Tri des éléments

L'instruction **xsl:sort** s'utilise exclusivement en complément des instructions **xsl:apply-template** ou **xsl:for-each** présentées précédemment. Elle permet de trier les différents fragments de code générés d'une manière répétitive par l'une des deux instructions citées précédemment. Elle possède plusieurs attributs dont le plus important est **select** qui permet de définir la clé du tri à effectuer. La valeur représentant la clé sera une expression X-Path qui identifiera l'élément ou l'attribut en fonction duquel sera effectué le tri. Un autre attribut optionnel nommé **order** permet de définir l'ordre du tri. La valeur de cet attribut peut être **ascending** (ordre ascendant) ou **descending** (ordre descendant). Par défaut, la valeur de **order** est **ascending**.

Cette instruction **xsl:sort** peut se présenter sous deux formes différentes :

Utilisation de **xsl:sort** en complément de l'instruction **xsl:for-each**.

```
<xsl:for-each select="nomAttribut" >
   <xsl:sort select="CLE" />
   <!—fragment de code répété -->
</xsl:for-each>
```

Utilisation de **xsl:sort** en complément de l'instruction **xsl:apply-template**.

```
<xsl:template match="MOTIF">
...<xsl:apply-templates >
   <xsl:sort select="CLE" />
<xsl:apply-templates />...
</xsl:template>
```

Pour illustrer l'utilisation de l'instruction **xsl:sort**, nous l'avons employée en complément d'une instruction **xsl:for-each** afin de trier la liste des propriétaires en fonction de leur étage. Le fichier XML source sera le même fichier **source.xml** déjà utilisé dans les exemples précédents.

test6.xsl

```
<?xml version="1.0" encoding="iso-8859-1"?><!-- DWXMLSource="test1.xml" -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="iso-8859-1"/>
<xsl:template match="/">
<html>
<head>
<title>Test 6</title>
</head>
<body>
```

```
<xsl:apply-templates />
</body>
</html>
</xsl:template>
<xsl:template match="immeuble">
   <p>Liste des propriétaires : </p>
   <xsl:for-each select="proprietaire" >
         <xsl:sort select="etage" />
           <p>Nom :<xsl:value-of select="@nom"/></p>
           <p>Etage :  <xsl:value-of select="etage"/></p>
           <p>E-mail : <xsl:value-of select="email"/></p>
      </xsl:for-each>
   <p>Copyright 2006 </p>
</xsl:template>
</xsl:stylesheet>
```

Le résultat obtenu avec cette feuille XSLT modifiée est le suivant (à comparer avec le résultat de l'exemple précédent) :

Liste des propriétaires : Nom :
BertautEtage : 1er
E-mail : abertaut@aol.com
Nom :Defrance
Etage : 2eme
E-mail : jmdefrance@aol.com
Copyright 2006

Pour consulter les codes source de ces exemples

Afin de vous permettre de tester vous-même les différents exemples de cette précédente partie concernant le langage XSLT, nous avons rassemblé tous les fichiers utilisés lors de nos démontrations dans un dossier `SITExslt` disponible dans le kit ressource que vous pouvez récupérer sur le site de l'éditeur (www.editions-eyrolles.com).

Le comportement XSLT de Dreamweaver

Dans cette partie, nous allons enfin passer à la pratique en utilisant le comportement serveur XSLT disponible depuis Dreamweaver 8. Nous vous présenterons pas à pas la procédure pour vous servir de ce comportement dans le cadre d'une transformation basique. La dernière partie de ce chapitre étant consacrée à l'application de ce comportement dans une série de transformations de complexité croissante. L'utilisation des comportements serveur XSLT ne nécessite aucune connaissance des langages X-Path et XSLT, bien qu'il soit conseillé d'en avoir. Nous vous invitons cependant à analyser le contenu des pages XSLT créées par Dream-

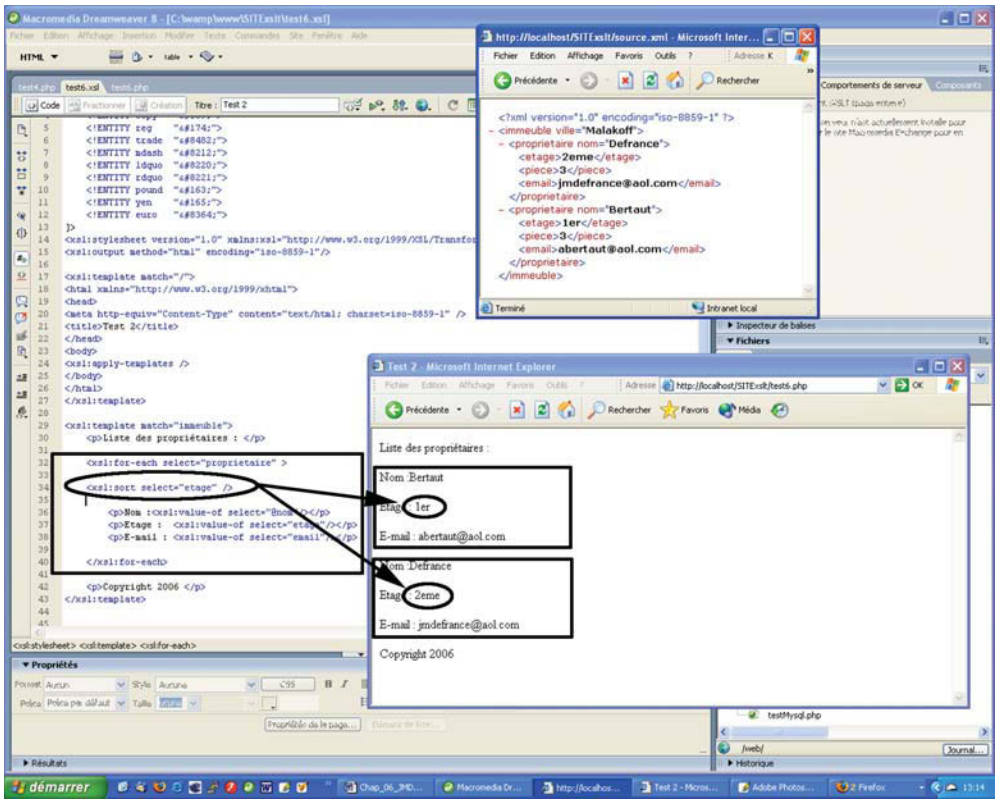


Figure 6-12
Test d'une feuille XSLT utilisant une instruction xsl:sort en complément d'une instruction xsl:for-each.

weaver, et à vous reporter aux ressources sur ces langages au début de ce chapitre, pour bien en comprendre les mécanismes et être capable de les modifier et, pourquoi pas, pour créer vos propres pages de transformation vous-même.

Configuration de l'environnement de développement

Vérification de la présence de l'extension XSL de PHP

Pour pouvoir utiliser le comportement serveur XSLT de Dreamweaver, il faut que l'extension XSL de PHP utilisée par le comportement soit installée. Avant toute chose, il faut donc commencer par vérifier si l'extension est disponible sur votre serveur. Pour cela, un simple fichier `phpinfo` vous renseignera sur cet état (sur Wamp c'est encore plus simple, car les extensions installées sont visibles depuis la rubrique `PHP extensions` du manager). Nous vous rappelons que pour créer un fichier `phpinfo.php`, il suffit d'insérer la fonction `phpinfo()` dans

votre page PHP. Une fois la page `phpinfo.php` affichée, lancer une recherche sur le mot clé « `xsl` » pour localiser rapidement l'information sur l'extension XSL (voir figure 6-13).

Si votre recherche reste infructueuse, l'extension n'est pas disponible. Vous devrez donc l'installer avant de créer vos comportements serveur XSLT.

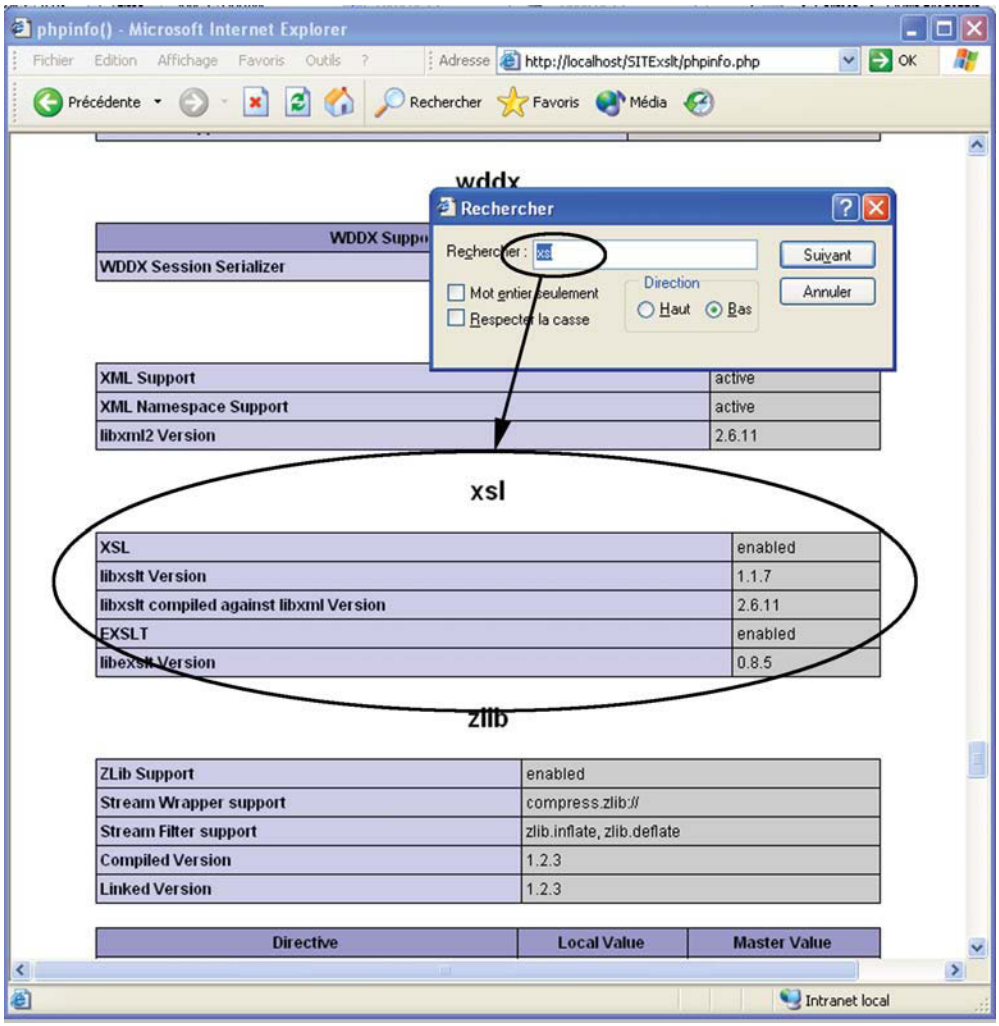


Figure 6-13

Test de présence de l'extension XSL de PHP à l'aide d'une recherche dans le phpinfo.

Installation de l'extension XSL de PHP

Dans le cadre de cet ouvrage, nous allons détailler la procédure (très simple) pour installer l'extension XSL sur Wamp5. À noter que selon votre infrastructure serveur, la procédure sera différente, et nous vous invitons à consulter la documentation ou le support technique de votre serveur pour plus d'informations.

L'installation de l'extension `XSL` sur Wamp5 est très facile. Depuis le manager de Wamp5, sélectionnez `PHP extensions`, puis recherchez l'extension `php_xsl` (vers le bas de la liste). Si le nom de l'extension n'est pas précédé d'une petite flèche, cela signifie que l'extension n'est pas encore installée. Cliquez alors sur le nom de l'extension pour l'activer. Redémarrez ensuite Wamp5 en cliquant sur l'entrée `Restart All Services` depuis le manager. Vous pouvez maintenant effectuer le test de présence de l'extension XSL en affichant le fichier `phpinfo.php` comme indiqué précédemment. À noter que sur Wamp5, un fichier `phpinfo` est déjà disponible depuis la page `localhost` accessible depuis l'entrée du même nom, dans le manager.

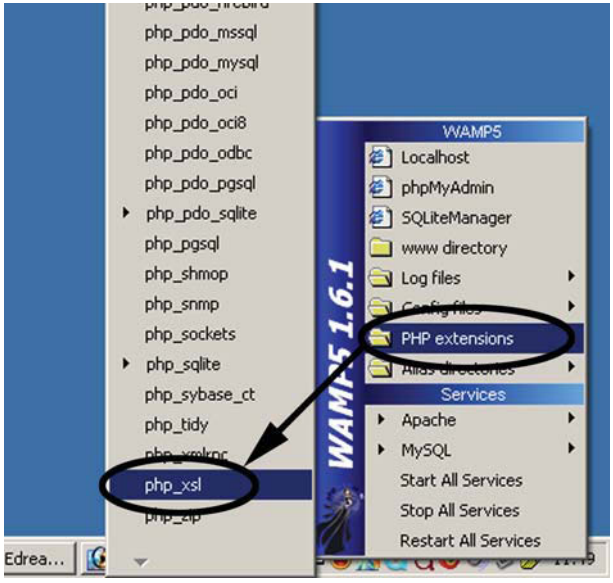


Figure 6-14
Test de présence de l'extension XSL de PHP à l'aide d'une recherche dans le phpinfo.

Présentation du comportement serveur XSLT

Rappels sur les transformations XSLT

Nous avons vu, au début de ce chapitre, que les documents XML ne contiennent pas d'instructions de mise en forme, mais qu'ils permettent de structurer l'information. Nous avons ensuite présenté le langage XSL qui permet d'organiser un document XML, au même titre qu'une feuille de style CSS permet d'ordonner un document HTML. Si on lie un document XML à

une feuille XSL, on peut ainsi afficher son contenu dans un navigateur au format XHTML (cas le plus fréquent) ou autres, selon l'application cliente ciblée (WML, etc.). Lorsque nous avons abordé les spécifications XSL, nous avons présenté trois sous-ensembles du XSL : le XSLT, X-Path et XSL-FO. Si on dispose d'un processeur XSLT (côté serveur ou côté client) et si on associe les deux langages XSLT et X-Path, il est alors possible de transformer le contenu d'un document XML parallèlement à la mise en forme des informations, opérée par le langage XSL.

Côté client ou côté serveur

En pratique, vous pouvez utiliser le comportement XSLT de Dreamweaver pour créer des pages permettant d'effectuer des transformations XSL. Celles-ci peuvent être réalisées côté serveur, à l'aide d'une technologie comme PHP, ou simplement côté client en exploitant le processeur XSLT intégré dans les nouveaux navigateurs Internet.

Ces deux méthodes présentent des avantages et des inconvénients. Les transformations côté serveur pourront être interprétées par tous les navigateurs, puisque le document généré par le serveur sera compatible XHTML, alors que les transformations côté client ne pourront pas être prises en charge par les anciens navigateurs (seuls les navigateurs récents comme Internet Explorer 6, Firefox 1.0.2, etc., disposent d'un processeur XSLT qui permet de les réaliser). La transformation côté client nécessite le téléchargement complet de la source XML, alors que pour celle côté serveur, seules les données sélectionnées par X-Path seront rapatriées dans le fichier XSLT. Ce qui peut être très avantageux pour traiter des sources XML volumineuses. D'autre part, dans le cas d'une transformation côté client, les fichiers XML doivent être obligatoirement stockés sur votre serveur Web (vous devez disposer des droits d'écriture sur le fichier XML), tandis que la transformation côté serveur n'a pas cette restriction et peut modifier dynamiquement des fichiers XML, quel que soit leur emplacement (sur votre serveur ou ailleurs sur le Web). Toutefois la transformation côté serveur nécessite de disposer d'une technologie serveur (comme PHP) et d'une configuration adaptée (posséder l'extension XSL sur le serveur d'application), alors que la transformation côté client nécessite d'avoir un simple serveur Web.

Dans le cadre de cet ouvrage, nous nous intéresserons uniquement aux transformations côté serveur.

Préparation du site Syndic

Pour appliquer la création d'un comportement serveur XSLT à une étude de cas concrète, nous vous proposons de créer un nouveau site nommé Syndic. L'objectif de ce site est de générer, à l'aide d'un comportement XSLT, une fiche d'état d'un immeuble affichant les différentes coordonnées des propriétaires de chaque appartement, classés par bâtiment, et dont la finalité sera de mettre en évidence ceux dont le compte est débiteur. La source XML contenant toutes les informations sur les propriétaires sera reprise du fichier utilisé dans l'introduction au XML présentée au début de ce chapitre (`immeuble.xml`). Dans un premier temps, nous allons nous servir de ce site pour présenter la procédure de création d'un compor-

tement serveur XSLT avec Dreamweaver, mais sachez qu'il sera aussi repris dans la partie suivante consacrée aux applications pratiques XSLT.

1. Depuis Dreamweaver, cliquez sur `Site` dans le menu, puis sélectionnez `Nouveau site`. Renseignez ensuite les informations de la catégorie `Infos locales` et `Serveur d'évaluation` (créer un nouveau dossier `SITEsyndic` dans le répertoire `www` de Wamp5, puis configurez les autres paramètres comme pour le site Score : revoir figures 2-90 et 2-92).
2. Ouvrez un nouveau fichier (`Ctrl + N`) en sélectionnant `Page de base`, puis `XML` dans la fenêtre `Nouveau document`.
3. Entrez ensuite la structure et les données du fichier XML `immeuble.xml` présenté au début de ce chapitre, dans la partie dédiée au XML (voir figure 6-15). Afin d'éviter cette saisie, sachez que vous pouvez aussi récupérer ce fichier XML dans le kit de ressource disponible sur le site de l'éditeur (le fichier `immeuble.xml` se trouve dans le dossier `SITEsyndic` du kit).

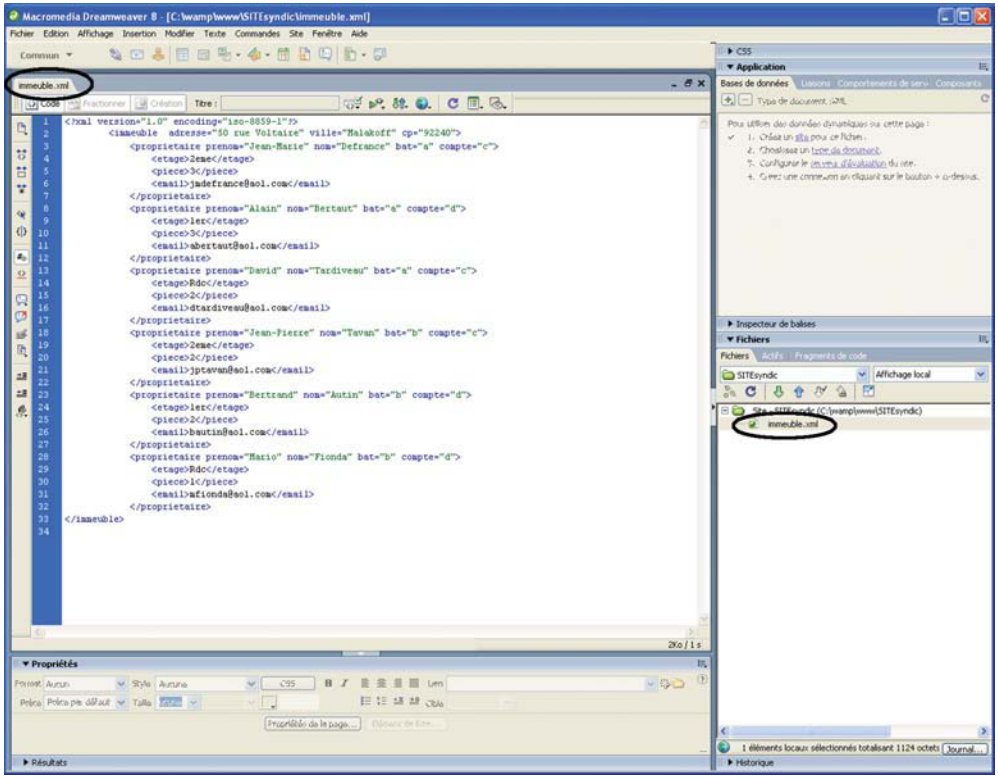


Figure 6-15
Création du fichier immeuble.xml dans Dreamweaver.

Page entière ou fragment XSLT

Dreamweaver permet de créer des pages XSLT générant entièrement une page HTML, ou des fragments XSLT ne produisant qu'une partie seulement de la page HTML.

Une page XSLT entière est semblable à une page HTML classique. Elle contient une balise `<body>` et une balise `<head>`, et permet d'afficher des données HTML et XML dans la même page. Un fragment XSLT est un ensemble de code regroupé dans un fichier séparé, engendrant des données XML formatées qui seront ensuite insérées dans la page HTML finale. À la différence d'une page XSLT, un fragment XSLT est un fichier indépendant qui ne contient pas de balises `<body>` ou `<head>`.

En pratique, l'usage d'une page XSLT entière est souvent mieux adapté à la transformation côté client, alors que les fragments XSLT sont fréquemment utilisés côté serveur, dans le cadre d'insertion ponctuelle d'un extrait de données XML, dans une page dynamique PHP.

Pour illustrer la création de comportements serveur XSLT, nous emploierons donc des fragments XSLT.

Création d'un fragment XSLT

1. Commencez par créer une nouvelle page PHP, et enregistrez-la sous le nom `ficheImmeuble.php`, à la racine du site Syndic.
2. Personnalisez cette page en lui attribuant une couleur de fond et un titre, en utilisant les styles de votre choix.
3. En dessous du titre, ajoutez une ligne d'information destinée par la suite à indiquer l'adresse, la ville et le code postal de l'immeuble concerné. Pour distinguer les informations qui seront issues du fichier XML, nous vous suggérons de les nommer, dans un premier temps, avec les mots repères suivants : `ADRESSE`, `VILLE` et `CP`, et de leur attribuer un style spécifique (voir figure 6-16).
4. Enregistrez votre page et passez en mode `Code`. Placez votre curseur dans le texte que vous venez d'ajouter, et cliquez sur le bouton `Sélectionner une balise parent` afin de sélectionner tout le groupe d'informations contenu entre les balises `<p>` et `</p>` (voir figure 6-17, cela peut bien sûr être différent selon la mise en forme que vous avez réalisée). Ces informations représentent le modèle de votre futur fragment XSLT. Coupez ce groupe d'informations en utilisant le raccourci `Ctrl + X`.
5. Créez un nouveau document (`Ctrl + N`) en sélectionnant `Page de base` puis `XSLT(fragment)` (voir figure 6-18).
6. Dès l'ouverture de la nouvelle page, une boîte de dialogue apparaît, vous demandant de localiser le fichier XML qui sera exploité dans ce fragment XSLT (voir figure 6-19). Deux alternatives sont proposées : Associer un fichier local (sur votre serveur Web) ou Associer un fichier distant (ailleurs sur Internet). Dans notre cas, nous choisirons la pre-

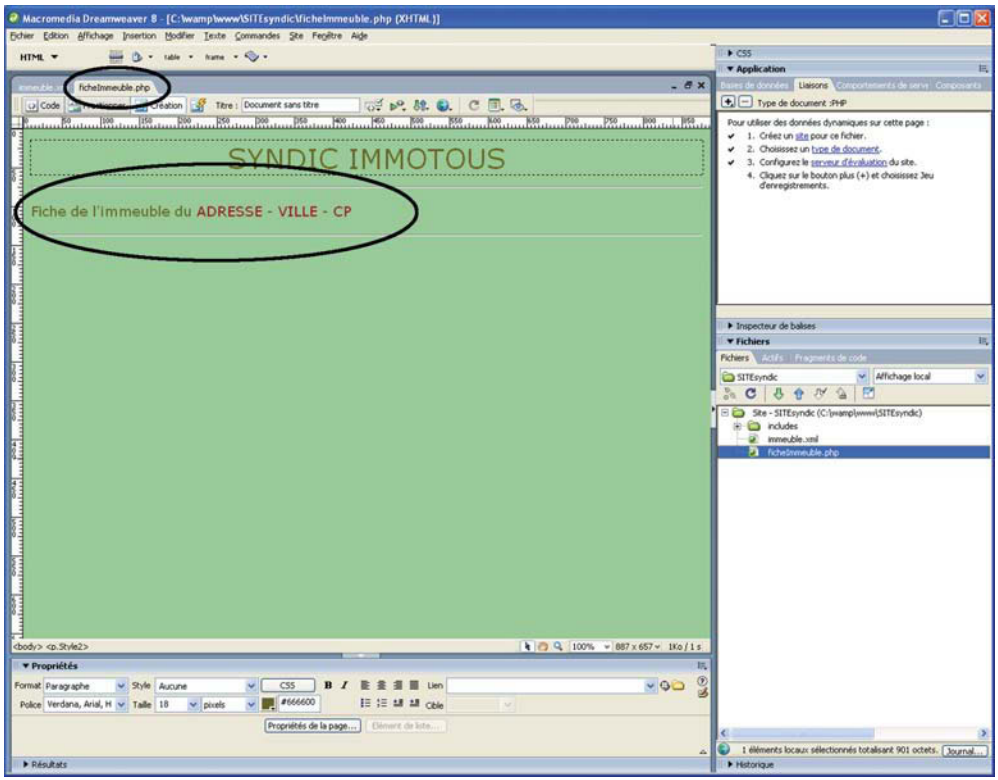


Figure 6-16
Mise en forme de la fiche immeuble (ficheImmeuble.php).

mière et nous cliquerons ensuite sur le bouton **Parcourir** afin de sélectionner le fichier **immeuble.xml** situé à la racine du site Syndic. Une fois le fichier XML configuré, cliquez sur le bouton **OK** pour confirmer votre choix.

7. Enregistrez dès maintenant votre nouveau fichier XSLT sous le nom **fragmentImmeuble.xsl**. Si vous observez le panneau **Liaisons** (voir figure 6-20), vous remarquerez qu'il contient le schéma de la structure du fichier XML **immeuble.xml**. On peut voir également que les différents éléments sont désignés par leur nom précédé du symbole « **<>** ». Les attributs sont aussi représentés par leur nom, précédé cette fois du symbole « **@** ». De même, le nom de l'élément répétitif **proprietaire** suit le symbole « **<>+** ». Passez en mode **Code** et positionnez ensuite votre curseur après la balise **<xsl:template match="/">**. Cliquez sur la touche **Entrée** de votre clavier pour passer à la ligne, et collez à cet emplacement le fragment de code que vous aviez coupé précédemment (**Ctrl + V**, voir figure 6-20).

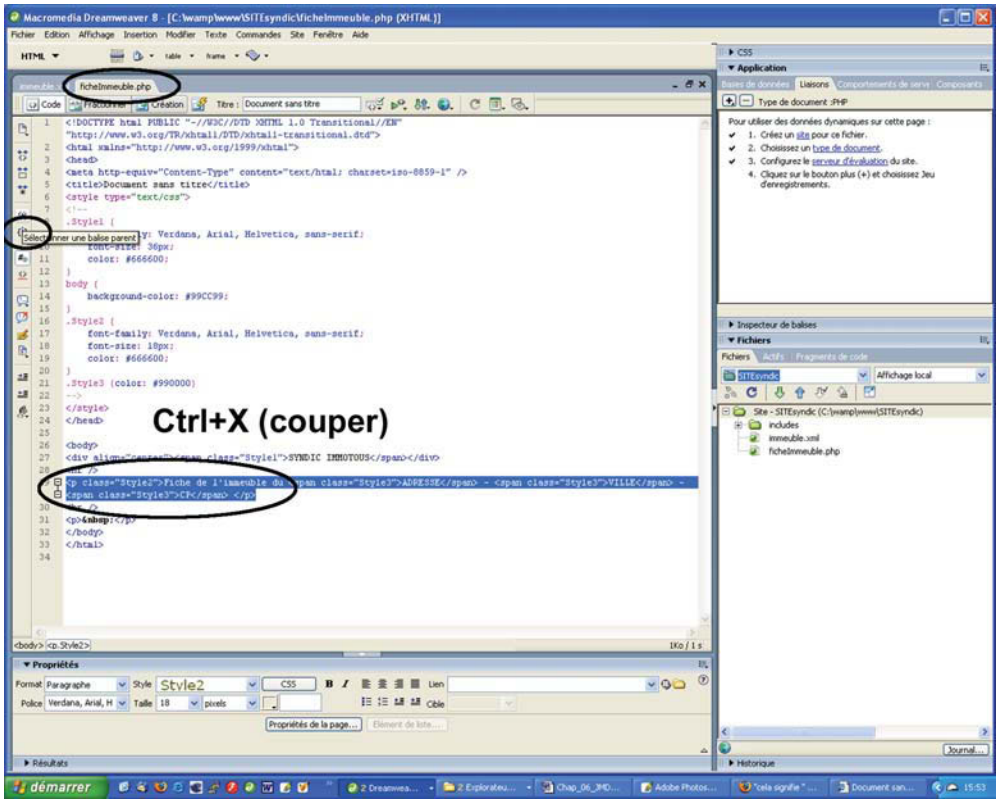
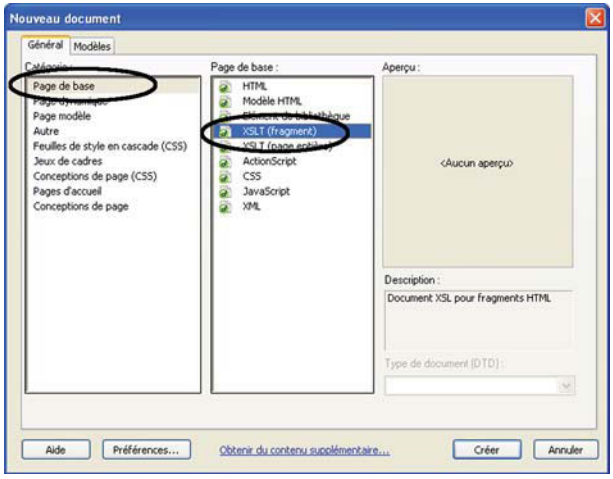


Figure 6-17
Sélectionnez puis coupez le groupe d'informations du futur fragment XSLT.

Figure 6-18
Création du fragment XSLT.



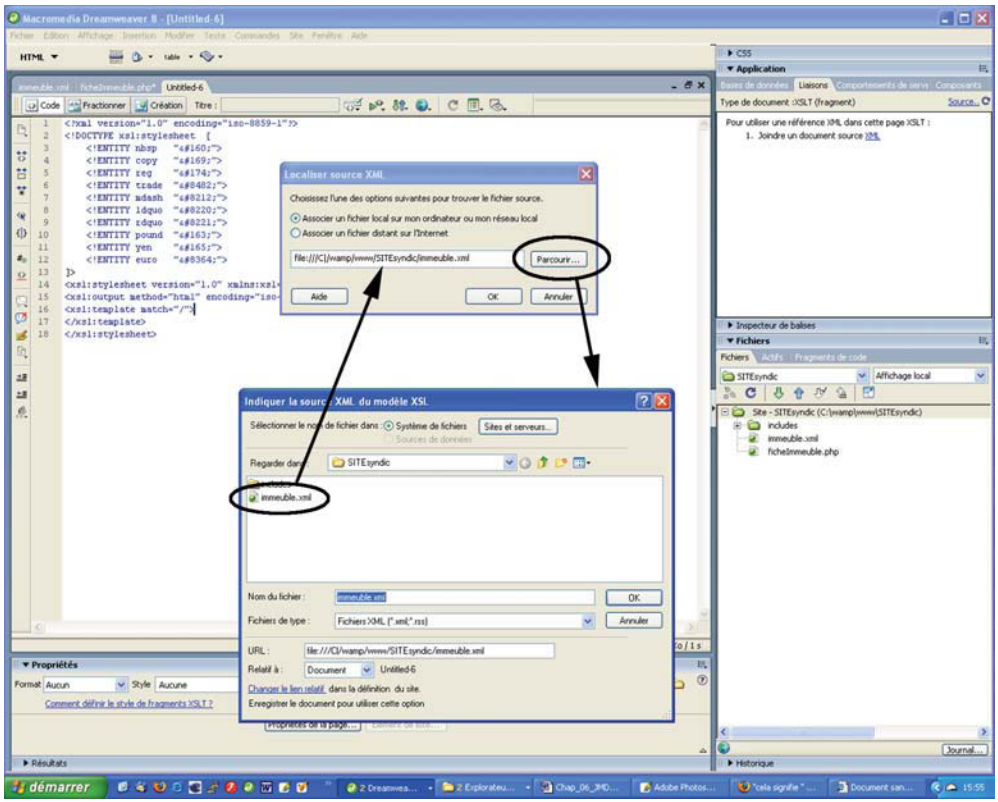


Figure 6-19
Localisation de la ressource XML associée au fragment XSLT.

8. Passez maintenant en mode **Création**. Sélectionnez le mot repère **ADRESSE**, puis faites un double-clic sur l'attribut correspondant dans le panneau **Liaisons**. Dreamweaver génère alors automatiquement le chemin de localisation X-Path relatif à cette information et remplace ainsi le mot repère **ADRESSE**. Procédez de la même manière avec les deux autres informations concernant l'immeuble : **VILLE** et **CP** (voir figure 6-21).

9. Enregistrez le fichier **fragmentImmeuble.xsl** et revenez maintenant au fichier **ficheImmeuble.php**. Passez en mode **Code** et placez votre curseur au même emplacement que celui du fragment de code coupé précédemment (voir figure 6-22 repère 1). Cliquez ensuite sur le bouton **+** du panneau **Application/Comportements de serveurs** et sélectionnez l'option **Transformation XSL** (voir figure 6-22 repères 2 et 3). Dans la boîte de dialogue **Transformation XSL**, cliquez sur le bouton **Parcourir** situé à droite du champ **Fichier XSLT**. Choisissez ensuite le fichier **fragmentImmeuble.xsl** et validez votre choix. Le fichier **immeuble.xml** doit alors automatiquement être renseigné, et il ne vous reste

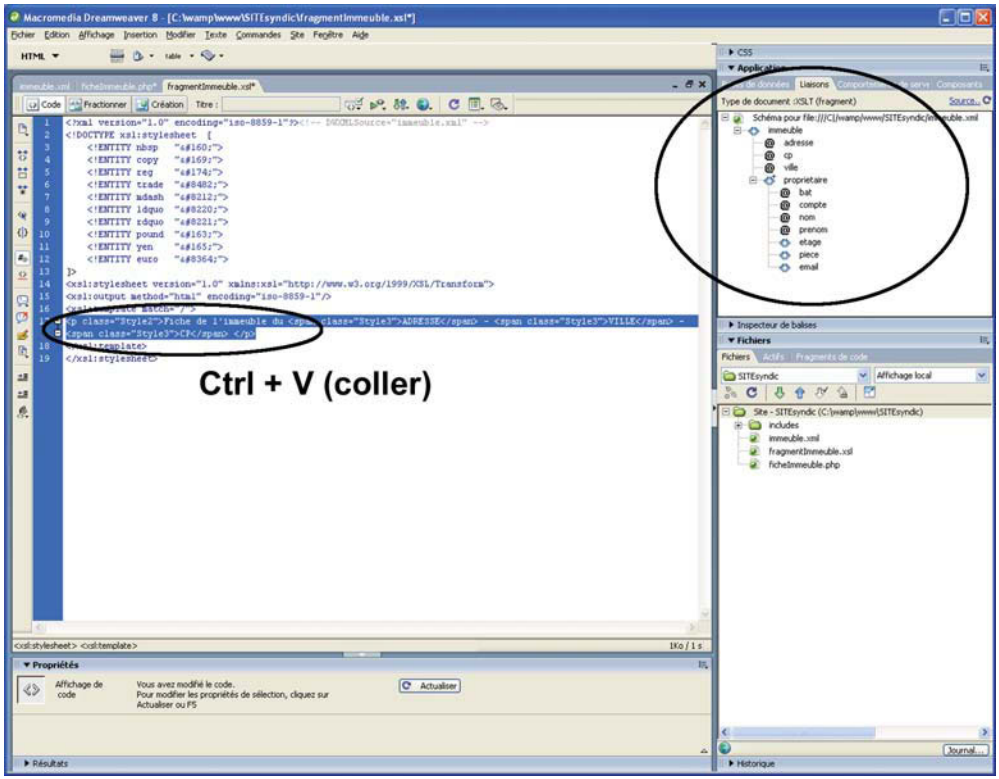


Figure 6-20
Insertion du fragment de code modèle dans le fichier XSLT.

plus qu'à valider cette dernière boîte de dialogue en cliquant sur le bouton `OK` (voir figure 6-22 repères 4 et 5).

10. Une fois le comportement serveur XSLT créé, on peut observer dans la page `ficheImmeuble.php` qu'un script PHP a été ajouté à l'endroit du point d'insertion (voir figure 6-23). D'autre part, dans le panneau `Fichier`, un nouveau dossier contenant le fichier de classe utilisé par ce script a été aussi automatiquement généré. Évidemment, si vous transférez par la suite votre site sur un serveur distant, il ne faudra pas oublier d'y déplacer aussi ce dossier.

11. Si nous passons maintenant en mode `Création`, les chemins de localisation X-Path sont alors visibles dans la page (voir figure 6-24). Faites un premier test en cliquant sur le bouton `Live Data` (voir figure 6-25). Enregistrez ensuite toutes vos pages et passez dans le Web local pour tester complètement le système (voir figure 6-26).

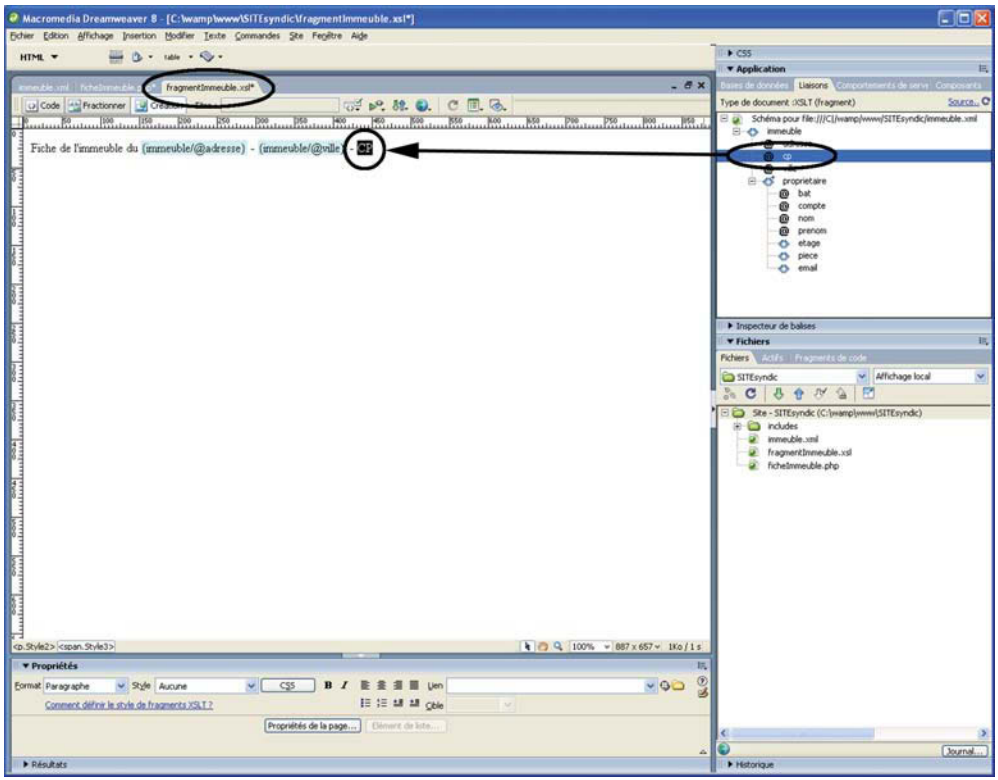


Figure 6-21

Configuration des chemins de localisation X-Path correspondant à chaque attribut de l'adresse de l'immeuble.

Applications pratiques XSLT

Dans cette partie, nous allons reprendre le développement, avec Dreamweaver, du site Syndic commencé dans la partie précédente. Cela va nous permettre de construire progressivement une feuille XSLT plus élaborée qui affichera la liste des propriétaires (avec toutes leurs coordonnées), intégrera un lien mailto dynamique dans les coordonnées, mettra en évidence les comptes débiteurs par l'ajout d'un point rouge devant les noms des propriétaires concernés, et enfin scindera en deux parties la liste (bâtiment A ou B de l'immeuble) selon la situation des propriétaires.

Création d'une liste des propriétaires

Pour commencer, nous allons créer une liste des propriétaires et de leurs coordonnées en dessous des informations de l'immeuble.

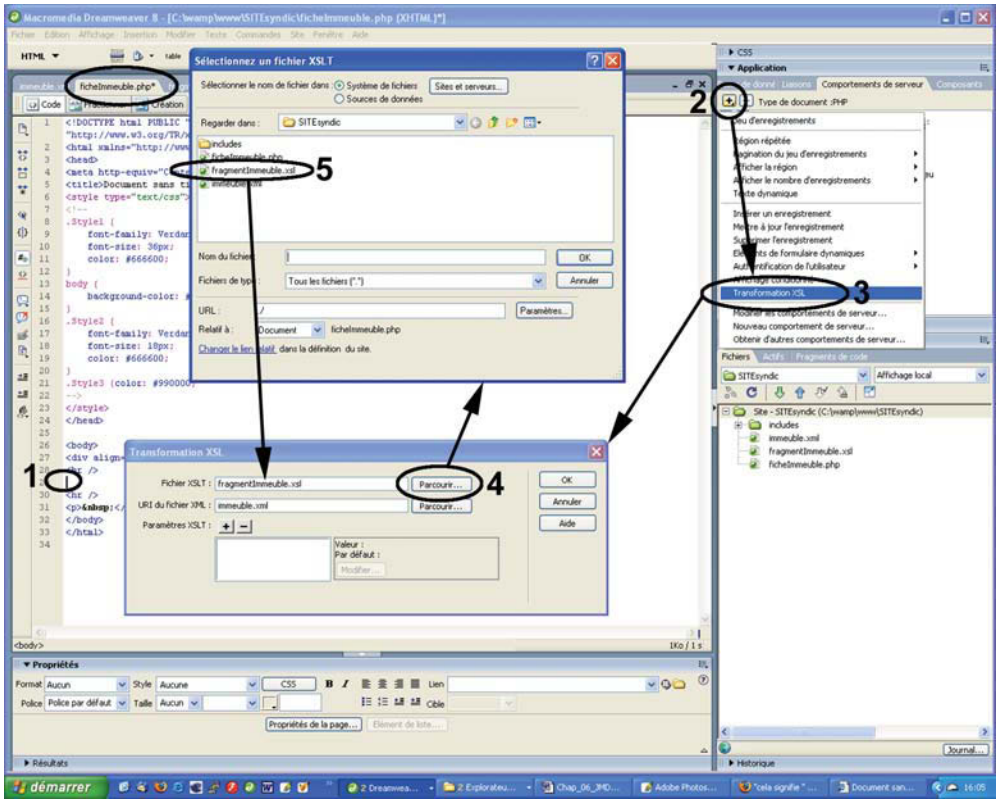


Figure 6-22

Création et configuration du comportement serveur XSLT dans la page ficheImmeuble.php.

La liste des propriétaires venant s'ajouter en dessous de la ligne d'information qui indique l'adresse de l'immeuble (ligne d'information déjà réalisée dans la partie précédente qui présentait la procédure pour créer votre premier comportement XSLT), nous pourrions repartir de la feuille XSL qui avait été créée à cette occasion pour continuer le développement. Cependant, il est préférable de définir la mise en forme complète (disposition des éléments, style, etc.) dans une maquette statique avant de configurer le fragment XSLT dans un fichier à part. En effet, par défaut, vous ne disposerez plus d'accès direct aux styles de la page lors de la conception du fragment XSLT, et l'ajout ou la modification d'un style ne sera alors plus possible (dans notre exemple les styles sont internes à la page, mais dans le cas de styles externes, sachez qu'il est possible de définir des feuilles de style « conception » pour contourner ce problème).

Aussi, nous vous proposons de créer une nouvelle maquette statique depuis laquelle nous extrairons un modèle avec sa mise en forme, afin d'élaborer un nouveau fragment XSLT ne