIWYG, d'uploader de nouveaux fichiers dans la zone téléchargements, etc.

Bien entendu, l'espace administrateur ne doit pas être disponible à n'importe quel visiteur passant par là. Il est donc important de se doter d'une protection efficace contre les indésirables.

Le framework d'authentification de Cocoon (*Cocoon Authentication Framework*) est un module flexible pour l'authentification, les autorisations et la gestion des utilisateurs. L'authentification d'un utilisateur peut se faire en utilisant des bases de données relationnelles ou XML, des annuaires LDAP, etc.

Le principe de base de ce framework est de protéger les documents générés par Cocoon. Par document, on entend le résultat d'une requête faite à Cocoon. Cela peut être le résultat d'un pipeline ou d'un reader défini dans le sitemap.

Pour plus d'informations :
http://cocoon.apache.org/2.1/developing/webapps/authentication.html

3.2. Espace client

L'espace client sera un espace personnel disponible pour chaque client d'eTransparence. Cet espace permettra aux clients de consulter l'état de leurs achats et de leurs commandes, de voir leurs statistiques personnelles, etc.

L'espace client utilisera le même mécanisme d'authentification que celui de l'espace administrateur.

3.3. Partie scientifique

La partie scientifique est consacrée à définir et expliquer certains termes et certaines technologies mentionnées dans les pages de la partie commerciale.

Remarque : cette partie nécessite des notions de compilation pour être parfaitement comprise.

3.3.1. Wikipédia

Au lieu d'écrire les articles nous-même, nous avons décidé de les importer de Wikipédia sur le site d'eTransparence. Wikipédia est une encyclopédie libre, gratuite, universelle et multilingue, écrite par des volontaires et basée sur un site Web (http://fr.wikipedia.org en français) utilisant la technologie wiki.

Chaque article de Wikipédia est disponible à l'adresse http://fr.wikipedia.org/wiki/[NOM]. Par exemple, l'article à propos de Linux est consultable sur http://fr.wikipedia.org/wiki/Linux. Un article de Wikipédia se présente de la manière suivante :

Figure 57 - Page type du site Wikipédia

De plus, chaque page dispose d'une version exportable à l'adresse http://fr.wikipedia.org/wiki/Special:Export/[NOM]. Cette version exportable possède une syntaxe hybride semi-XML, semi-wiki. Pour mieux nous en rendre compte, voici le canevas de ces fichiers :

```
<?xml version="1.0" encoding="UTF-8" ?>
<mediawiki version="0.1" xml:lang="fr">
  <page>
    <title>Tux</title>
    <revision>
      <timestamp>2005-05-09T15:33:06Z</timestamp>
      <contributor>
        <username>Vargenau</username>
      </contributor>
      <minor/>
      <comment>[[Tyrol]]</comment>
      <text>
       <!-- source de l'article en syntaxe wiki -->
      </text>
    </revision>
  </page>
</mediawiki>
```

Quelques informations d'entête se trouvent dans des balises XML mais le contenu proprement dit est écrit en syntaxe wiki à l'intérieur de l'élément `<text/>`. C'est pour cette raison qu'on parle de syntaxe hybride.

La seule information qui nous intéresse est celle qui se trouve à l'intérieur de l'élément `<text/>`. Pour l'afficher sur le site d'eTransparence, nous devons donc transformer la syntaxe wiki en XML, ou mieux, en XHTML.

3.3.2. Chaperon

L'idée qui a été retenue[10] pour effectuer la transformation de la syntaxe Wiki en XML est l'utilisation de Chaperon (http://sourceforge.chaperon.org). Il s'agit d'un projet Open Source permettant de transformer un texte structuré (Java, TeX, Wiki, etc.) en XML. Il inclut un puissant parseur Canonical LALR et un tree builder (constructeur d'arbre) pour créer les fichiers XML. Chaperon est inclus dans la distribution de base de Cocoon.

Chaperon nécessite un fichier lexical et un fichier grammatical pour s'exécuter. Ces fichiers possèdent une syntaxe XML. Le fichier lexical (d'extension `.xlex`) sert à définir des lexèmes tandis que le fichier grammatical (d'extension `.xgrm`) sert à définir la grammaire utilisée. Des exemples de tels fichiers sont disponibles en annexe iv.

La présence de Chaperon dans la distribution de Cocoon se manifeste par l'existence de deux transformers, le *LexicalTransformer* et le *ParserTransformer*. Le *LexicalTransformer* sert à transformer un fichier texte (dans notre cas, un fichier texte en syntaxe wiki) en une liste de lexèmes à l'aide du fichier lexical. Le *ParserTransformer* transforme cette suite de lexèmes en un arbre syntaxique XML grâce au fichier grammatical.

Ces deux transformers doivent être déclarés dans le sitemap :

```
<map:components>
  <map:transformers>
    <map:transformer name="lexer"
         src="org.apache.cocoon.transformation.LexicalTransformer"/>
    <map:transformer name="parser"
         src="net.sourceforge.chaperon.adapter.cocoon.ParserTransformer"/>
  </map:transformers>
</map:components>
```

Les étapes pour la transformation de l'article distant sont les suivantes :

1. Transformation du fichier hybride en fichier texte ne contenant plus que la syntaxe wiki
2. Transformation du fichier texte en une liste de lexèmes en utilisant le *LexicalTransformer*
3. Transformation de la liste de lexèmes en un arbre en utilisant le *ParserTransformer*
4. Transformation de l'arbre XML en XHTML en utilisant une feuille de style XSLT
5. Sérialisation au format XHTML

[10] Notre première idée était d'utiliser l'outil Wiki2xhtml mais nous avons malheureusement du l'abandonner car il s'agit d'une classe en PHP qui n'est donc pas exécutable avec notre serveur Tomcat.

Cela se traduit de la manière suivante dans le pipeline :

```
<map:match pattern="wikipedia/*">
  <map:generate type="text" src="cocoon:/wikipedia2txt/{1}"/>
  <map:transform type="lexer" src="chaperon/lexicon.xlex"/>
  <map:transform type="parser" src="chaperon/grammar.xgrm"/>
  <map:transform src="stylesheets/wikipedia.xsl"/>
  <map:serialize type="xhtml"/>
</map:match>

<map:match pattern="wikipedia2txt/*">
  <map:generate src="http://fr.wikipedia.org/wiki/Special:Export/{1}"/>
  <map:transform src="stylesheets/wikipedia2txt.xsl"/>
  <map:serialize type="text"/>
</map:match>
```

La feuille de style `wikipedia2txt.xsl` permet d'extraire le contenu wiki (pour rappel, il se trouve dans l'élément `<text/>`) du fichier distant et la feuille de style `wikipedia.xsl` permet de transformer le XML pour l'affichage final en XHTML.

Malheureusement, cette fonctionnalité ne marche pas encore assez bien pour pouvoir être utilisée sur le site. Certains éléments ne s'affichent pas correctement et d'autres ne sont pas bien traités. En particulier, les fichiers `lexicon.xlex` et `grammar.xgrm` doivent être optimisés.

3.4. Et après?

La particularité d'un site Web est de toujours évoluer, grâce à de nouveaux contenus et à de nouvelles fonctionnalités. Un site Web qui n'évolue pas ou plus peut être considéré comme mort.

Dans cette optique, la réalisation du site d'eTransparence ne s'arrête pas au présent travail et devra être prise en main par des administrateurs et des développeurs qualifiés dans l'avenir (proche). Le site n'est pas encore complètement terminé mais on peut comprendre aisément qu'il ne le sera jamais, car de nouvelles technologies impliqueront de nouvelles idées qui mèneront à de nouvelles fonctionnalités, et ainsi de suite. Cependant, dans l'état actuel des choses, le site est prêt à être mis en ligne dans ce qu'on pourrait appeler sa "version 1.0".

La mise en ligne du site ne sera pas immédiate car il faudra dans un premier temps réserver un nouvel hébergeur supportant Tomcat et Cocoon (l'actuel hébergeur d'eTransparence est de type LAMP) et ensuite transférer le nom de domaine http://www.etransparence.be vers ce nouvel hébergeur, c'est-à-dire mettre à jour les tables DNS du monde entier.

C'est pour cette raison que si vous consultez http://www.etransparence.be pour l'instant, vous serez en présence de l'ancien site, réalisé dans le cadre de mon stage avec les technologies PHP et MySQL.

Chapitre 9. Conclusion générale

Ce travail avait pour but de créer le site Web d'eTransparence avec des technologies libres (Open Source), en utilisant le plus possible le langage XML et ses dialectes. Après des recherches approfondies des outils existants, il a été décidé d'utiliser le framework Cocoon, qui permet de créer des applications Web (sites Internet, Intranet) avec l'aide de XML et XSLT. Cocoon a de nombreux avantages, parmi lesquels la séparation des tâches (basée sur le patron de conception MVC) et la publication en de nombreux formats (HTML, PDF, XML, etc.). Pour créer une application Web avec Cocoon, il faut déclarer des pipelines de composants dans un fichier spécial nommé le sitemap. Les composants ont chacun un rôle particulier et leur assemblage permet de réaliser le traitement complet d'une requête (qui s'exprime sous forme d'URI). La conception d'applications avec Cocoon est donc très simple, il suffit de déclarer les composants dont on a besoin dans le sitemap.

L'exemple récapitulatif et l'implémentation du site d'eTransparence nous ont donné un bon aperçu des possibilités énormes de Cocoon, et de manière plus générale, des possibilités de la création d'applications Web en XML. Pour aller plus loin, il serait intéressant d'étudier l'aspect développement qui consiste à créer ou modifier des composants de Cocoon avec le langage Java.

A l'heure actuelle, Cocoon a du mal à s'imposer face à des ogres comme PHP et dans une moindre mesure JSP. Ce ne sont pourtant pas les qualités qui manquent à Cocoon, mais le PHP, présent depuis très longtemps s'est imposé comme un standard de fait pour la création de sites Web dynamiques, malgré ses lacunes dans certains secteurs.

Cocoon devrait cependant s'imposer dans le futur, aussi bien chez le particulier qu'en entreprise (qui demeure son secteur de prédilection), au même titre que d'autres frameworks et outils en Java, XML ou même en PHP. En effet, les développeurs se soucient de plus en plus de créer des applications propres, fiables et faciles à maintenir. Je ne me mouille pas trop en disant que le concept de séparation des tâches devrait également devenir quelque chose d'incontournable dans les années à venir. Quoi qu'il en soit, une chose est sure : la façon de développer évolue et continuera à évoluer dans le bon sens.

Les contributions de ce mémoire sont multiples. D'une part, il pourrait s'agir d'un premier manuel de référence en français sur Cocoon, qui en manque cruellement. D'autre part, ce document est un hymne à la création d'applications Web d'un très haut niveau d'accessibilité et de visibilité tout en respectant les normes établies et sans débourser le moindre centime.

D'un point de vue personnel, ce travail m'a donné l'occasion de me spécialiser dans le langage XML ainsi que dans les dialectes et les outils qui en découlent. Ce domaine qui m'intéressait il y a un an me passionne aujourd'hui au plus haut point. D'un autre côté, ce mémoire en entreprise m'a permis de combiner l'aspect scientifique et l'aspect commercial de l'informatique, qu'on présente souvent comme incompatibles. Le site d'eTransparence en est le contre exemple parfait ! Pour terminer, je dirais que la société eTransparence m'a offert la possibilité d'expérimenter le travail "sur le terrain", ce qui est un acquis et un atout non négligeable pour mon entrée très prochaine dans le monde du travail.

Bibliographie

1. Livres

[1] *Art of Java Web Development (Struts, Tapestry, Commons, Velocity, JUnit, Axis, Cocoon, Interbeans, Webwork)*. Neal Ford. Manning (http://www.manning.com/ford). 1-9323-9406-0.

[2] *Beginning PHP, Apache, MySQL Web Development*. Michael K. Glass, Yann Le Scouarnec, Elizabeth Naramore, Gary Mailer, Jeremy Stolz, and Jason Gerner. Wrox (http://www.wrox.com). 0-7645-5744-0.

[3] *Cocoon : Building XML Applications*. Matthew Langham and Carsten Ziegeler. Sams Publishing (http://www.samspublishing.com). 0-7357-1235-2.

[4] *Cocoon 2 Programming : Web Publishing with XML and Java*. Bill Brogden, Conrad D'Cruz, and Mark Gaither. Sybex (http://www.sybex.com). 0-7821-4131-5.

[5] *Content Syndication with RSS*. Ben Hammersley. O'Reilly (http://www.oreilly.com). 0-596-00383-8.

[6] *DocBook : The Definitive Guide*. Norman Walsh and Leonard Muellner. O'Reilly (http://www.oreilly.com). 1-5659-2580-7.

[7] *Professional XML Development with Apache Tools*. Théodore W. Leung. Wrox (http://www.wrox.com). 0-7645-4355-5.

[8] *Référence PHP 5*. Nicolas Borde, Arnaud Mahrin, and Marc Thévenet. Micro Application (http://www.microapp.com). 2-7429-3217-8.

[9] *XML in a Nutshell, 2nd Edition*. Elliotte Rusty Harold and W. Scott Means. O'Reilly (http://www.oreilly.com). 0-596-00292-0.

2. Articles

[10] *Building XML Portals with Cocoon*. Julien Mercay and Gilbert Bouzeid. JavaWorld.com (http://www.javaworld.com/javaworld/jw-02-2002/jw-0201-strutsxslt.html). 1er février 2002.

[11] *Boost Struts with XSLT and XML - An introduction to Model 2X*. Matthew Langham and Carsten Ziegeler. XML.com (http://www.xml.com/pub/a/2002/07/24/xmlportal.html). 24 juillet 2002.

[12] *Combining Stylesheets with Include and Import*. Bob DuCharme. XML.com (http://www.xml.com/pub/a/2000/11/01/xslt/). 1er novembre 2000.

[13] *Evaluation des technologies XML sur base d'une étude de cas*. Dimitri Fontaine. Université de Mons-Hainaut (ftp://ftp.umh.ac.be/pub/ftp_ssi/teaching/memoires/Fontaine.zip). Juin 2003.

[14] *Initiation au positionnement CSS*. Laurent Denis. Openweb.eu.org (http://www.openweb.eu.org/articles/initiation_flux/). 21 mars 2003.

[15] *Introducing Cocoon 2.0*. Stefano Mazzocchi. XML.com (http://www.xml.com/pub/a/2002/02/13/cocoon2.html). 13 février 2002.

[16] *From Wiki to XML, through SGML*. Rick Jelliffe. XML.com (http://www.xml.com/pub/a/2004/03/03/sgmlwiki.html). 3 mars 2004.

[17] *Getting Started With Cocoon 2*. Steve Punte. XML.com (http://www.xml.com/pub/a/2002/07/10/cocoon2.html). 10 juillet 2002.

[18] *La page d'accueil de debian-fr passée aux standards*. Laurent Jouanneau. Ljouanneau.com (http://ljouanneau.com/standards/convertion/debian-fr/). Mai 2003.

[19] *La syntaxe wiki sur le chemin de la standardisation*. Laurent Jouanneau. Ljouanneau.com (http://ljouanneau.com/blog/2003/09/29/158-LaSyntaxeWikiSurLeCheminDeLaStandardisation). 29 septembre 2003.

[20] *Les fils RSS pour rassembler l'info*. Serge Courrier. SVM (http://svm.vnunet.fr). Mars 2005.

[21] *Les problèmes de la mise en page par tableaux*. Mathieu Pillard. Openweb.eu.org (http://www.openweb.eu.org/articles/problemes_tableaux/). 24 octobre 2003.

[22] *Mais où va l'open source?*. Patrick Bénichou. ZDNet (http://www.zdnet.fr/actualites/informatique/0,39040745,39219270,00.htm). 21 avril 2005.

[23] *MyGoogle: A Simple Cocoon Application that Uses Google's SOAP API*. Pankaj Kumar. Pankaj-k.net (http://www.pankaj-k.net/sdwest2002/readme.html). 23 avril 2002.

[24] *New-Window Links in a Standards-Compliant World*. Kevin Yank. Sitepoint (http://www.sitepoint.com/article/standards-compliant-world). 4 mars 2003.

[25] *Online Magazines with Apache Cocoon*. Steve Punte. XML.com (http://www.xml.com/pub/a/2003/04/23/cocoon-magazine.html). 23 avril 2003.

[26] *PHP en entreprise - White Paper*. Association Française des Utilisateurs de PHP (http://www.afup.org). 13 janvier 2005.

[27] *Portal Syndication with Web Services and Cocoon*. Ivelin Ivanov. The Apache Software Foundation (http://cocoon.apache.org/2.1/userdocs/generators/wsproxy-generator.html).

[28] *Référencement gratuit sur Google : 10 conseils*. Web Rank Info (http://www.webrankinfo.com/referencement/conseils/conseils.php).

[29] *RSS - Dates essentielles*. Fréderic Laurent. Opikanoba (http://www.opikanoba.org/xml/040315/). 15 mars 2004.

[30] *Standards Applied: Using Apache Cocoon and Forrest*. Steven Noels. Idealliance (http://www.idealliance.org/papers/dx_xmle03/papers/02-01-05/02-01-05.html). 02 janvier 2005.

[31] *The Object-Oriented Evolution of PHP*. Zeev Suraski. DevX (http://www.devx.com/webdev/Article/10007). 15 novembre 2002.

[32] *What is RSS ?*. Mark Pilgrim. XML.com (http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html). 18 décembre 2002.

[33] *XML + XSLT*. Luc Vanlancker. Publication HTML (http://www.ccim.be/ccim328/xml/xml_doc.pdf). 15 août 2001.

[34] *XSLT Notes*. Jef Wijsen. Université de Mons-Hainaut. 12 mai 2004.

[35] *XSLTs for RSS and Atom Feeds*. Rich Manalang. Manalang.com (http://manalang.com/archives/2004/06/17/xslts-for-rss-and-atom-feeds/). 17 juin 2004.

3. Sites Internet

- Adobe SVG Zone : http://www.adobe.com/svg.
- Ambivalence : http://ambivalence.sourceforge.net.
- Anyware Technologies : http://www.anyware-tech.com.
- Apache Cocoon : http://cocoon.apache.org.
- Apache Cocoon | Authentication Framework : http://cocoon.apache.org/2.1/developing/webapps/authentication.html.
- Apache Cocoon | Components | Actions : http://cocoon.apache.org/2.1/userdocs/actions/actions.html.
- Apache Cocoon | Components | Actions | Sendmail : http://cocoon.apache.org/2.1/userdocs/actions/sendmail-action.html.
- Apache Cocoon | Components | Generators : http://cocoon.apache.org/2.1/userdocs/generators/generators.html.
- Apache Cocoon | Components | Generators | Directory : http://cocoon.apache.org/2.1/userdocs/generators/directory-generator.html.
- Apache Cocoon | Components | Generators | File (default) : http://cocoon.apache.org/2.1/userdocs/generators/file-generator.html.
- Apache Cocoon | Components | Generators | Mp3directory : http://cocoon.apache.org/2.1/userdocs/generators/mp3directory-generator.html.
- Apache Cocoon | Components | Generators | Xpathdirectory : http://cocoon.apache.org/2.1/userdocs/generators/xpathdirectory-generator.html.
- Apache Cocoon | Components | Matchers : http://cocoon.apache.org/2.1/userdocs/matchers/matchers.html.
- Apache Cocoon | Components | Matchers | WildCard URI (default) : http://cocoon.apache.org/2.1/userdocs/matchers/wildcarduri-matcher.html.
- Apache Cocoon | Components | Readers : http://cocoon.apache.org/2.1/userdocs/readers/readers.html.
- Apache Cocoon | Components | Readers | Resource : http://cocoon.apache.org/2.1/userdocs/readers/resource-reader.html.
- Apache Cocoon | Components | Selectors : http://cocoon.apache.org/2.1/userdocs/selectors/selectors.html.
- Apache Cocoon | Components | Selectors | Browser (default) : http://cocoon.apache.org/2.1/userdocs/selectors/browser-selector.html.
- Apache Cocoon | Components | Selectors | Parameter : http://cocoon.apache.org/2.1/userdocs/selectors/parameter-selector.html.
- Apache Cocoon | Components | Serializers : http://cocoon.apache.org/2.1/userdocs/serializers/serializers.html.
- Apache Cocoon | Components | Serializers | HTML (default) : http://cocoon.apache.org/2.1/userdocs/serializers/html-serializer.html.
- Apache Cocoon | Components | Serializers | PDF : http://cocoon.apache.org/2.1/userdocs/serializers/pdf-serializer.html.
- Apache Cocoon | Components | Serializers | SVG : http://cocoon.apache.org/2.1/userdocs/serializers/svg-serializer.html.
- Apache Cocoon | Components | Serializers | XHTML : http://cocoon.apache.org/2.1/userdocs/serializers/xhtml-serializer.html.
- Apache Cocoon | Components | Serializers | XML : http://cocoon.apache.org/2.1/userdocs/serializers/xml-serializer.html.
- Apache Cocoon | Components | Transformers : http://cocoon.apache.org/2.1/userdocs/transformers/transformers.html.
- Apache Cocoon | Components | Transformers | I18N : http://cocoon.apache.org/2.1/userdocs/transformers/i18n-transformer.html.
- Apache Cocoon | Components | Transformers | Lexical : http://cocoon.apache.org/2.1/userdocs/transformers/lexer-transformer.html.
- Apache Cocoon | Components | Transformers | Parser : http://cocoon.apache.org/2.1/userdocs/transformers/parser-transformer.html.
- Apache Cocoon | Components | Transformers | XSLT (default) : http://cocoon.apache.org/2.1/userdocs/transformers/xslt-transformer.html.
- Apache Cocoon | Concepts | Redirection : http://cocoon.apache.org/2.1/userdocs/concepts/redirection.html.
- Apache Cocoon | Concepts | Mounting sitemaps : http://cocoon.apache.org/2.1/userdocs/concepts/sitemap.html#Mounting+sitemaps.
- Apache Cocoon | Hosting : http://cocoon.apache.org/link/hosting.html.
- Apache Cocoon | XSP Logicsheet : http://cocoon.apache.org/2.1/userdocs/xsp/logicsheet.html.
- Apache Cocoon Wiki : http://wiki.apache.org/cocoon.
- Apache Forrest : http://forrest.apache.org.
- Apache Jakarta Tomcat : http://jakarta.apache.org/tomcat.
- Apache Jakarta Regexp : http://jakarta.apache.org/regexp.
- Apache Lenya : http://lenya.apache.org.
- Apache Struts : http://struts.apache.org.

- Apache Xindice : http://xml.apache.org/xindice.
- Batik SVG Toolkit : http://xml.apache.org/batik.
- binarycloud : http://www.binarycloud.com.
- Bloglines : http://www.bloglines.com.
- Chaperon : http://chaperon.sourceforge.net.
- Cocoon Center : http://www.cocooncenter.org.
- Cocoondev : http://www.cocoondev.org.
- CSS Zen Garden : http://www.csszengarden.com.
- DocBook : http://www.docbook.org.
- DocBook XSL : http://docbook.sourceforge.net/projects/xsl.
- EasyPHP : http://www.easyphp.org.
- eXist : http://exist.sourceforge.net.
- Feed Validator : http://www.feedvalidator.org.
- Feeddemon : http://www.bradsoft.com/feeddemon.
- Feedreader : http://www.feedreader.com.
- FOP : http://xml.apache.org/fop.
- GNU General Public License : http://www.gnu.org/copyleft/gpl.html.
- Google : http://www.google.be.
- Google APIs : http://www.google.com/apis.
- Google Fight : http://www.googlefight.com.
- Horde Application Framework : http://www.horde.org/horde.
- HSQLDB : http://hsqldb.sourceforge.net.
- InterJinn : http://www.interjinn.com.
- Java Technology : http://java.sun.com.
- Jetty : http://jetty.mortbay.org/jetty.
- jfor : http://www.jfor.org.
- K-Meleon : http://kmeleon.sourceforge.net.
- Krysalis : http://www.kompletecms.com/products/krysalis.
- LatoServer : http://www.latoserver.it.
- Lepido : http://www.eclipse.org/proposals/eclipse-lepido/index.html.
- Magpie RSS : http://magpierss.sourceforge.net.
- Mojavi : http://www.mojavi.org.
- Mozilla Firefox : http://www.mozilla.org/products/firefox.
- Netcraft : http://news.netcraft.com.
- OASIS : http://www.oasis-open.org.
- OpenWeb : http://www.openweb.eu.org.
- Opera : http://www.opera.com.
- PEAR : http://pear.php.net.
- PHP : http://www.php.net.
- PHP Designer : http://www.mpsoftware.dk/phpdesigner.php.
- PHP Editors : http://www.php-editors.com.
- php.MVC : http://www.phpmvc.net.
- Planet Cocoon : http://www.planetcocoon.com.
- Phrame : http://phrame.sourceforge.net.
- Pollo : http://pollo.sourceforge.net.
- Quanta Plus : http://quanta.kdewebdev.org.
- RSS 0.91 Specification : http://my.netscape.com/publish/formats/rss-spec-0.91.html.
- RSS 1.0 Specification : http://web.resource.org/rss/1.0/spec.
- RSS 2.0 Specification : http://blogs.law.harvard.edu/tech/rss.
- RSS4you : http://www.rss4you.com.
- SAX Project : http://www.saxproject.org.
- Seagull : http://seagull.phpkitchen.com.
- SELF HTML : http://fr.selfhtml.org.
- Sharpreader : http://www.sharpreader.net.
- Skype : http://www.skype.com.
- Smart : http://developer.berlios.de/projects/smart.
- Smarty : http://smarty.php.net.
- Spread Cocoon : http://www.spreadcocoon.com.
- Stefano Mazzocchi's Home Page : http://www.betaversion.org/~stefano.
- Studs : http://mojavelinux.com/projects/studs.
- Syndic8 : http://www.syndic8.com.
- TagTag : http://www.tagtag.com.

- The Apache XML Project : http://xml.apache.org.
- The Apache Jakarta Project : http://jakarta.apache.org.
- The Apache Software Foundation : http://www.apache.org.
- W3C : http://www.w3.org.
- W3C | CSS : http://www.w3.org/Style/CSS.
- W3C | CSS Validator : http://jigsaw.w3.org/css-validator.
- W3C | DOM : http://www.w3.org/DOM.
- W3C | HTML : http://www.w3.org/MarkUp.
- W3C | HTML Validator : http://validator.w3.org.
- W3C | MathML : http://www.w3.org/Math.
- W3C | PNG : http://www.w3.org/Graphics/PNG.
- W3C | RDF : http://www.w3.org/RDF.
- W3C | SOAP/XMLP : http://www.w3.org/2000/xp/Group.
- W3C | SVG : http://www.w3.org/Graphics/SVG.
- W3C | XML : http://www.w3.org/XML.
- W3C | XML Query : http://www.w3.org/XML/Query.
- W3C | XML Schema : http://www.w3.org/XML/Schema.
- W3C | XML Path Language : http://www.w3.org/TR/xpath.
- W3C | XSL and XSLT : http://www.w3.org/Style/XSL.
- W3Schools : http://www.w3schools.com.
- Wact : http://wact.sourceforge.net.
- WAMP : http://www.wampserver.com.
- WAP Forum : http://www.wapforum.org.
- Web Rank Info : http://www.webrankinfo.com.
- Wiki2xhtml : http://www.neokraft.net/sottises/wiki2xhtml.
- Wikipedia, l'Encyclopédie Libre : http://fr.wikipedia.org.
- Winwap : http://www.winwap.org.
- Xalan-Java : http://xml.apache.org/xalan-j.
- Xaraya : http://www.xaraya.com.
- Xerces Java Parser : http://xml.apache.org/xerces-j.
- XML.com : http://www.xml.com.
- XMLfr.org : http://www.xmlfr.org.

ANNEXES

Annexe i

DTD du RSS 2.0

```
<!-- RSS 2.0 DTD -->

<!ELEMENT rss (channel)>
<!ATTLIST rss version CDATA #FIXED "2.0">

<!-- A channel can apparently either have one or more items,
     or just a title, link, and description of its own -->

<!ELEMENT channel ((item+)|
                   (title,link,description,(language|copyright|
                    managingEditor|webMaster|pubDate|lastBuildDate|
                    category|generator|docs|cloud|ttl|image|
                    textInput|skipHours|skipDays)*))>

<!ELEMENT item ((title|description)+,link?,

(author|category|comments|enclosure|guid|pubDate|source)*)>

<!ELEMENT author (#PCDATA)>
<!ELEMENT category (#PCDATA)>
<!ATTLIST category domain CDATA #IMPLIED>
<!ELEMENT cloud (#PCDATA)>
<!ATTLIST cloud domain CDATA #IMPLIED
                port CDATA #IMPLIED
                path CDATA #IMPLIED
                registerProcedure CDATA #IMPLIED
                protocol CDATA #IMPLIED>
<!ELEMENT comments (#PCDATA)>
<!ELEMENT copyright (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT docs (#PCDATA)>
<!ELEMENT enclosure (#PCDATA)>
<!ATTLIST enclosure url CDATA #REQUIRED
                    length CDATA #REQUIRED
                    type CDATA #REQUIRED>
<!ELEMENT generator (#PCDATA)>
<!ELEMENT guid (#PCDATA)>
<!ATTLIST guid isPermaLink (true|false) "true">
<!ELEMENT height (#PCDATA)>
<!ELEMENT image (url,title,link,(width|height|description)*)>
<!ELEMENT language (#PCDATA)>
<!ELEMENT lastBuildDate (#PCDATA)>
<!ELEMENT link (#PCDATA)>
<!ELEMENT managingEditor (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT pubDate (#PCDATA)>
<!ELEMENT skipDays (#PCDATA)>
<!ELEMENT skipHours (#PCDATA)>
<!ELEMENT source (#PCDATA)>
<!ATTLIST source url CDATA #REQUIRED>
<!ELEMENT textInput (title,description,name,link)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT ttl (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT webMaster (#PCDATA)>
<!ELEMENT width (#PCDATA)>
```

Annexe ii

Installation de Cocoon

Installer Cocoon sous Windows XP

Testé avec J2SE 1.4.2, Tomcat 5.0.27, Cocoon 2.1.5.1

1. **Installer Java 2 Standard Edition**
   a. Télécharger la dernière version de J2SE ici : http://java.sun.com/j2se
   b. Installer en suivant les instructions.
   c. Modifier le "Path" dans les variables systèmes (Poste de travail > Propriétés > Avancé > Variables d'environnment).
   Ajouter en tête de la liste le chemin des fichiers binaires de Java.
   Exemple : c:\j2sdk1.4.2_06\bin; (Ne pas oublier le point-virgule pour séparer)
   d. Ajouter une variable système dont le nom est "JAVA_HOME" et la valeur est le chemin de Java.
   Exemple : c:\j2sdk1.4.2_06
   Tester que la variable a bien été prise en compte en tapant : `cd %JAVA_HOME%` dans l'invite de commandes.
2. **Installer Apache Jakarta Tomcat**
   a. Télécharger la version binaire de Tomcat 5 ici :
   http://jakarta.apache.org/site/binindex.cgi
   b. Décompresser l'archive dans un répertoire au choix.
   Exemple : c:\webdev\
   c. Ajouter une variable système dont le nom est "CATALINA_HOME" et la valeur est le chemin de Tomcat.
   Exemple : c:\webdev\jakarta-tomcat-5.0.27
   Tester que la variable a bien été prise en compte en tapant : `cd %CATALINA_HOME%` dans l'invite de commandes.
   d. Créer deux fichiers batch :
   Le premier se nomme "start_tomcat.bat" et son contenu est :
   `@%CATALINA_HOME%\bin\startup`
   Le second se nomme "shutdown_tomcat.bat" et son contenu est :
   `@%CATALINA_HOME%\bin\shutdown`
   e. Pour démarrer Tomcat, double-cliquer sur le fichier "start_tomcat.bat".
   Une fois démarré, vérifier que l'installation a été réussie en tapant l'adresse suivante dans le navigateur : http://localhost:8080
   Vous devriez voir apparaitre la page d'accueil de Tomcat.
   f. Pour arrêter Tomcat, double-cliquer sur le fichier "shutdown_tomcat.bat".
3. **Installer Apache Cocoon**
   a. Télécharger le code source de Cocoon 2.1 ici : http://cocoon.apache.org/mirror.cgi
   b. Décompresser l'archive dans le répertoire de votre choix.
   Exemple : c:\webdev\
   c. Ajouter une variable système dont le nom est "COCOON_HOME" et la valeur est le chemin de Cocoon.
   Exemple : c:\webdev\cocoon-2.1.5.1
   d. Compiler Cocoon :
   Ouvrir l'invite de commandes et se placer dans le répertoire de Cocoon en tapant : `cd %COCOON_HOME%`
   Taper : `build war`
   e. Copier le fichier "cocoon.war" se trouvant dans le répertoire "%COCOON_HOME%\build\cocoon-2.1.5.1\" vers le répertoire "%CATALINA_HOME%\webapps\".
   f. Démarrer Tomcat et vérifier que l'installation de Cocoon a été réussie en tapant l'adresse suivante dans le navigateur : http://localhost:8080/cocoon

Annexe iii

Feuilles de style XSLT pour le RSS

```
<?xml version="1.0" ?>
<!--  RSS 0.9x Stylesheet    -->
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
  - <xsl:template match="/">
    - <html>
      - <head>
          <link rel="stylesheet" type="text/css" href="/cocoon/
        etransparence/css/rss.css" />
          <title>eTransparence News</title>
      </head>
      - <body id="page">
          <xsl:apply-templates select="/rss/channel" />
      </body>
    </html>
  </xsl:template>
  - <xsl:template match="/rss/channel">
    - <div class="syndication-content-area">
      - <div class="syndication-title">
          <xsl:value-of select="title" />
      </div>
      - <div class="syndication-image">
          <xsl:apply-templates select="image" />
      </div>
      - <div class="syndication-description">
          <xsl:value-of select="description" />
      </div>
      - <ul class="syndication-list">
          <xsl:apply-templates select="item" />
      </ul>
    </div>
  </xsl:template>
  - <xsl:template match="/rss/channel/item">
    - <li class="syndication-list-item">
      - <a href="{link}" class="ternal">
          <xsl:value-of select="title" />
      </a>
      - <div class="syndication-list-item-description">
          <xsl:value-of select="description" />
      </div>
    </li>
  </xsl:template>
  - <xsl:template match="image">
    - <a>
      - <xsl:attribute name="href">
          <xsl:value-of select="link" />
      </xsl:attribute>
      - <img class="noborder">
        - <xsl:attribute name="src">
```

```
                <xsl:value-of select="url" />
            </xsl:attribute>
            - <xsl:attribute name="alt">
                <xsl:value-of select="title" />
            </xsl:attribute>
        </img>
    </a>
  </xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" ?>
<!--  RSS 1.0 Stylesheet    -->
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:dc="http://purl.
org/dc/elements/1.1/" xmlns:foo="http://purl.org/rss/1.0/">
  - <xsl:template match="/">
    - <html>
      - <head>
          <link rel="stylesheet" type="text/css" href="/cocoon/
        etransparence/css/rss.css" />
          <title>eTransparence News</title>
        </head>
      - <body id="page">
          <xsl:apply-templates select="/rdf:RDF/foo:channel" />
        </body>
      </html>
  </xsl:template>
  - <xsl:template match="/rdf:RDF/foo:channel">
    - <div class="syndication-content-area">
      - <div class="syndication-title">
          <xsl:value-of select="foo:title" />
        </div>
      - <div class="syndication-image">
          <xsl:apply-templates select="foo:image" />
        </div>
      - <div class="syndication-description">
          <xsl:value-of select="foo:description" />
        </div>
      - <ul class="syndication-list">
          <xsl:apply-templates select="/rdf:RDF/foo:item" />
        </ul>
      </div>
  </xsl:template>
  - <xsl:template match="/rdf:RDF/foo:item">
    - <li class="syndication-list-item">
      - <a href="{foo:link}" title="{foo:description}" class="ternal"
      rel="external">
          <xsl:value-of select="foo:title" />
        </a>
      - <span class="syndication-list-item-date">
        (
          <xsl:value-of select="dc:date" />
        )
        </span>
      - <div class="syndication-list-item-description">
          <xsl:value-of select="foo:description" />
        </div>
      </li>
  </xsl:template>
  - <xsl:template match="foo:image">
```

```
      - <a>
        - <xsl:attribute name="href">
            <xsl:value-of select="foo:link" />
          </xsl:attribute>
        - <img class="noborder">
          - <xsl:attribute name="src">
              <xsl:value-of select="foo:url" />
            </xsl:attribute>
          - <xsl:attribute name="alt">
              <xsl:value-of select="foo:title" />
            </xsl:attribute>
        </img>
      </a>
    </xsl:template>
  </xsl:stylesheet>
```

```
<?xml version="1.0" ?>
<!--  RSS 2.0 Stylesheet    -->
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
  - <xsl:template match="/">
    - <html>
      - <head>
          <link rel="stylesheet" type="text/css" href="/cocoon/
        etransparence/css/rss.css" />
          <title>eTransparence News</title>
      </head>
      - <body id="page">
          <xsl:apply-templates select="/rss/channel" />
      </body>
    </html>
  </xsl:template>
  - <xsl:template match="/rss/channel">
    - <div class="syndication-content-area">
      - <div class="syndication-title">
          <xsl:value-of select="title" />
      </div>
      - <div class="syndication-image">
          <xsl:apply-templates select="image" />
      </div>
      - <div class="syndication-description">
          <xsl:value-of select="description" />
      </div>
      - <ul class="syndication-list">
          <xsl:apply-templates select="item" />
      </ul>
    </div>
  </xsl:template>
  - <xsl:template match="/rss/channel/item">
    - <li class="syndication-list-item">
      - <a href="{link}" title="{description}" class="ternal" rel="external">
          <xsl:value-of select="title" />
      </a>
      - <span class="syndication-list-item-date">
        (
          <xsl:value-of select="pubDate" />
        )
      </span>
      - <div class="syndication-list-item-description">
          <xsl:value-of select="description" />
      </div>
    </li>
  </xsl:template>
  - <xsl:template match="image">
    - <a>
```

```
        - <xsl:attribute name="href">
            <xsl:value-of select="link" />
          </xsl:attribute>
          - <img class="noborder">
            - <xsl:attribute name="src">
                <xsl:value-of select="url" />
              </xsl:attribute>
            - <xsl:attribute name="alt">
                <xsl:value-of select="title" />
              </xsl:attribute>
          </img>
      </a>
    </xsl:template>
</xsl:stylesheet>
```

Annexe iv

Exemples de fichiers .xlex et .xgrm

```
<?xml version="1.0" ?>
- <lexicon>
  - <lexeme symbol="id">
    - <cclass minOccurs="1" maxOccurs="*">
        <cinterval min="A" max="Z" />
        <cinterval min="a" max="z" />
      </cclass>
    </lexeme>
  - <lexeme symbol="number">
    - <cclass minOccurs="1" maxOccurs="*">
        <cinterval min="0" max="9" />
      </cclass>
    </lexeme>
  - <lexeme symbol="mult">
      <cstring content="*" />
    </lexeme>
  - <lexeme symbol="div">
      <cstring content="/" />
    </lexeme>
  - <lexeme symbol="plus">
      <cstring content="+" />
    </lexeme>
  - <lexeme symbol="minus">
      <cstring content="-" />
    </lexeme>
  - <lexeme symbol="pleft">
      <cstring content="(" />
    </lexeme>
  - <lexeme symbol="pright">
      <cstring content=")" />
    </lexeme>
  - <lexeme>
    - <cclass minOccurs="1" maxOccurs="*">
        <cset content="" />
      </cclass>
    </lexeme>
</lexicon>
```

```
<?xml version="1.0" ?>
- <grammar>
  - <priority>
      <terminal symbol="mult" />
      <terminal symbol="div" />
      <terminal symbol="plus" />
      <terminal symbol="minus" />
  </priority>
    <associativity symbol="mult" type="right" />
    <associativity symbol="div" type="right" />
    <associativity symbol="plus" type="right" />
    <associativity symbol="minus" type="right" />
  - <production symbol="exp">
      <nonterminal symbol="exp" />
      <terminal symbol="plus" />
      <nonterminal symbol="exp" />
  </production>
  - <production symbol="exp">
      <nonterminal symbol="exp" />
      <terminal symbol="minus" />
      <nonterminal symbol="exp" />
  </production>
  - <production symbol="exp">
      <nonterminal symbol="exp" />
      <terminal symbol="mult" />
      <nonterminal symbol="exp" />
  </production>
  - <production symbol="exp">
      <nonterminal symbol="exp" />
      <terminal symbol="div" />
      <nonterminal symbol="exp" />
  </production>
  - <production symbol="exp">
      <terminal symbol="pleft" />
      <nonterminal symbol="exp" />
      <terminal symbol="pright" />
  </production>
  - <production symbol="exp">
      <terminal symbol="id" />
  </production>
  - <production symbol="exp">
      <terminal symbol="number" />
  </production>
    <start symbol="exp" />
</grammar>
```

Annexe v

Présentation de DocBook

DocBook, DocBook

DocBook est un modèle de documentation technique actuellement maintenu par un comité technique du consortium *OASIS Open*. Son utilisation s'avère particulièrement intéressante dès lors qu'il s'agit de créer des documents techniques ou des articles.

Cette recommandation est issue d'utilisateurs de l'industrie de l'informatique et de l'électronique. Du coup, il existe beaucoup de structures permettant de coder des informations liées à ce domaine : qu'il s'agisse des messages de spécification d'interface utilisateur ou d'objets et autres classes liées à une documentation issue d'une modélisation objet.

Les modèles issus de DocBook sont de conception extrêmement moderne et permettent d'ajouter ses propres éléments dans le modèle de base (la recommandation parle alors de "*wrapper*"). De même, si des objets ne sont pas utiles, ils peuvent être aisément supprimés. Beaucoup d'industriels se sont donc emparés de ce modèle et l'ont adapté à leurs propres besoins.

Note. Certains trouvent que DocBook est trop difficile à paramétrer et, par conséquent, inutilisable. Les concepteurs travaillent alors aujourd'hui sur une vision simplifiée de DocBook appelée "Simplified DocBook".

Toujours d'un point de vue conception, le modèle est *disponible* et *extrêmement bien documenté* sur Internet. Mais aussi, sont disponibles des *feuilles de styles*, utilisant XSLT et XSL, destinées à faciliter la diffusion de ce type d'informations sur le *Web*, sur Wap, et sur *papier*.

Quelle sera la prochaine version de DocBook ? C'est un débat qui agite en ce moment la communauté *OASIS Open*, entre les "pro" de la spécification actuelle et ceux qui veulent, soit en faire quelque chose de plus modulaire quant à sa mise en œuvre, soit quelque chose de plus indépendant de l'industrie de l'informatique et de l'électronique, soit, enfin, quelque chose se rapprochant de DITA et de l'organisation de la documentation en modules autonomes d'information.

Objectifs

DocBook est le modèle de documentation technique actuellement maintenu par un comité technique du consortium *OASIS Open*.

DocBook a maintenant plus de dix ans. À l'origine créé par l'éditeur O'Reilly, son objectif était de simplifier l'échange de documentations UNIX. Puis, avec la création du groupe Davenport, son objectif s'est élargi, pour devenir le format d'écriture et d'échange de documentations techniques issues du monde des industries de l'informatique et de l'électronique.

La vision des documentations proposées par DocBook est proche du livre. Un document DocBook est soit un ensemble de livres, soit un livre seul, voire un seul article. Dans ce livre, tout peut être défini : qu'il s'agisse des Métadonnées de gestion comme les copyrights ou noms d'auteurs ; que ce soit aussi l'organisation des contenus par sections, avec des annexes ; qu'il s'agisse, enfin, des informations d'accès que représentent les tables des matières et autres index.

Dans les contenus eux-mêmes, le modèle différencie le niveau des composants (*component*) de celui des objets apparaissant directement dans le texte. Pour les deux types d'éléments, il existe toujours des éléments de présentation et des éléments plus sémantiques, liés à l'industrie visée.

Qui utilise DocBook ? Tout d'abord la communauté UNIX (plus particulièrement aujourd'hui la communauté LINUX), pour ses manuels. Ensuite, ce sont aussi les industriels et surtout les secteurs de l'Après-Vente. Enfin, avec l'avènement de "simplified DocBook", une vision simplifiée de DocBook, beaucoup s'emparent du modèle pour l'adapter à leurs besoins, dès lors qu'ils ont besoin d'un modèle "documentaire" de base pour commencer à travailler.

Ce qui fait le succès de DocBook est, certes, le fait d'exister et d'être soutenu par le consortium *OASIS Open*. C'est aussi le fait que beaucoup d'outils implémentent ce modèle, tant pour créer les documents (*Frame SGML*, *Epic*, *XMLmind*, *XML Spy*, *XMetaL*, etc.) que pour utiliser les données issues de ce modèle. Dans le cadre de l'utilisation, on citera en tout premier lieu les feuilles de styles disponibles en "open source" proposées par Norman Walsh, l'éditeur du standard au sein d'*OASIS Open* ; on citera aussi les différentes implémentations disponibles dans les produits libres et/ou commerciaux (*Frame SGML*, *Epic*, *XPP*, etc.)

Principes

Les différents éléments du modèle

Les éléments de DocBook peuvent être classés selon la hiérarchie suivante :

- ensembles ;
- livres ;
- divisions ;
- articles ;
- sections.

À la différence des standards de documentation technique basés sur la réutilisation de fragments pour créer une publication (voir par exemple S1000D), DocBook contient toute l'information nécessaire à une documentation technique. Ainsi, et par exemple, un livre comprendra des informations d'identification formelle, tous les composants d'index permettant la navigation, une préface, des chapitres, des

annexes, des glossaires et une bibliographie.

Bien sûr, tous ces éléments sont optionnels et c'est à l'utilisateur de choisir ce qu'il veut mettre en place. L'utilisateur peut être restreint dans ces choix et il faut alors paramétrer le modèle.

Aux niveaux les plus bas on trouve les notions de :

- **Blocs**
  Ce sont des éléments de type paragraphe, avec un interligne avant et après dès lors qu'ils sont composés.
  On trouve parmi eux différentes catégories de listes, d'avertissements, de synopsis, de tableaux, de figures, d'exemples, de média comme la vidéo, etc.
  On trouve aussi des éléments plus sémantiques comme ce qui permet de décrire, en ingénierie logicielle, des options et des paramètres d'une commande. Pour décrire des programmes, on trouvera ce qui permet d'identifier des fonctions, leurs arguments et valeurs de retour, etc ; il y aura de la même façon les notion de message, voire de procédure de maintenance.
  La provision pour ce type d'élément est extrêmement vaste, d'où la nécessité, souvent, d'adapter DocBook à ses propres besoins, afin de ne pas être obligé de manipuler tous les composants DocBook.
- **Éléments en ligne**
  Ce sont les éléments que l'on retrouve directement mixés avec les phrases, les mots et les caractères.
  À nouveau, on y trouve des éléments de présentation comme la mise en relief, les citations, les abréviations et autres acronymes. On trouve aussi tout ce qui permet de gérer les renvois (à des sections, des tableaux, des médias, etc.).
  À nouveau aussi, on trouve un certain nombre d'informations liées à l'industrie visée comme des objets d'identification d'interface utilisateur, d'objets d'un langage de programmation, d'un système d'exploitation, etc.

Avec cette provision d'éléments, il est possible de réaliser ses propres documentations techniques, dès lors que l'on appartient à l'industrie visée. Qu'en est-il alors des utilisateurs d'une industrie différente ? Qu'en est-il aussi des utilisateurs de cette industrie qui ont des besoins beaucoup plus limités ?

C'est là qu'intervient la force de DocBook. En effet, le modèle est conçu pour qu'il soit possible soit d'adjoindre de nouvelles structures, soit d'en supprimer. Les deux opérations se font par redéfinition et la méthode fonctionne tant lorsque l'on utilise les DTD que lorsque l'on utilise les Schema.

Modèle simplifié

Si DocBook contient autant d'objets structurels, c'est qu'ils sont nécessaires pour atteindre tous les objectifs documentaires de l'industrie visée. On a par ailleurs dit qu'il était possible de simplifier l'utilisation de DocBook par redéfinition du modèle.

Pour prouver la faisabilité de la chose et pour proposer aux implémenteurs une solution finalisée simple, les membres du comité DocBook proposent un modèle dit simplifié.

Dans ce modèle, beaucoup d'éléments ont été supprimés pour ne conserver que le strict minimum, toujours dans le cadre d'un utilisation en documentation technique.

Ce travail a été réalisé dans les "règles de l'art", par redéfinition du modèle complet.

Pour réaliser sa propre application, il est alors possible de choisir parmi les possibilités suivantes :

- utilisation de tout DocBook ;
- utilisation du modèle simplifié ;
- redéfinition de DocBook ;
- redéfinition du seul modèle simplifié.

Après avoir travaillé en redéfinition sur les deux modèles, la question est de savoir s'il existe une préférence sur celui sur lequel s'appuyer. Le choix est uniquement lié à la complexité du modèle souhaité et à ce que l'on souhaite utiliser dans la provision d'éléments DocBook. Il faut donc, dans un premier temps choisir le cadre dans lequel on s'insère, puis le redéfinir.

Informations connexes

On notera l'implémentation de DocBook réalisée dans le cadre de *Tirème* pour mettre en place la technologie *SchemaDoc* : un atelier de documentation de modèles issus d'XML et représentés sous forme de Schema, voire de DTD.

Spécification(s) associé(e)s


DocBook XML
Recommandation, version 4.4, du 27-01-2005
Document sur
http://www.oasis-open.org/docbook/specs/cd-docbook-docbook-4.4.html


DocBook SGML
Recommandation, version 4.2, du 17-07-2002
Document sur http://www.oasis-open.org/docbook/sgml/4.2/index.1.shtml


"Simplified" DocBook
Recommandation, version 1.0, du 19-11-2002
Document sur http://www.oasis-open.org/specs/cs-docbook-simple-1.0.html