Traitement de FICHES XML

dans une application construite sur SPIP

François HAMONNO
fhamonno@club-internet.fr

Table des matières

1. Objectifs

Le package décrit ici s'adresse à ceux qui souhaitent utiliser **SPIP** (http://www.spip.net) pour manipuler des objets (fiches techniques par exemple) dont les attributs (ou champs) ne correspondent pas à ceux gérés par ce système de publication : titre, sous-titre, chapeau, texte, …

L'objectif est alors de pouvoir conserver SPIP comme structure d'accueil, en associant un article ou une rubrique à chacun de ces objets.

Ceci est possible en utilisant un codage XML de l'objet (ou plutôt de ses attributs), cette représentation de l'objet étant stockée dans le corps d'un article ou d'une rubrique SPIP qui lui sera associé.

Le couple SPIP+XML est particulièrement performant puisqu'il allie d'une part une gestion "globale" de l'objet en tant qu'article ou rubrique SPIP : hiérarchie, mots-clés, indexation, forum, …, et d'autre part une gestion "applicative" de l'objet en tant qu'ensemble de champs : formulaire de saisie, affichage et traitements personnalisés.

> NB: Pour faciliter la compréhension de la suite de ce document, au lieu de parler "d'objets ayant des attributs ou champs", nous parlerons de "Fiches descriptives", ou plus simplement de "Fiches XML".

2. Utilisation des « Fiches XML »

2.1. Intégration à SPIP : le filtre « processXML »

Les fiches descriptives seront codées en XML dans le champ « Texte » d'un article ou d'une rubrique SPIP, en exploitant toutes les fonctionnalités offertes par SPIP pour classer, accéder et rechercher ces articles ou rubriques.

De plus, à chaque type de fiche, sera associé un modèle qui décrira à la fois l'affichage et la saisie de chacun de ses champs.

Dans la pratique, toutes les balises SPIP pouvant avoir un contenu XML pourront être traitées par le filtre "processXML".

Le traitement de mise en page normalement effectué par SPIP sera inhibé en faisant suivre le nom de la balise par une astérisque (cf. documentation Spip «Mise en page : Manuel de référence» : Court-circuiter le traitement par Spip).

NB : Dans le cas où la balise ne contiendrait pas de code XML, la mise en page standard de SPIP (filtre "propre") sera effectuée par le filtre **"processXML"**.

Par exemple : Affichage du texte d'un article ou d'une rubrique :

```
[(#TEXTE*|processXML)]
```

Afin d'identifier le traitement à effectuer sur le contenu XML d'une fiche, ce dernier commencera par la balise XML « **<DocumentXML>** » décrite plus loin :

<DocumentXML TypeDocXML="*Type_de_fiche*" Handler="*parser*">

Par exemple :
Première ligne de la fiche descriptive d'un utilisateur :

```
<DocumentXML TypeDocXML="FicheUser" Handler="FicheXML">
```

Première ligne de la fiche descriptive d'un devis :

```
<DocumentXML TypeDocXML="DevisPiscine" Handler="FicheXML">
```

NB: Si les données traitées par le filtre "processXML" ne commencent pas par la balise "<DocumentXML>", les données sont retournées à SPIP inchangées, après application du filtre Spip «propre».

2.2. Utilisation du filtre « processXML » par l'application

Le code PHP «applicatif» pourra appeler la fonction "processXML()" pour afficher un document XML contenu dans une variable PHP (comme lors de l'utilisation du filtre par Spip)

Par exemple :

```
$dataHTML = processXML($dataXML);
echo $dataHTML ;
```

2.3. Les paramètres d'environnement du filtre « processXML »

Plusieurs variables globales sont utilisées pour contrôler le traitement d'une « Fiche XML » :

- $xml_TypeFicheXML — Le type de la fiche (au choix de l'application)
- $xml_ObjetSpip — L'objet Spip qui contient ou contiendra la fiche (='article' ou ='rubrique')
- $xml_ParamsObjetSpip — Paramètres pour la création d'un nouvel objet Spip contenant la fiche saisie.
  Syntaxe des formulaires html :
  - Si objet 'rubrique' alors = 'id_secteur=xxx&id_rubrique=0'
  - Si objet 'article' alors = 'id_secteur=xxx&id_rubrique=xxx&id_article=0'

Ces variables sont normalement renseignées par le filtre « **processXML** ».

Exécution du filtre par Spip

Lorsque Spip exécute le filtre « processXML », ce dernier appelle la procédure « **setParamsXml()** » qui renseigne ces variables globales en fonction du contexte en cours.

Les variables $xml_ObjetSpip et $xml_ParamsObjetSpip seront renseignées en exploitant le contexte Spip : nom du squelette en cours d'exécution, id de l'article ou de la rubrique en cours de traitement et paramètres associés…

Le « Type de Fiche XML » sera le mot-clé de l'article ou de la rubrique en cours de traitement, appartenant au **groupe de mots-clés « TypeFicheXML »**.

Le « Modèle de Fiche XML » à traiter sera donné par l'article Spip ayant le **mot-clé « ModeleFicheXML »** et ayant son **SURTITRE='Modele< *Type de Fiche XML*>'** (par ex. 'ModeleFicheClient' ou 'ModeleDevisPiscine').

Rappel : L'article ou la rubrique contenant une « fiche XML » commence toujours par la balise XML
<DocumentXML TypeDocXML="*Type_de_fiche XML*" …>

Renseignement du lien vers la création d'une nouvelle fiche

La procédure « **setParamsXml($typeFicheXML, $id_article_modele)** » doit être appelée pour obtenir l'url du lien vers le formulaire de création d'une nouvelle fiche.
Les paramètres (optionnels) sont:

- $typeFicheXML — Type de « Fiche XML ».
  Si vide, sera renseigné avec la valeur du mot-clé « TypeFicheXML » associé à l'objet Spip en cours donné par $xml_ObjetSpip
- $id_article_modele — Id de l'article Spip contenant le modèle à traiter.
  Si vide, sera l'id de l'article de mot-clé « TypeArticle='ModeleFicheXML' » et de « SURTITRE='Modele<type de fiche XML>' »
- $xml_ObjetSpip — Objet Spip qui contiendra la future fiche.
  Si cette variable globale est vide, elle sera renseignée d'après le squelette en cours (cf. $PHP_SELF).

- $xml_ParamsObjetSpip Paramètres à utiliser pour visualiser la future fiche.
  Si cette variable globale est vide, elle sera renseignée d'après la « Fiche XML » en cours si elle existe.

L'appel à « **setParamsXml()** » renseignera en retour la variable globale **$lienNewFicheXML** avec l'url recherché.

Exemple de code affichant les liens de création de « Devis » dans le squelette « rubrique.html »

```
… … … … … … … … … …
< !—Rechercher la rubrique à traiter →
<BOUCLE_principale(RUBRIQUES){id_rubrique}{doublons}>
< ?php
/// Mémoriser le n° de la rubrique à traiter 'container' et son secteur
/// NB : On suppose qu'elle contient des devis… et on veut afficher les liens vers de nouveaux devis.
    $id_rub  = #ID_RUBRIQUE;
    $id_sect = #ID_SECTEUR;
    $fiches = array();
    $i = 0;
//-----------------------------------------------------
/// Rechercher les modèles de devis (type de fiche « DevisXXXX »)
///   Conventions :
///    => Les modèles sont des articles de mot-clé « ModeleFicheXML »
///    => Le type de modèle est dans le surtitre, et il est de la forme « ModeleDevisXXX »
///    => Le texte à afficher pour le lien est dans le titre
///    => Le code XML à traiter est dans le texte
//-----------------------------------------------------
?>
    <BOUCLE_ModelesFiches(ARTICLES){titre_mot=ModeleFicheXML}>
< ?php
      $modele = '#SURTITRE';
      if ( strpos("_$modele", '_ModeleDevis')==1  ) {                  // $modele commence par 'ModeleDevis'
          $typeFiche = substr($modele, strpos($modele, 'Modele' ) ; // $typeFiche='DevisXXXX'
          $fiches[$i]['texte'] = '[(#TITRE|texte_script)]';           // Conserver le « titre » du modèle

/// Renseigner les variables d'environnement pour « setparamsXML() »
          $xml_ObjetSpip          = 'article';                    // Les devis seront dans des articles
          $xml_ParamsObjetSpip = "id_rubrique=$id_rub&id_secteur=$id_sect&id_article=0";
          setParamsXML($typeFiche, #ID_ARTICLE);           // Renseigner le lien $lienNewFicheXML

/// Renseigner les liens pour la création de fiches
          $fiches[$i]['lien']     = $lienNewFicheXML;
          $i++;
      } // Fin if ( strpos("_$modele", '_ModeleDevis')==1  )
?>
    </BOUCLE_ModelesFiches>
… … … … … … … … … …
<?php
    foreach ($fiches as $fiche) {                                    // Balayer les liens mémorisés
/// Afficher les liens
?>     <A HREF="<?=$fiche['lien'];?>" > <?=$fiche['texte']?> </A> <BR />
<?php
    }
?>
… … … … … … … … … …
```

2.4. L'importation d'une « Fiche XML » dans Spip

Des modèles d'importation, respectant la syntaxe (xml aussi) des exportations de la base réalisées par Spip, sont utilisés pour importer les objets Spip (rubrique, article, mots-clés, …) associés à une « Fiche XML ».

L'importation d'une « Fiche XML » dans Spip, ainsi que la sauvegarde de ses modifications, sont réalisées par la procédure « **importObjectIntoSpip($objetAppli, $attributs, $pathToImportDir)** » appelée avec les paramètres suivants :

- **$objetAppli** Objet "applicatif" à créer ou mettre à jour
  => Les modèles d'importation seront trouvés
  - via la variable globale "${import_$objetAppli}"
  - ou dans le fichier "$pathToImportDir/import_${objetAppli}_Std.xml"
  NB : La variable globale "${import_$objetAppli}" donnera la liste des fichiers "import_xxx.xml" du répertoire "$pathToImportDir" qui devront être traités pour réaliser l'importation de l'objet applicatif.

- **$attributs** Tableau associatif contenant la valeur de chaque « champ » de la « Fiche XML » :
  $attributs[<champ>] = <valeur champ>
  Les champs '%<champ>%' présents dans les modèles d'importation seront remplacés par $attributs[<champ>] avant de donner ce modèle à traiter à la procédure Spip d'importation (cf. **import_objet()**).

- **$pathToImportDir** Chemin, relatif au site, vers le répertoire contenant les modèles d'importation.
  Ce paramètre est optionnel : si vide, il sera renseigné avec la valeur du paramètre **PathToImportModels de la table MySQL « spip_params_appli »**. Si ce dernier paramètre est vide, alors $pathToImportDir = 'Tools/Modeles_import'

NB : La procédure utilise aussi la variable globale « $userId » donnant l'id de l'auteur Spip identifié.

Avant importation, la procédure « **importObjectIntoSpip()** » renseignera la valeur des attributs suivants, s'ils ne sont pas déjà renseignés :

- $attributs['Heure'] = <hh :mm :ss> // 13:57:23
- $attributs['DateJour'] = <aaaa-mm-jj> // 2002-01-15
- $attributs['Date'] = <aaaa-mm-jj> <hh :mm :ss>
- $attributs['DateRedac'] = <aaaa-mm-jj> <hh :mm :ss>
- $attributs['DateFr'] = <jj/mm/aaaa> // 15/01/2002
- $attributs['id_auteur'] = $userId // id de l'auteur Spip identifié
- $attributs['idAuteurModif'] = $userId; // si création d'une nouvelle fiche

et pour chaque mots-clés « <mot> » existant :
- $attributs["id_mot_<mot> "] = <id du mot-clé>
- $attributs["id_groupe_<mot> "] = <id du groupe contenant le mot-clé>
- $attributs["nom_groupe_<mot> "] = <type du mot-clé>

Pour effectuer l'importation, la procédure « **importObjectIntoSpip()** » remplacera dans chaque modèle d'importation à traiter, les chaînes '%xxx%' par la valeur de l'attribut $attributs['xxx'], puis **appelera la procédure Spip import_objet()** avec ce modèle en paramètre.

Après importation, la procédure « **importObjectIntoSpip()** » renseignera, pour le 1er objet importé dans Spip, la valeur des attributs suivants :

- $attributs['first_objetSpip'] = 'rubrique' ou 'article' ou ...
- $attributs['first_id_rubrique'] ou $attributs['first_id_article'] ou ...
- $attributs['id_rubrique'] ou $attributs['id_article'] ou ...

et retournera une valeur 'vraie' si l'importation s'est bien passé. En cas d'erreur, la variable globale **$errNouvelleFiche** contiendra le message d'erreur.

3. Principes de traitement des « Fiches XML »

3.1. Le type d'une FICHE XML

On peut être amené à gérer plusieurs types de fiches : par exemple, fiche "Document", fiche "Client", fiche "Fournisseur", …
Le type d'une fiche sera donné par l'attribut "TypeDocXml" de la balise XML "<DocumentXML>" .

> Par exemple : Introduction de la fiche descriptive d'un client par
> **<DocumentXML TypeDocXML="FicheClient">**
>
> Par exemple : Introduction de la fiche descriptive d'un devis par
> **<DocumentXML TypeDocXML="DevisPiscine">**

3.2. Le traitement d'une FICHE XML : Modèle et « parsers »

Le traitement d'une fiche XML sera contrôlé par un « modèle de fiche XML », et sera effectué par des « parsers » XML .

Un « modèle de fiche XML » et les « parsers » associés sont spécifiques à un type de fiche XML :

- **A chaque type de fiche XML sera associé un couple de « parsers » XML** chargés de générer le code HTML nécessaire pour visualiser le contenu d'une fiche ou pour présenter le formulaire de saisie / modification de cette fiche.
  Chaque « parser » est composé d'un ensemble de procédures prenant en charge les traitements à effectuer sur chaque balise XML de la fiche :
  - Lors de son ouverture
  - Sur son contenu
  - Lors de sa fermeture

  Chaque parser à utiliser est identifié par l 'attribut "Handler" de la balise XML "<DocumentXML>" :
  - Dans la fiche, cet attribut identifie le « parser » à utiliser pour traiter la valeur des différents champs de la fiche
  - Dans le « modèle de fiche XML », cet attribut identifie le « parser » à utiliser pour afficher la fiche et le formulaire associé.
  - **Règle de nommage : Si le « parser » de la fiche XML est « xxx », celui de son modèle doit être « Modelexxx »**

  Le couple de parsers génériques par défaut est
  - « Handler="FicheXML" » pour la fiche
  - « Handler="ModeleFicheXML" » pour son modèle

  NB : Les 2 « parsers » sont donc toujours utilisés, quel que soit le contexte.

- **A chaque type de fiche XML sera associé un « modèle de fiche XML »** qui sera interprété par le parser. Ce modèle décrit le code HTML à générer, ainsi que les éventuels traitements à effectuer, pour chaque balise XML de la fiche :
  - Dans le cas de l'affichage du contenu de la fiche, le modèle décrit la mise en page de la fiche
  - Dans le cas de la présentation du formulaire de saisie ou de modification d'une fiche, le même modèle décrit la mise en page du formulaire et les valeurs par défaut des différents champs
  - **Règle de nommage : Si le type de la fiche XML est « yyy », celui de son modèle doit être « Modeleyyy »**

Par exemple :
Première ligne de la fiche descriptive d'un utilisateur :
<DocumentXML TypeDocXML="FicheUser" Handler="FicheXML">

Première ligne du modèle associé :
<DocumentXML TypeDocXML="ModeleFicheUser" Handler="ModeleFicheXML">

3.3. Le contenu d'un modèle de FICHE XML

Un modèle de fiche XML va décrire chaque champ (attribut) de la fiche. La description devra donner les informations nécessaires tant à l'affichage du champ, qu'à sa saisie dans un formulaire.

Un modèle de fiche XML aura la structure suivante (la description complète de tous les attributs possibles est donnée en annexe) :

NB: Dans la description ci-dessous :
- les balises XML et leurs attributs sont en **gras**
- les attributs en **gras** sont choisis par le développeur dans une liste imposée
- les mots en *italique* ou en ***gras-italique*** sont choisis librement par le développeur
- seuls ont été mentionnés les principaux attributs associés aux différentes valeurs de l'attribut « TypeInput »

<DocumentXML TypeDocXML="*Fiche***">**

<Cadre Titre="*titre du groupe de champs***">**

<*nomChampDeTypeTexte* **Label="** *texteDsFormulaire* **" InputOblig="" TypeInput="Texte"**
Cmt ="*cmtDsFormulaire* **"**
>
Valeur par défaut du champ
</*nomChampDeTypeTexte*>

<*nomChampDeTypeTextArea* **Label="** *texteDsFormulaire* **" InputOblig=" *" TypeInput="TextArea"**
NbLignes="*nbLignesBoite* **"**
Cmt="*cmtDsFormulaire* **"**
>
Valeur par défaut du champ
</*nomChampDeTypeTextArea*>

<Titre Titre="*titre sans cadre associé* **" />**

<*nomChampDeTypeSelect* **Label="***texteDsFormulaire* **" InputOblig=" *" TypeInput="Select"**
Table="spip_*table* **"**
ChpValOption="*champTable, champTable , champTable* **"**
ChpTxtOption="*champTable* **"**
Test1="*critère***"**
Cmt="*cmtDsFormulaire* **"**
>
Valeur par défaut du champ
</*nomChampDeTypeSelect*>

</Cadre>

</DocumentXML>

3.4. Le traitement d'une FICHE XML : les « parsers »

Le traitement d'un document XML fait appel au parser XML fourni par PHP. Ce parser est principalement composé de procédures appelées lors du parcours du document, pour chaque entrée, contenu et fin de balise rencontrés.

Le traitement d'une « fiche XML » est réalisé par 2 « parsers » dont les procédures permettront de générer le code nécessaire à l'affichage de la fiche ou du formulaire associé.

NB : Pour plus d'information sur l'utilisation du parser et des procédures associées, voir la documentation PHP.

Les 2 « parsers » associés au type de fiche "*type_de_fiche*" sont donnés par les fichiers
« DocXML_*type_de_fiche*.php »
et « DocXML_Modele*type_de_fiche*.php »

Chacun de ces fichiers contient la définition des procédures de traitement appelées lors de l'analyse de la « fiche XML » :

« **DocXML_*type_de_fiche*.php** » fournit les procédures :
- *type_de_fiche*_startElementHandler() : Ouverture d'une balise XML
- *type_de_fiche*_characterDataHandler() : Contenu d'une balise XML
- *type_de_fiche*_endElementHandler() : Fermeture d'une balise XML

« **DocXML_Modele*type_de_fiche*.php** » fournit les procédures :
- Modele*type_de_fiche*_startElementHandler()
- Modele*type_de_fiche*_characterDataHandler()
- Modele*type_de_fiche*_endElementHandler()

Pour être exact, le filtre « processXML » fournit à un « parser » commun à tout document XML ses propres procédures de traitement des balises XML. Ce sont ces procédures communes qui vont appeler les procédures spécifiques associées au type de fiche traitée.

Lors de l'analyse d'une « fiche XML », le fonctionnement du filtre « processXML » est le suivant :

- La balise <DocumentXML> est recherchée au début du texte à traiter.
  Si elle n'est pas trouvée, on applique le filtre « propre » au texte et on retourne le résultat.

- Un parser PHP est initialisé et activé sur le texte à traiter (procédures communes).

- La 1ère balise ouvrante rencontrée (<DocumentXML>) est soumise à la procédure commune « startElementHandler » qui analyse ses attributs (TypeDocXML et Handler) et les transmet aux autres procédures communes « characterDataHandler » et « endElementHandler » : par la suite, tout appel à l'une des procédures commune sera relayée à la procédure spécifique du handler spécifié.

- Dans le cas du traitement d'un « modèle de fiche XML », chaque balise rencontrée est traitée par les procédures spécifiques au type de fiche. Le code HTML adéquat est généré, et les valeurs par défaut des balises sont sauvegardées.

- Dans le cas du traitement d'une « fiche XML » :

  - La 1ère balise ouvrante (<DocumentXML>) est traitée par la procédure spécifique « *type_de_fiche*_startElementHandler » qui va d'abord ré-activer le filtre « processXML » sur le modèle associé (appel ré-entrant du filtre avec un 2ème paramètre « *type_de_fiche* »).

Le modèle à traiter sera alors recherché, d'abord dans le fichier

« Tools/Modeles_fichesXML/modele*type_de_fiche*.xml »

puis, si ce fichier n'existe pas, dans le texte de l'article Spip ayant les caractéristiques suivantes :

- « Surtitre='Modele*type_de_fiche*' »
- Mot-clé « TypeArticle='ModeleFicheXML' »

❖ Toutes les balises du modèle sont alors traitées par le « parser » associé, tel que décrit ci-dessus (1er cas : Traitement d'un « modèle de fiche XML »).

❖ Le modèle ayant été traité, la procédure « *type_de_fiche*_startElementHandler » termine sont traitement et l'analyse de la fiche se poursuit (les valeurs de chaque balise sont sauvegardées).

❖ Lorsque la balise fermante </DocumentXML> marquant la fin de la fiche est traitée par la procédure « *type_de_fiche*_endElementHandler », les valeurs des balises de la fiche sont renseignées dans le code généré par le traitement du modèle, puis ce dernier est retourné à l'appelant du filtre.

4. La distribution « SPIP – XML »

4.1. Les composants

La distribution est composée de la présente documentation '**SPIP-XML.pdf**' et d'une archive '**Spip-XML.zip**' dont les fichiers doivent être extraits dans le répertoire contenant le site sous Spip.

Les fichiers livrés (et leurs répertoires) peuvent être regroupés en 3 familles :

- Les fichiers d'installation (voir leur utilisation dans le chapitre « Mise en œuvre »)
- Les squelettes utilisés dans l'exemple d'application « Devis »
- Le filtre « processXML » et les procédures associées au traitement des fiches XML

Les fichiers d'installation

❖ **install_Spip-XML.php** : Fichier à exécuter après extraction de l'archive « Spip-XML.zip ».

❖ **Répertoire « Install_SPIP-XML »** : Répertoire créé par l'extraction de l'archive « Spip-XML.zip ».

- **install_MotsCles.php** : Définition des mots-clés Spip utilisés par le package « SPIP-XML ».
  NB : Fichier appelé par le fichiers d'installation 'install_Spip-XML.php'.

- **install_Modeles.php** : Définition des modèles de « Fiches XML » utilisés dans cette documentation et utilisés par l'exemple d'utilisation « Devis »».
  NB : Fichier appelé par le fichier d'installation 'install_Spip-XML.php'.

- **spip_params_appli.sql** : Structure et données de la table MySQL '<préfixe des tables>_params_appli' contenant les paramètres du package « SPIP-XML » (et de l'application).
  NB : Ce fichier est exploité par le fichier d'installation 'install_Spip-XML.php'.

- **Répertoires « Modifs_SPIP_xxx »** : Fichiers modifiés dans SPIP Version xxx
  Chacun de ces répertoires correspond à une version de Spip :
  - **Modifs_Spip_1.465** : édition 1.465 de Spip 1.4
  - **Modifs_Spip_1.604** : édition 1.604 de Spip 1.6

  NB : Voir l'utilisation de ces répertoires dans le chapitre « Mise en œuvre ».

Les squelettes de l'exemple « Devis »

❖ **affLienIdentification.php3** : Squelette à inclure là où on veut afficher un lien vers une identification Spip. Ce lien active un popup nommé « loginSpip » et y affiche le **squelette « login.php3 »** (cf. login-xml.html ci-dessous).
Voir un exemple d'utilisation de ce squelette dans l'exemple « Devis ».

❖ **login.php3** : Activation du squelette « login.html »

- **login-xml.html** : Squelette associé à « login.php3 » (**à renommer « login.html »**)
  Ce squelette, ouvert dans un popup nommé « loginSpip », permet à l'internaute de s'identifier (« auteur » connu de Spip).
  Si l'internaute est déjà identifié, ce popup lui permet de se déconnecter ou, s'il est « administrateur non restreint », d'accéder à l'espace privé de Spip (ouvert alors dans une fenêtre séparée).
  Après identification ou déconnexion, la fenêtre ayant activé le popup est rafraîchie par appel au fichier « refreshLoginOpener.php ».

  Une fois l'utilisateur identifié, la fonction booléenne « adminRubrique(<id_rubrique>) » (définie via le fichier «Tools/paramsAppli.php ») permet de savoir s'il administre une rubrique donnée (voir un exemple d'utilisation dans le squelette « affLiensNewFichesXML » ci-dessous).

  NB : Ce squelette est compatible avec son utilisation par l'espace privé de Spip.

- **refreshLoginOpener.php** : Activé par « login-xml.html » pour rafraîchir la fenêtre ayant ouvert le popup d'identification Spip.
  Le rafraichissement permet de modifier l'affichage de la fenêtre si celle-ci exploite le fait que l'utilisateur s'est identifié ou non.

- **affLiensNewFichesXML.php3**: Activation du squelette « affLiensNewFichesXML.html »

- **affLiensNewFichesXML.html** : Squelette à inclure là où on veut afficher les **liens vers les formulaires de création de nouvelles « Fiches XML » associées à la rubrique en cours**.

  Dans ce squelette, une fiche est dite « **associée** » à une rubrique si son type contient :
  - le contenu de la **variable globale « $xml_TypeFicheXML »** si celle-ci est non-vide
  - ou le **mot-clé de type « TypeRubriqueXML »** de la rubrique

  Par exemple, si $xml_TypeFicheXML='Devis' ou si la rubrique a le mot-clé TypeRubriqueXML='Devis' , ce squelette affichera les liens vers le formulaire de création de fiches de type « DevisPiscine » ou « mon_Devis_a_moi »…
  NB : En fait, on recherchera les modèles de fiche 'Modele…Devis…'

- **rubrique-xml.html** : Squelette livré pour l'exemple d'utilisation « Devis » (**à renommer « rubrique.html »**).
  C'est une version de « rubrique-dist.html » montrant une utilisation du XML dans Spip.
  Ce squelette affiche un lien vers une identification « Spip » par inclusion du squelette « **affLienIdentification** », puis, si l'utilisateur est administrateur de la rubrique en cours ou d'une de ses parents (cf. fonction « **adminRubrique(id_rubrique)** », affiche les liens vers les formulaires de création des fiches associées à la rubrique par inclusion du squelette « **affLiensNewFichesXML** »,.
  NB : Les lignes ajoutées ou modifiées par rapport à « rubrique-dist.html » sont identifiées par le commentaire '/// XML ///'.
  NB : Ce squelette gère aussi une redirection vers le squelette désigné par l'éventuel mot-clé de type « Squelette », et il n'affiche pas de liens vers des rubriques ou articles ayant le mot-clé « Dummy » ou « ModeleFicheXML ».

- **article-xml.html** : Squelette livré pour l'exemple d'utilisation « Devis » (**à renommer « article.html »**).
  C'est une version de « article-dist.html » montrant l'utilisation du XML dans Spip.
  On a simplement ajouté l'appel au filtre « processXML » lors du traitement du 'TEXTE' de l'article.
  NB : Les lignes ajoutées ou modifiées par rapport à « article-dist.html » sont identifiées par le commentaire '/// XML ///'.

Le filtre « processXML » et le traitement des fiches XML

- **fonctionsXml.php** : Filtre « processXML » et interface avec les parsers.
  Le filtre « processXML » est utilisé pour traiter le contenu XML d'un objet SPIP (typiquement un article ou une rubrique). Ce fichier contient aussi l'ensemble des fonctions nécessaires au fonctionnement des parsers décrits ci-dessous.

  Exemple d'utilisation : [(#TEXTE*|processXML)]

- **DocXml_FicleXml.php** : Parser XML par défaut.
  Ce parser traite une « Fiche XML ». Il active le parser traitant le modèle de la fiche et renseigne les valeurs des différents champs dans le code html visualisant la fiche ou le formulaire associé.

- **DocXml_ModeleFicleXml.php** : Parser XML par défaut.
  Ce parser traite un modèle de « Fiche XML » et génère le code html visualisant une fiche et le formulaire associé.

- **nouvelleFiche.php** : Importation d'une nouvelle fiche dans Spip.
  Ce fichier est activé par la soumission du formulaire de création / modification d'une fiche.

- **mes_fonctions.php** : Filtres divers.
  NB : Ce fichier appelle « Tools/paramsAppli.php » et « mes_fonctionsAppli.php ».

- **mes_fonctions-xml.php3** : Simple inclusion du fichier « mes_fonctions.php »
  (**à renommer « mes_fonctions.php3 »**).
  NB : Si le fichier 'mes_fonctions.php3' existe déjà dans le site, y reporter simplement l'inclusion du fichier 'mes_fonctions.php'.

- **mes_fonctionsAppli.php** : Procédures et filtres utilisées lors du traitement des fiches.
  En particulier, ce fichier définit :

  - Le **filtre « balisesXML »** qui convertit les balises d'une « Fiche XML » en un tableau associatif.
    Notez que seules les valeurs sont retournées, et que les attributs des balises ne sont pas disponibles.

    Exemple d'utilisation : <? $tableau = balisesXML('[(#TEXTE*|texte_script)]') ; ?>
    On a alors : $tableau[<balise>] = <valeur_balise> .

  - La procédure **« importObjectIntoSpip(objetAppli, attributs) »** qui prend en charge l'importation d'un objet applicatif de type « objetAppli » et dont les attributs seront donnés par le tableau associatif « attributs ».
    Une variable globale « $import_<objetAppli> » devra être définie (par exemple dans le fichier Tools/paramsAppli.php) : elle contiendra la séquence des fichiers d'importation Spip élémentaires à traiter (cf répertoire Tools/Modeles_import décrit ci-dessous).
    Cette fonction retourne « vrai » si l'objet a été importé dans la base Spip, retourne le texte explicatif de l'erreur sinon.
    Si l'importation a été effectuée, le tableau « attributs » contiendra en plus au retour :
    $attributs['first_objetSpip'] => 'rubrique' ou 'article' ou ... (i-e type du 1er objet Spip créé)
    $attributs["first_id_".$attributs['first_objetSpip']] => id du 1er objet Spip créé
    $attributs["id_".$attributs['first_objetSpip']] => id du 1er objet Spip créé

    Exemple d'utilisation :

Supposons le code suivant :

```
$attributs = array(
    'id_rubrique'          => 0,
    'id_secteur'           => $idSecteur,
    'id_parent'            => $idRubrique,
    'TitreRubrique '       => $TitreRubrique,
    'DescriptionRubrique' => $DescriptionRubrique,
    'TexteRubrique'        => $TexteRubrique
                    ) ;
importObjectIntoSpip('RubPrivee', $attributs);
```

Si le fichier « Tools/paramsAppli.php » définit la variable globale

```
$GLOBALS['import_RubPrivee']   = 'import_Rubrique_Std.xml,'.
                                 'import_Mot_Prive.xml' ;
```

alors l'appel ci-dessus à la fonction « importObjectIntoSpip » créera une rubrique Spip

- avec le mot-clé « Prive » (qui devra préexister dans Spip)
- et dont le titre, la description, … seront renseignés avec les variables utilisées dans le code pris en exemple.

NB : Le fichier 'import_Mot_Prive.xml' devra être créé en prenant comme exemple les fichiers 'Tools/Modeles_import/import_Mot_xxx.xml'.

Autres exemples d'utilisation :

Voir les fichiers d'installation du package « SPIP-XML » :

- install_Spip-XML.php
- Install_SPIP-XML/install_Modeles.php
- Install_SPIP-XML/install_MotsCles.php

❖ Répertoire « Ressources »

Ce répertoire contiendra les fichiers éventuellement chargés via le champ d'une « Fiche XML ».
Cf. la procédure « getUploadedFile » dans le fichier « Tools/getUploadedFile.php », ainsi que le paramètre « **PathToRessources** » **de la table MySQL « spip_appli_params ».**

❖ Répertoire « icones »

Ce répertoire contient des icônes… dont certaines sont utilisées par ce package !

❖ Répertoire « Tools »

- **paramsAppli.php** : Paramètres de l'application.
  C'est dans ce fichier que devront être définies les variables globales « import_<type_de_fiche> » donnant les modèles d'importation dans Spip (voir le paragraphe « Mise en œuvre » ci-dessous).
  Ce fichier inclut les fichiers « paramsSpip.php » et « paramsXML.php ».

- **paramsSpip.php** : Paramètres associés à l'utilisation de Spip par l'application.
  Ce fichier définit la procédure « **adminRubrique($id_rubrique)** » qui retourne la valeur booléenne « vrai » si l'internaute est identifié comme connu de Spip avec les droits

d'administration de la rubrique « $id_rubrique » ou de l'une de ses parents.

Ce fichier renseigne la variable globale

$userIdentified	Vrai si l'internaute est connu de Spip

Si l'internaute est connu de Spip, alors ce fichier renseigne aussi les variables globales

$userId	Identifiant interne de Spip (id_auteur)
$userName, $userLogin, $userEmail	Informations connues de Spip
$userRights	Droits de l'internaute dans l 'espace privé de Spip : ='admin', ='adminRestreint', ='redacteur' ou ='forum'

- **paramsXml.php** : Ce fichier définit la procédure
  « setParamsXML($typeFicheXML, $id_article_modele) »
  qui renseigne les différentes variables globales nécessaires au traitement des fiches XML.

  Les paramètres d'appel de cette fonction sont :

  $typeFicheXML : Type de la fiche à traiter. Si vide, le type sera donné
  - par la variable globale "$xml_TypeFicheXML"
  - ou par le mot-clé de type « TypeFicheXML » de la rubrique ou de l'article en cours

  $id_article_modele : N° de l'article Spip contenant le modèle de la fiche à traiter. Si vide, sera renseigné en recherchant l'article
  - de mot-clé "ModeleFicheXML"
  - et de surtitre "Modele<type_de_fiche>"

- **getUploadedFile.php** : Ce fichier définit la procédure « **getUploadedFile(…)** » utilisée lors de la création ou modification d'une fiche XML dont l'un des champs est un fichier à charger sur le serveur.
  NB : Seules certaines extensions sont autorisées (.txt, .csv, .pdf, .doc, .xls, .jpg, .jpeg, .gif, .zip, .tar, .gz ,.z).
  NB : Le paramètre « **PathToRessources » de la table MySQL « spip_appli_params »** donne le chemin, relatif au site, vers le répertoire où seront rangés les fichiers chargés (défaut « Ressources »).

- **site.css** : Feuille de style appelée dans les squelettes distribués (en plus de spip_style.css).

- **util.js** : Variables et procédures JavaScript (utilisé pour les dimensions de la fenêtre de login)

- **classeDebug.php** : Support aux moyens de debug intégrés au code php.

- **<u>Répertoire « Tools/Modeles_fiches_XML »</u>**

  Ce répertoire rassemble les modèles de fiches (répertoire par défaut ; sinon, les modèles de fiches sont dans le répertoire dont le chemin, relatif au site, est donné par le paramètre « **PathToModelsXML » de la table MySQL « spip_appli_params »**).

  NB : Même si les fichiers sont utilisés à la place des articles Spip, ces derniers doivent exister (mot-clé, titre et surtitre renseignés) avec un texte contenant une balise <DocumentXML> sans contenu mais avec ses attributs « TypeDocXML » et « Handler » correctement renseignés.

NB : Ce répertoire par défaut accueillera les modèles livrés avec le package.

- **Répertoire « Tools/Modeles_import »**

  Ce répertoire rassemble les fichiers contenant les modèles d'importation des objets Spip (répertoire par défaut ; sinon, les modèles d'importation sont dans le répertoire dont le chemin, relatif au site, est donné par le paramètre « **PathToImportModels » de la table MySQL « spip_appli_params »**).

  NB : Ce répertoire par défaut accueillera les modèles livrés avec le package.

4.2. La mise en œuvre de l'installation : Installer les fichiers

Installation du package « Spip-XML »

Commencez par installer et configurer Spip dans un répertoire que nous appellerons ici « SiteSpip ».

Ensuite, extrayez l'archive « Spip-XML.zip » dans ce répertoire (ou dans le répertoire d'un site existant basé sur Spip).

Modifications des fichiers Spip

Le répertoire « Install_SPIP-XML », créé à la base de votre site lors de l'extraction de l'archive « Spip-XML.zip », contient des répertoires « Modifs_SPIP_v.vvv » donnant les 3 fichiers Spip qui doivent remplacer ceux existant de même nom :

- inc-login.php3
- ecrire/inc_import.php3
- ecrire/inc_texte.php3

Pour installer les « bons » fichiers, visualisez le fichier « ecrire/inc_version.php3 » pour connaître la version « v.vvv » de la base (donnée par l'affectation de la variable « $spip_version » vers la ligne 150 du fichier).

Par exemple :

```
……
// version de la base
$spip_version = 1.604;

// version de spip
$spip_version_affichee = "1.6";
……
```

=> version 1.604

Si l'un des répertoires « Install_SPIP-XML/Modifs_SPIP_v.vvv » correspond à la version de Spip installée dans votre site ('Install_SPIP-XML/Modifs_SPIP_1.604' dans l'exemple ci-dessus), copiez son contenu dans le répertoire « SiteSpip » (les fichiers modifiés remplaceront automatiquement, mais avec votre accord, les fichiers existants de même nom).

Sinon, choisissez le répertoire le mieux adapté à votre version de Spip (i-e la version antérieure la plus « proche »), visualisez les modifications introduites dans les fichiers Spip (les lignes modifiées sont identifiées par la chaîne « /// XML /// ») et reportez manuellement ces modifications dans les fichiers de même nom présents dans votre site.

Renommage des squelettes et fichiers livrés

Pour utiliser le package :

Si le fichier 'mes_fonctions.php3' existe déjà dans votre site, ajoutez à sa fin l'instruction php :

```
include 'mes_fonctions.php' ;
```

Sinon, renommez le fichier 'mes_fonctions-xml.php3' en 'mes_fonctions.php3' .

Pour activer l'exemple « Devis » livré avec le package, renommez les squelettes suivants (ou reportez leur contenu dans les fichiers existants) :

- article-xml.html => article.html
- rubrique-xml.html => rubrique.html
- login-xml.html => login.html

4.3. La mise en œuvre de l'installation : Mettre à jour la base

Une fois l'archive installée et les fichiers Spip modifiés, exécutez le programme d'installation en tapant l'url suivante dans votre navigateur :

http://<url_de_votre_site>/install_Spip-XML.php

Les actions suivantes sont effectuées :

Création des groupes de mots-clés utilisés par le package
Création de la table MySQL « spip_appli_params » si elle n'existe pas déjà.
NB : L'éventuel préfixe des tables de votre site sera pris en compte.
NB : Cette table contient des paramètres qui deviendront des variables globales, en particulier :

$PathToModelsXML
$PathToImportModels

- Création des groupes de mots-clés utilisés par le package

- Copie des modèles d'importation non présents dans le répertoire donné par le paramètre « PathToImportModels » de la table « spip_appli_params » (si non vide et différent de 'Tools/Modeles_import')

- Copie des modèles de fiches non présents dans le répertoire donné par le paramètre « PathToModelsXml » de la table « spip_appli_params » (si non vide et différent de 'Tools/Modeles_fiches_XML')

- Création de la rubrique « Modèles de Fiches XML » et création des articles définissant les modèles présents dans le répertoire 'Tools/Modeles_fiches_XML' (seule balise <DocumentXML> si « PathToModelsXml » est renseigné)

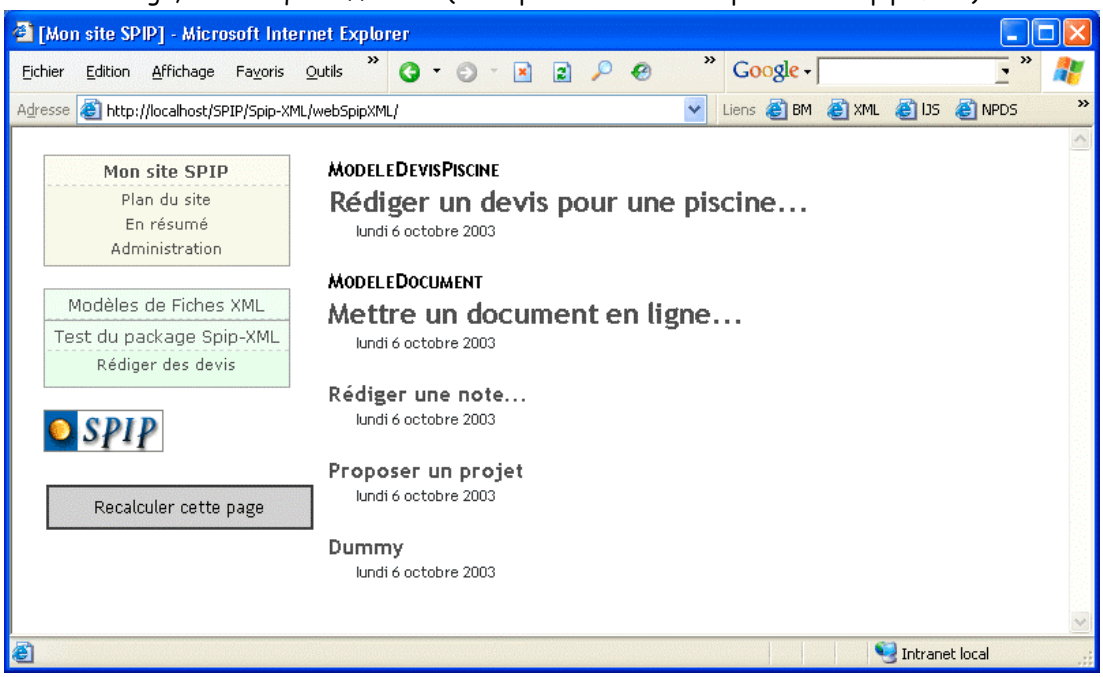
- Création des articles contenant les modèles de fiches.
  NB : Les modèles déjà présents dans le répertoire donné par le paramètre « PathToModelsXml » de la table « spip_appli_params » (si non vide et différent de 'Tools/Modeles_fiches_XML') ne sont pas créés ou mis-à-jour.

- Création de la rubrique « Test du package Spip-XML » et de la sous-rubrique « Rédiger des devis » (exemple d'utilisation)

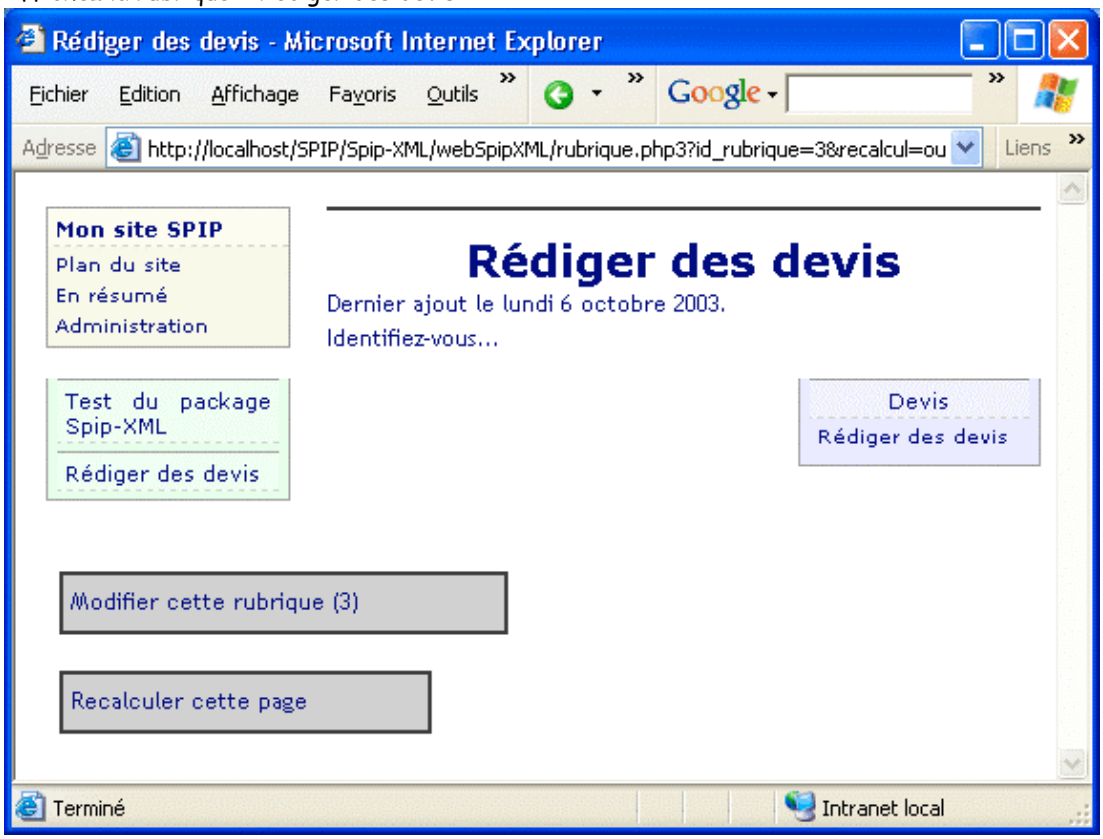
4.4. La mise en œuvre de l'installation : Activer l'exemple « Devis »

Visualisez votre site (http://<url_de_votre_site>).

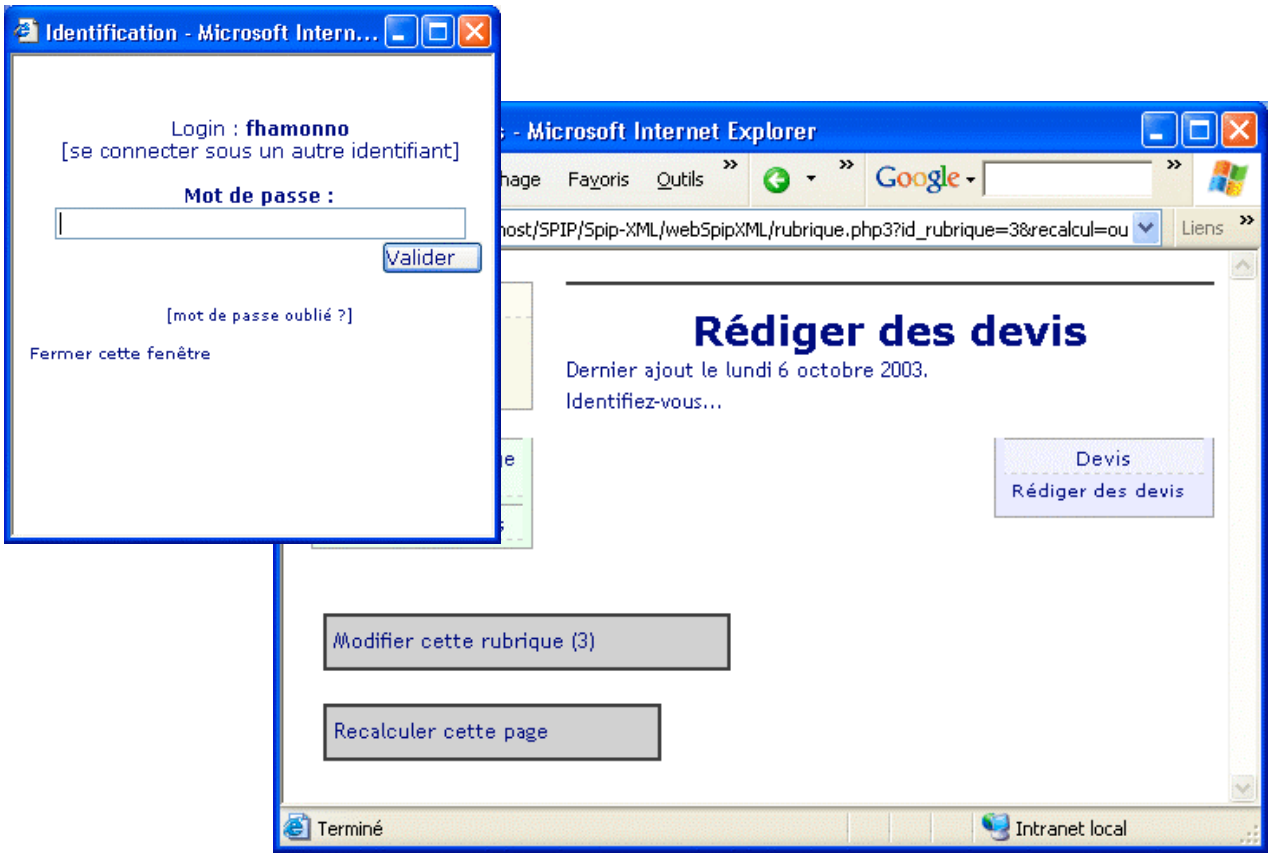
S'il était vierge, voici ce qui s'affichera (les copies d'écran correspondent à Spip 1.604) :



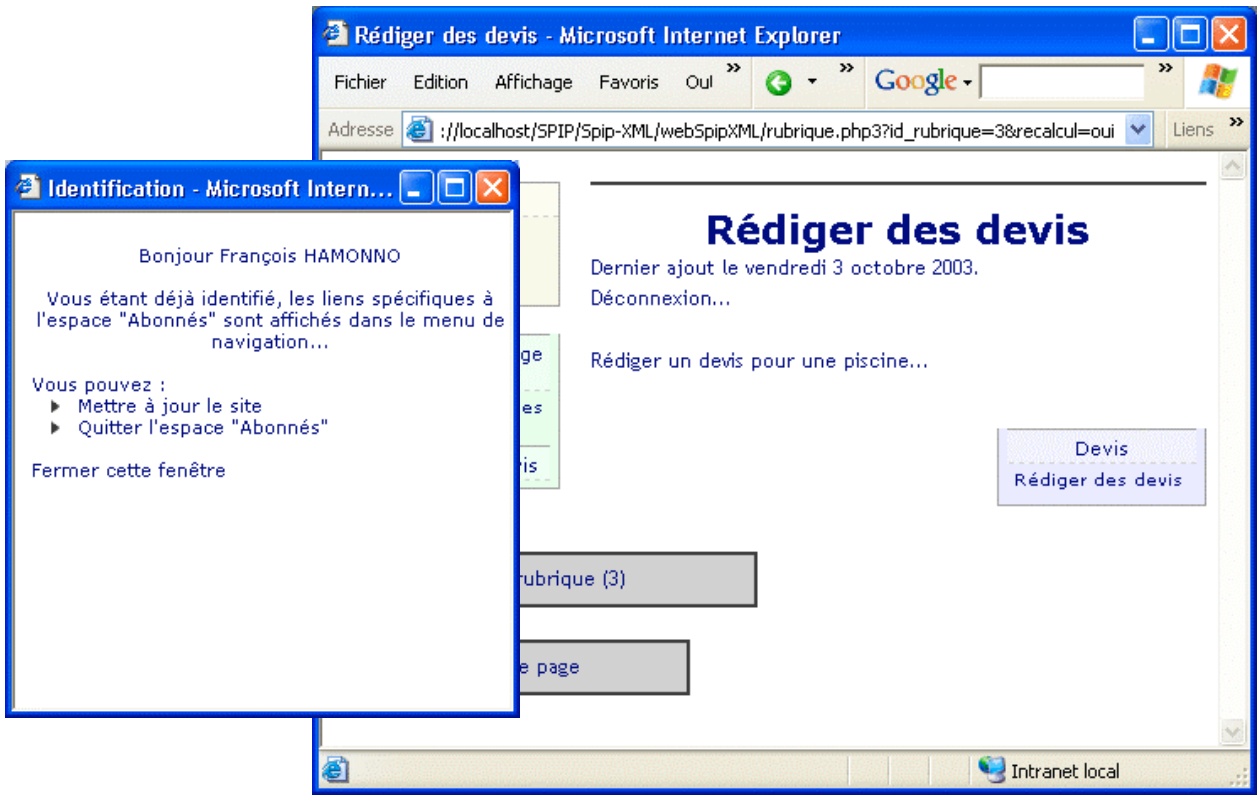
Affichez la rubrique « Rédiger des devis » :



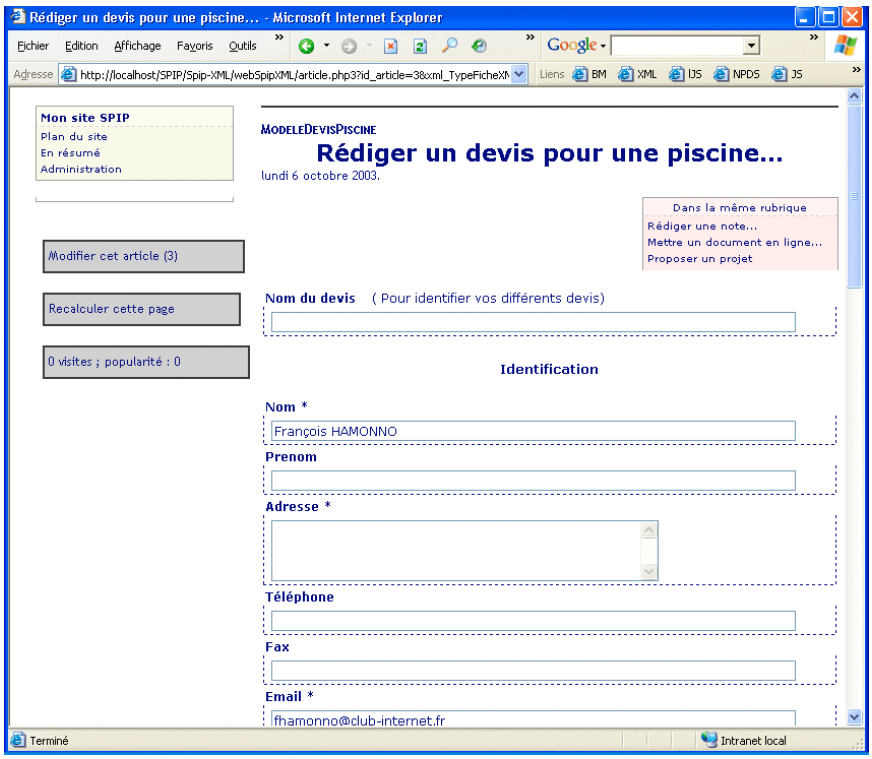
Activez le lien « Identifiez-vous ». Le popup suivant s'affiche :



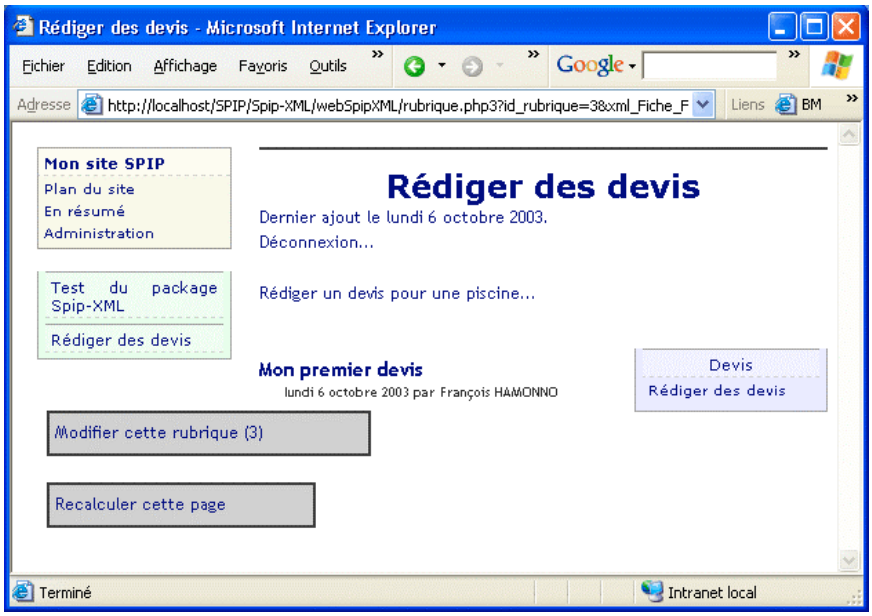
Après avoir entré le login et le mot de passe qui vous donne accès à l'espace privé de Spip, les fenêtres deviennent :



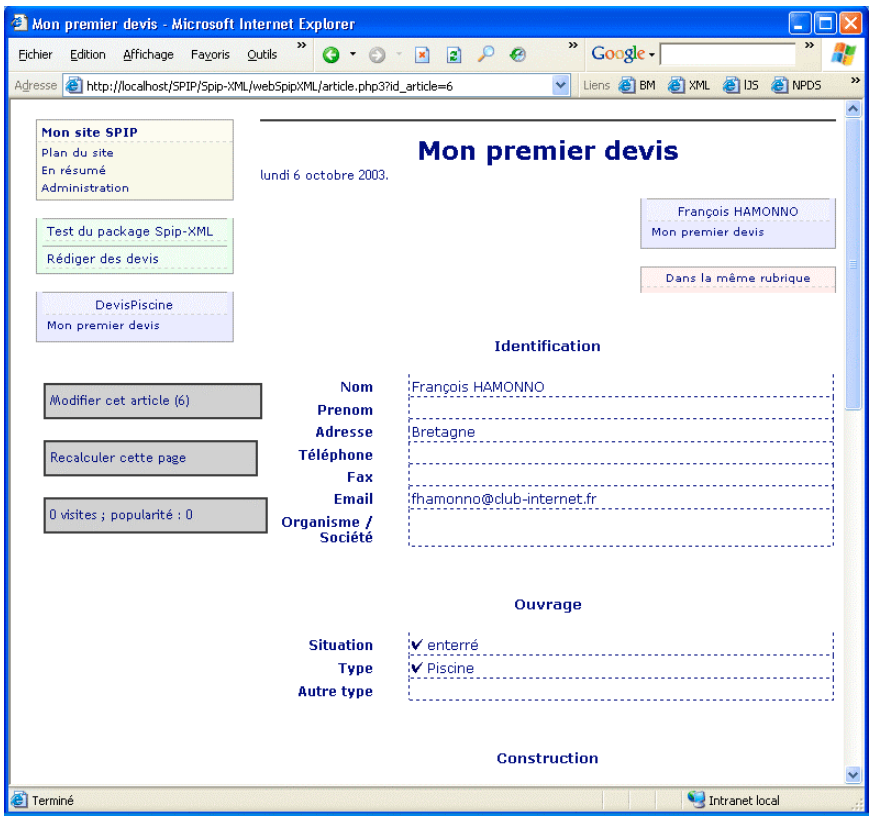
Activez le lien « Rédiger un devis pour une piscine » : le formulaire s'affiche (avec les champs « Nom » et « Email » renseignés avec les paramètres de Spip)



Après renseignement du formulaire et enregistrement, la page ayant activé le formulaire est re-affichée :



Si vous cliquez sur le nom du devis, sa fiche s'affiche et, si vous êtes « identifiée », un lien vous permet de la modifier :



Annexe 1 – Modèle de FICHE XML : Syntaxe des balises et de leurs attributs

Balise <DocumentXML>

Cette balise contient la totalité de la « fiche XML » et doit être le contenu intégral d'une balise Spip (typiquement : texte d'un article ou d'une rubrique).

Attributs

TypeDocXML="Modele*type_de_fiche*"

Identifie le type de fiche auquel est associé le modèle.

Une fiche de ce type commencera par la balise <DocumentXML TypeDocXML="*type_de_fiche*" >
Au type de fiche "*type_de_fiche*" devra correspondre une variable globale
« $import_*type_de_fiche* » donnant la séquence des fichiers d'importation à traiter pour importer la fiche dans Spip :

$import_*type_de_fiche* = '*fichier1*.xml, *fichier2*.xml,…, *fichierk*.xml' ;

NB : Ces fichiers sont attendus dans le répertoire « Tools/Modeles_import »
Les variables « $import_*type_de_fiche* » sont définies dans le fichier « Tools/paramsAppli.php »
L'importation dans Spip de la fiche est assurée par le fichier « nouvelleFiche.php » activé suite à la validation du formulaire de saisie / modification d'une fiche.

Handler="Modele*parser*"

Identifie le parser à utiliser pour générer le code HTML d'affichage de la fiche et du formulaire associé à la fiche à traiter.
Rappel : Celle-ci commencera par la balise <DocumentXML Handler="*parser*" … >

Format

Type de présentation par défaut des champs dans le cas de l'affichage de la fiche.
Syntaxe :

Format="Horizontal" : Le libellé du champ et sa valeur sont sur la même ligne
Format="Vertical" : Le libellé du champ est sur une ligne, la valeur est sur une autre ligne

FormatForm

Type de présentation par défaut des champs dans le cas de l'affichage du formulaire.
Syntaxe :

FormatForm="Horizontal" : Le libellé du champ, sa valeur et le commentaire associé sont sur la même ligne (cf. tableau html)
FormatForm="Vertical" : Le libellé du champ et le commentaire associé sont sur une ligne, la valeur est sur une autre ligne

Html_Titre

Code html par défaut à afficher avant les titres. Ce code sera inséré avant chaque titre produit par une balise <Cadre> ou <Titre> .
Syntaxe : Html_Titre="*code html*".
Les caractères '<' et '>' doivent être remplacés par le caractère '§'.

Par exemple, Html_Titre="§BR /§" génèrera le code html "
"

Class_Titre

Style par défaut à utiliser pour l'affichage des titres (cf. balises <Cadre> et <Titre>).
Syntaxe : Class_Titre="*classe définie dans une feuille de style*"

Class_Label

Style par défaut à utiliser pour l'affichage du libellé des champs
Syntaxe : Class_Label="*classe définie dans une feuille de style*"

Class_Valeur

Style par défaut à utiliser pour l'affichage de la valeur des champs
Syntaxe : Class_ Valeur ="*classe définie dans une feuille de style*"

Class_Valeur_Form

Style par défaut à utiliser pour l'affichage de la valeur des champs dans les formulaires.
Défaut : L'attribut "Class_Valeur"
Syntaxe : Class_Valeur_Form="*classe définie dans une feuille de style*"

Class_Cmt

Style par défaut à utiliser pour l'affichage du commentaire associé à la saisie des champs
Syntaxe : Class_Cmt="*classe définie dans une feuille de style*"

Width

Largeur des encadrements éventuels.
Défaut : 40 caractères
Syntaxe : Attribut « WIDTH » de la balise html <TABLE>.

Border

Epaisseur par défaut à utiliser pour les balises <Cadre>.
Syntaxe : Attribut « BORDER » de la balise html <TABLE>.

BorderForm

Epaisseur par défaut à utiliser pour les balises <Cadre>, lors de l'affichage du formulaire
Défaut : 1 pixel
Syntaxe : Attribut « BORDER » de la balise html <TABLE>.

AlignLabels

Alignement par défaut à utiliser pour le libellé des champs.
Défaut : Alignement à droite des libellés
Syntaxe : AlignLabels="Right"
AlignLabels="Left"

DefautNbCars

Nombre de caractères par défaut dans les boites de dialogue permettant la saisie des champs de type 'texte'.
Syntaxe : Attribut « SIZE » de la balise html <INPUT>.

Traitement

Indicateurs par défaut contrôlant le traitement des champs. Les indicateurs spécifiés ici s'appliqueront à tous les champs de la fiche.
Syntaxe :

Traitement="NoAffVide " : Ne pas afficher les lignes (vides) d'un champ vide

Evnt_*event_js*

Evènement JavaScript à inclure dans la balise html <FORM>.

Syntaxe : Code JavaScript.

« event_js » est un événement JavaScript de la balise html <FORM>.

Exemple : Evnt_onSubmit="if(this.NbSacs.value>10) ↵
{alert('Nombre de sacs trop important (max=10)');return false;}"

NB : Le nom du formulaire html est <*type de fiche XML*>.

Exemple de balise <DocumentXML>

```
<DocumentXML TypeDocXML="ModeleFicheProjet" Handler="ModeleFicheXML"
  Evnt_onSubmit="if(!this.NomRubrique.value || this.NomRubrique.value.substring(0,5)=='Nom d') ↵
      {alert('Nom du projet absent!...');return false;}"
  FormatForm="Vertical"
  Format="Horizontal"
  AlignLabels="left"
  Class_Titre="titre3"
  Class_Label="txtmauveptit"
  Class_Valeur="txtbleunuit"
  Class_Valeur_Form="txtbleunuit"
  Class_Cmt="txtbleunuitsmallitalic"
  DefautNbCars="40"
  Html_Titre="§BR /§§IMG SRC='images/filetmauve.gif' WIDTH='100%' HEIGHT='9'§§BR /§"
  Border="0"
  Width="100%"
  Traitement="NoAffVide" > ..... </DocumentXML >
```

On trouvera par exemple dans un feuille de style, les classes suivantes :

```
.titre3      {font-weight:bold; font-size:12px; text-align:center;}
.txtbleunuit,A.txtbleunuit,A.txtbleunuit:visited,A.txtbleunuit:link,A.txtbleunuit:active {
        font-family: Arial, Helvetica, sans-serif;
        font-size: 12px;
        font-style: normal;
        font-weight: normal;
        color: #09005D;
        text-decoration: none;
}
.txtbleunuitsmallitalic,A.txtbleunuitsmallitalic {
        font-family: Arial, Helvetica, sans-serif;
        font-size: 10px;
        font-style: italic;
        font-weight: normal;
        color: #09005D;
        text-decoration: none;
}
.txtmauveptit {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 11px;
  font-style: normal;
  font-weight: normal;
  color: #9696D2;
  text-decoration: none;
}
```

Balise <Cadre>

Cette balise active l'encadrement de l'affichage associé aux champs qu'elle contient.

Attributs

Titre

Identifie le texte à afficher pour introduire les champs contenus dans cette balise.
Syntaxe : Titre="*Texte à afficher*"

Html_Titre

Code html à afficher avant le titre.
Défaut : L'attribut « Html_Titre » de la balise <DocumentXML>
Syntaxe : Html_Titre="*code html*".
Les caractères '<' et '>' doivent être remplacés par le caractère 'S'.

Class_Titre

Style à utiliser pour l'affichage du titre.
Défaut : L'attribut « Class_Titre » de la balise <DocumentXML>
Syntaxe : Class_Titre="*classe définie dans une feuille de style*"

Border

Epaisseur du cadre en pixels.
Défaut : L'attribut « Border » de la balise <DocumentXML>
Syntaxe :
Border="*n* " avec n en pixels.
Si n=0 alors il n'y aura pas de cadre visible.

BorderForm

Epaisseur du cadre dans le cas du formulaire (en pixels).
Défaut : L'attribut « BorderForm » de la balise en cours,
puis l'attribut « BorderForm » de la balise <DocumentXML>,
puis l'attribut « Border » de la balise en cours
Syntaxe :
Border="*n* " avec n en pixels.
Si n=0 alors il n'y aura pas de cadre visible.

Traitement

Indicateurs contrôlant le traitement du cadre.
Attention : Les indicateurs spécifiés sans l'attribut « Traitement » de la balise <DocumentXML> seront aussi traités.
Syntaxe : Traitement="indicateur "

Les indicateurs possibles sont :

NoAffInForm	Ne pas afficher le titre dans le formulaire
NoAff	Ne pas afficher le titre dans la fiche

Exemple de balise <Cadre>

```
<Cadre
   Titre="Le projet"
   Border="1"
   Traitement="NoAff"
>
    <Champ1>…</Champ1>
    <Champ2>…</Champ2>
    <Champ3>…</Champ3>
</Cadre>
```

Balise <Titre>

Cette balise affiche un titre.

Attributs

Titre

Identifie le texte à afficher pour introduire les champs contenus dans cette balise.
Syntaxe : Titre=" *Texte à afficher* "

Html_Titre

Code html à afficher avant le titre.
Défaut : L'attribut « Html_Titre » de la balise <DocumentXML>
Syntaxe : Html_Titre=" *code html* ".
Les caractères '<' et '>' doivent être remplacés par le caractère 'S'.

Class_Titre

Style à utiliser pour l'affichage du titre.
Défaut : L'attribut « Class_Titre » de la balise <DocumentXML>
Syntaxe : Class_Titre=" *classe définie dans une feuille de style* "

Traitement

Indicateurs contrôlant le traitement du cadre.
Attention : Les indicateurs spécifiés sans l'attribut « Traitement » de la balise <DocumentXML> seront aussi traités.
Syntaxe : Traitement="indicateur "

Les indicateurs possibles sont :

NoAffInForm	Ne pas afficher le titre dans le formulaire
NoAff	Ne pas afficher le titre dans la fiche

Exemple de balise <Titre>

```
<Titre
   Titre="Le projet"
    Html_Titre="§HR SIZE='1'  WIDTH='80%' /§"
   Traitement="NoAff"
/>
```

Balise <*nom_champ*>

Cette balise, **dont le nom est libre**, introduit les paramètres de traitement d'un champ (attributs).

Attributs communs à tous les types de champs

Label

Libellé du champ.
Syntaxe : Label="*Libellé* "

AlignLabels

Alignement par défaut à utiliser pour le libellé du champ.
Défaut : L'attribut « AlignLabels » de la balise <DocumentXML> , sinon « à droite »
Syntaxe : AlignLabels="*Left*"
AlignLabels="*Right*"

Format

Type de présentation du champ dans le cas de l'affichage de la fiche.
Défaut : L'attribut « Format » de la balise <DocumentXML>
Syntaxe :
Format="Horizontal" : Le libellé du champ et sa valeur sont sur la même ligne
Format="Vertical" : Le libellé du champ est sur une ligne, la valeur est sur une autre ligne

FormatForm

Type de présentation du champ dans le cas de l'affichage du formulaire.
Défaut : L'attribut « FormatForm » de la balise <DocumentXML> ,
puis l'attribut « Format » de la balise en cours
Syntaxe :
FormatForm="Horizontal" : Le libellé du champ, sa valeur et le commentaire associé sont sur la même ligne (cf. tableau html)
FormatForm="Vertical" : Le libellé du champ et le commentaire associé sont sur une ligne, la valeur est sur une autre ligne

Class_Label

Style à utiliser pour l'affichage du libellé du champ
Défaut : L'attribut « Class_Label » de la balise <DocumentXML>
Syntaxe : Class_Label="*classe définie dans une feuille de style*"

Class_Valeur

Style à utiliser pour l'affichage de la valeur du champ
Défaut : L'attribut « Class_Valeur» de la balise <DocumentXML>
Syntaxe : Class_Valeur="*classe définie dans une feuille de style*"

Class_Valeur_Form

Style à utiliser pour l'affichage de la valeur du champ dans le formulaire.
Défaut : L'attribut « Class_Valeur_Form » de la balise <DocumentXML> ,
puis l'attribut "Class_Valeur" de la balise en cours
Syntaxe : Class_Valeur_Form="*classe définie dans une feuille de style*"

Class_Cmt

Style à utiliser pour l'affichage du commentaire associé à la saisie du champ

Syntaxe : Class_Cmt="*classe définie dans une feuille de style*"

TypeInput

Type de dialogue pour la saisie, et par voie de conséquence, type de valeur du champ.
Dans le cas de l'affichage du formulaire, un dialogue dépendant de cet attribut sera généré pour la saisie du champ.
De même, l'affichage du champ dans une fiche dépendra de cet attribut.
Syntaxe :

TypeInput="Texte"	=> La valeur du champ est une chaîne de caractères sur 1 ligne
TypeInput="Textarea"	=> La valeur du champ est une chaîne de caractères sur n lignes
TypeInput="Date"	=> La valeur du champ est une chaîne contenant une date
TypeInput="Select"	=> La valeur du champ est une chaîne choisie parmi plusieurs
TypeInput="SelectM"	=> La valeur du champ est n chaînes choisies parmi plusieurs
TypeInput=" Select+"	=> La valeur du champ est une chaîne choisie parmi plusieurs, ou une chaîne spécifiée par l'utilisateur
TypeInput=" SelectM+"	=> La valeur du champ est n chaînes choisies parmi plusieurs, ou une chaîne spécifiée par l'utilisateur
TypeInput="Radio"	=> La valeur du champ est une chaîne choisie parmi plusieurs
TypeInput="Checkbox"	=> La valeur du champ est n chaînes choisies parmi plusieurs
TypeInput="File"	=> La valeur du champ est le nom d'un fichier à télécharger

InputOblig

Indicateur spécifiant si le renseignement du champ à une valeur non vide, est obligatoire.
Si un champ obligatoire n'est pas renseigné par l'utilisateur, celui-ci sera prévenu et le formulaire ne sera pas transmis.
Défaut : Champ optionnel.
Syntaxe : InputOblig=" *Texte à afficher après le libellé du champ* " => le renseignement du champ est alors obligatoire
InputOblig=" " (espace) => champ obligatoire sans indication
InputOblig="" (chaine vide) => champ optionnel
Exemple : InputOblig=" *" => affichage d'une astérisque après le libellé du champ obligatoire

Traitement

Indicateurs contrôlant le traitement du cadre.
Attention : Les indicateurs spécifiés dans l'attribut « Traitement » de la balise <DocumentXML> seront aussi traités.
Syntaxe : Traitement="indicateur_1 ;indicateur_2 ;… ;indicateur_k"

Les indicateurs possibles sont :

NoAffInForm	Ne pas afficher l'entrée dans le formulaire (=> input "hidden")
NoAff	Ne pas afficher l'entrée dans la fiche
NoAffLabelInForm	Ne pas afficher le libellé du champ dans le formulaire
NoAffLabel	Ne pas afficher le libellé du champ dans la fiche
NoAffVide	Ne pas afficher les lignes (vides) d'un champ vide
InitDate	Initialiser la date par défaut (aaaa-mm-jj hh:mm:ss)
InitDateFr	Initialiser la date par défaut (jj/mm/aaaa)
Url	Forcer l'ouverture du lien associé dans une nouvelle fenêtre
UrlSansNom	Forcer l'ouverture du lien associé dans une nouvelle fenêtre
UrlRubrique	Créer un lien vers une rubrique
MailTo	Créer un lien vers la messagerie

AddHidden

Indicateur activant la création d'un champ « caché » (i-e non visible par l'utilisateur). Ce champ caché et nommé, est alors accessible par du code JavaScript (cf. l' attribut <Evnt_xxx> décrites ci-après).

Syntaxe : AddHidden="*nom du champ caché* "

NB : La syntaxe AddHidden="1" est équivallente à AddHidden="*nom du champ XML*"

Exemple : AddHidden="NbSacs"

ou

```
<NbSacs>
  ……
 AddHidden="1"
  ……
</NbSacs>
```

Evnt_<*event_js*>

Evènement JavaScript à inclure dans la balise html <INPUT> généré pour ce champ.

Syntaxe : Code JavaScript. « event_js » est un événement JavaScript de la balise html <INPUT>.

Exemple :

```
Evnt_onFocus="if( !this.value )  this.value='1' ; // Valeur min "
Evnt_onBlur="if( !this.value || this.value>10) ↵
                    {alert('Nombre de sacs trop important (max=10)'); ↵
                     this.value=10; return false;} "
```

Utilisé avec l'attribut « AddHidden="*nomCaché* " », on trouve souvent le code suivant, permettant de garder accessible la valeur du champ en cours :

```
Evnt_onFocus="this.value=document.<typeFicheXml>.<nomCaché>.value; "
Evnt_onBlur="document.<typeFicheXml>.<nomCaché>.value=this.value; "
```

D'autre champs peuvent alors accéder à la valeur ainsi sauvegardée ; de même, cette valeur peut être traitée avant l'envoi du formulaire (cf. attribut « Evnt_*event_js* » de la balise <DocumentXML>).

NB : Le nom du formulaire html est <*type de fiche XML* >.

Filtre

Filtre à exécuter avant d'afficher la valeur du champ.

Syntaxe : Filtre=" *procédure_1;procédure_2; … ;procédure_k* "

NB : Les procédures doivent être définies dans le fichier de code « mes_fonctions.php3 »

Cmt

Commentaire explicitant la saisie du champ.

Syntaxe : Cmt=" *texte* "

TypoSpip

Indicateur activant l'affichage d'une image supportant un lien vers l'aide en ligne de Spip.

Syntaxe : TypoSpip=" *xxx* "

L'image est fournie par le fichier « **icones/aide*xxx*.gif** ». Le lien est affiché après le commentaire éventuel.

Attributs spécifiques aux champs de TypeInput
"Texte" ou "Textarea"

TypeInput="**Texte**"	Texte sur 1 ligne
TypeInput="**Textarea**"	Texte sur plusieurs lignes

NB : Les raccourcis Spip de mise en page sont exploitables.

NbCaracteres

Taille de la boite de saisie du champ (TypeInput='Texte' ou 'Textarea').
Défaut : L'attribut « DefautNbCars» de la balise <DocumentXML>, ou 83 si format vertical, ou 60 si format horizontal.
Syntaxe : Attribut « SIZE » de la balise html <INPUT>

Attributs spécifiques aux champs de TypeInput
"Radio" ou "Checkbox"

TypeInput="**Radio**"	Choix d'une valeur parmi plusieurs
TypeInput="**Checkbox**"	Choix d'une ou plusieurs valeurs

Labels

Choix possibles pour la valeur du champ.
Syntaxe : Labels="*choix_1;choix_2; … ;choix_k*"

Attributs spécifiques aux champs de TypeInput
"Select" ou " Select+" ou " SelectM" ou " SelectM+"

TypeInput="**Select**" Choix d'une valeur parmi plusieurs
TypeInput="**Select+**" Choix d'une valeur parmi plusieurs, ou création d'un nouveau choix possible.
TypeInput="**SelectM**" Choix d'une ou de plusieurs valeurs.
TypeInput="**SelectM+**" Choix d'une ou de plusieurs valeurs, ou création d'un nouveau choix possible.

DefautTexte

Texte affiché par défaut dans la boite de sélection.
Syntaxe : DefautTexte="*texte*"

Table

Table Spip dont les enregistrements fourniront les choix à propose dans la boite de sélection.
Syntaxe : Table="*spip_xxxxx* "
Table="*spip_xxxxx as yyyy*"
Exemple : Table="*spip_rubriques as rubriques* "

ChId

Nom de la clé primaire à utiliser dans la table.
NB : Sa valeur sera associée à la sélection.
Syntaxe : ChId="*nom de la clé primaire*"

ChpTxt

Nom du champ de la table qui donnera le texte à afficher pour la sélection.
Syntaxe : ChTxt="*nom du champ*"

ChpValOption

Nom du(des) champ(s) de la table qui donneront la valeur associée à la sélection.
NB : La valeur associée la sélection sera
« ‹valeur de ChId›,‹valeurs des champs ChpValOption› »
Syntaxe : ChValOption="*nom_1,nom_2,…*"

TableWhere

Nom de la table utilisée pour l'éventuelle clause « where » de la requête SQL.
Syntaxe : TableWhere="*spip_xxxxx* "
TableWhere="*spip_xxxxx as yyyy*"
TableWhere="*spip_xxxxx , spip_yyyyy as zzzz* "
Exemple : TableWhere="*spip_mots as mots , spip_mots_rubriques as lien_mots* "

ChpWhere

Eventuelle clause « where » de la requête SQL.
Syntaxe : ChpWhere ="*expression logique SQL adhoc*"
Exemple : ChpWhere="*rubriques.titre like %spip%* "

ChpClasst

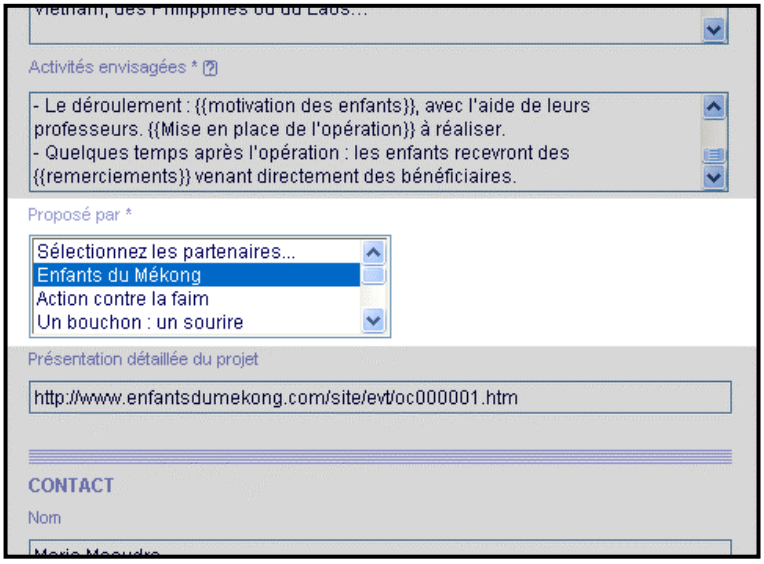
Eventuelle clause « order » de la requête SQL.
Syntaxe : ChpClasst ="*expression SQL adhoc*"

Exemple : ChpClasst="*rubriques.titre , rubriques.date* "

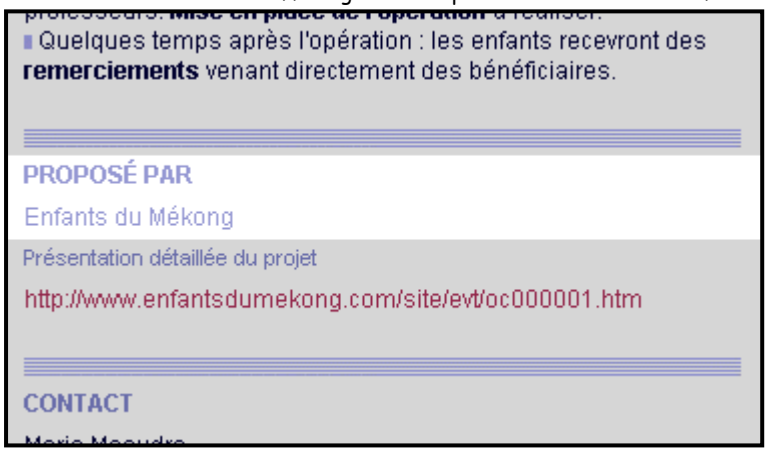
Exemple de balise avec TypeInput="Select"

```
<Partenaires
    Label="Proposé&amp;nbsp;par"
    Traitement="NoAffLabel,UrlRubrique"
    InputOblig=" *"
    TypeInput="SelectM"
    AlignLabels="left"
    DefautTexte="Sélectionnez les partenaires..."
    Table="spip_rubriques AS rubriques"
    ChpId="rubriques.id_rubrique"
    ChpTxt="rubriques.titre"
    ChpValOption="rubriques.titre"
    TableWhere="spip_mots_rubriques AS lien_mot, spip_mots AS mots"
    ChpWhere="rubriques.id_parent!=0 AND rubriques.id_rubrique=lien_mot.id_rubrique AND ↵
        lien_mot.id_mot=mots.id_mot AND mots.titre='Partenaire' AND rubriques.statut='publie' "
    FiltreApresImport="liensProjetPartenaires"
    ></Partenaires>
```

Affichage du champ « Partenaires » dans le formulaire



Affichage du champ « Partenaires » dans la fiche



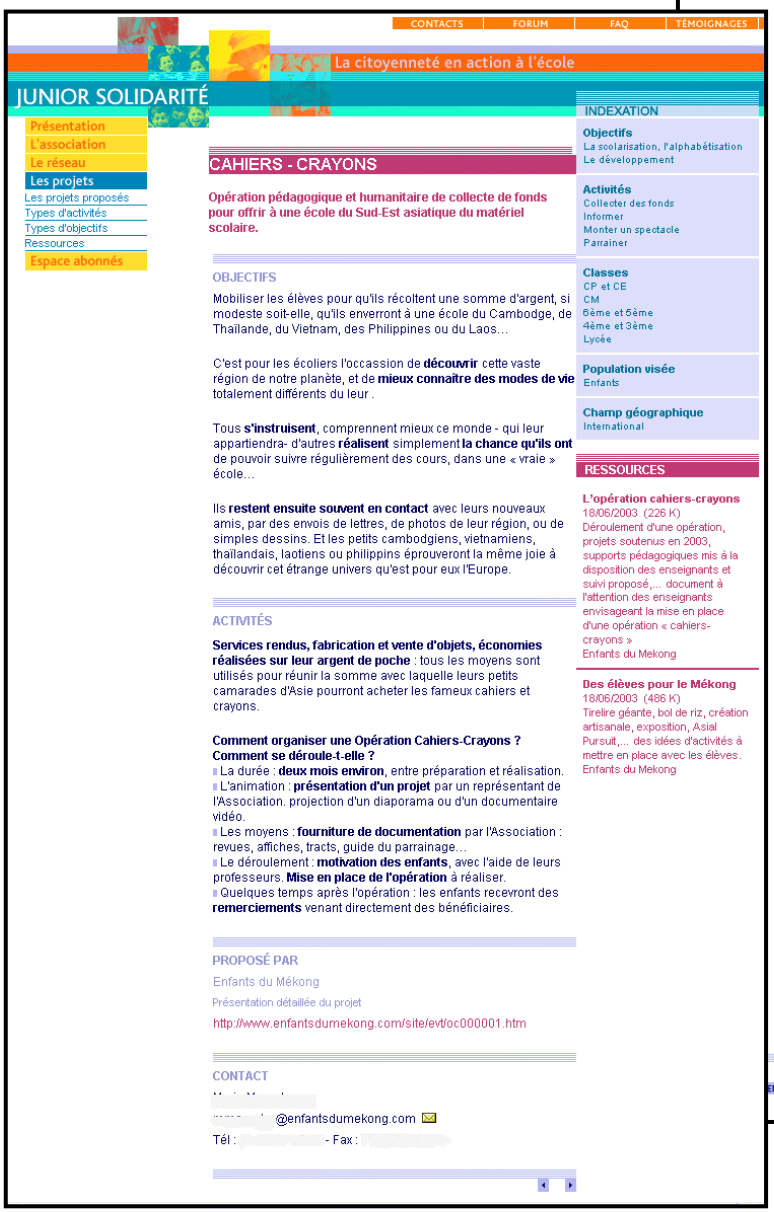
Annexe 2 - Exemple de FICHE XML complète: Description d'un projet de solidarité

(cf. http://juniorsolidarite.org)

A titre d'exemple, nous décrirons ici la fiche descriptive d'un projet de solidarité géré par le site « http://juniorsolidarite.org », site entièrement construit avec Spip.

La description de chaque champ dans le modèle est listée, accompagnée d'une vue du champ dans le formulaire de création/modification d'une fiche « Projet », ainsi que d'une vue du champ tel qu'il se présente lors de l'affichage d'une fiche.

NB : Le modèle de fiche décrit ci-dessous est livré à titre d'exemple dans la distribution du package « SPIP-XML ».



Modèle de fiche « Projet » - Fiche et formulaire associés

```
<DocumentXML
    TypeDocXML="ModeleFicheProjet"
    Handler="ModeleFicheXML"
    Evnt_onSubmit="if(!this.NomRubrique.value ||
     this.NomRubrique.value.substring(0,5)=='Nom d')
     {alert('Nom du projet absent!...');return false;}"
   Class_Titre="h3Spip"
   FormatForm="Vertical"
   Format="Horizontal"
  AlignLabels="left"
  Traitement="NoAffVide"
  Class_Label="txtmauveptit"
  Class_Valeur="txtbleunuit"
  Class_Valeur_Form="txtbleunuit"
  Class_Cmt="txtbleunuitsmallitalic"
  DefautNbCars="40"
  Html_Titre="§BR /§§IMG
     SRC='images/filetmauve.gif' WIDTH='100%'
     HEIGHT='9'§§BR /§"
  Border="0"
  Width="100%"
>
```

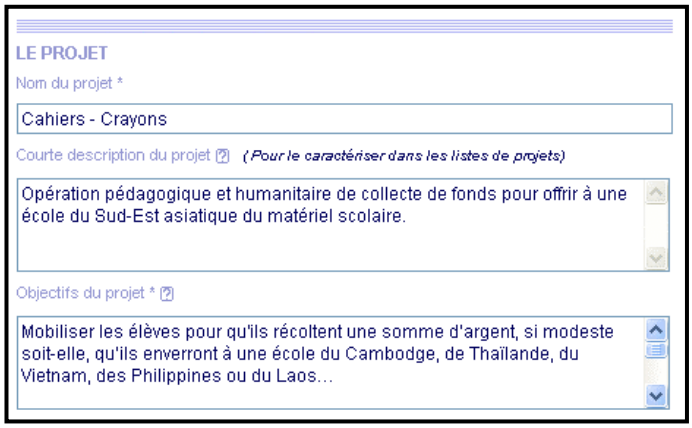
```
<Titre_1_
  Titre="Le projet"
  Traitement="NoAff"
  />
```

```
<NomRubrique
  Label="Nom du projet"
  Traitement="NoAff"
  InputOblig=" *"
  TypeInput="Texte"
  AddHidden="1"
  Filtre="supprimer_numero"
  Evnt_onFocus=
      "if(!this.value || this.value.substring(0,5)=='Nom d')
        this.value='';"
  Evnt_onBlur=
     "document.FicheProjet.NomRubrique.value=this.value;"
  ></NomRubrique>
```

```
<CourteDescription
  Label="Courte description du projet"
  Traitement="NoAff"
  InputOblig=""
  TypeInput="TextArea"
  TypoSpip="Mauve"
  Cmt="Pour le caractériser dans les listes de projets"
  ></CourteDescription>
```

```
<Titre_2_
  Titre="Objectifs"
  Traitement="NoAffInForm"
  Html_Titre="§IMG SRC='images/filetmauve.gif'
                  WIDTH='100%' HEIGHT='9'§§BR /§"
  />
```

```
<ObjectifsProjet
  Label="Objectifs du projet"
  Traitement="NoAffLabel"
  InputOblig=" *"
  TypeInput="TextArea"
  TypoSpip="Mauve"
  ></ObjectifsProjet>
```



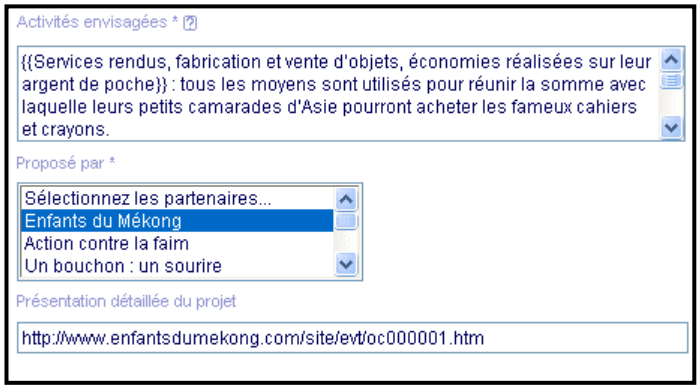
```
<Titre_3_
   Titre="Activités"
   Traitement="NoAffInForm"
   />
```

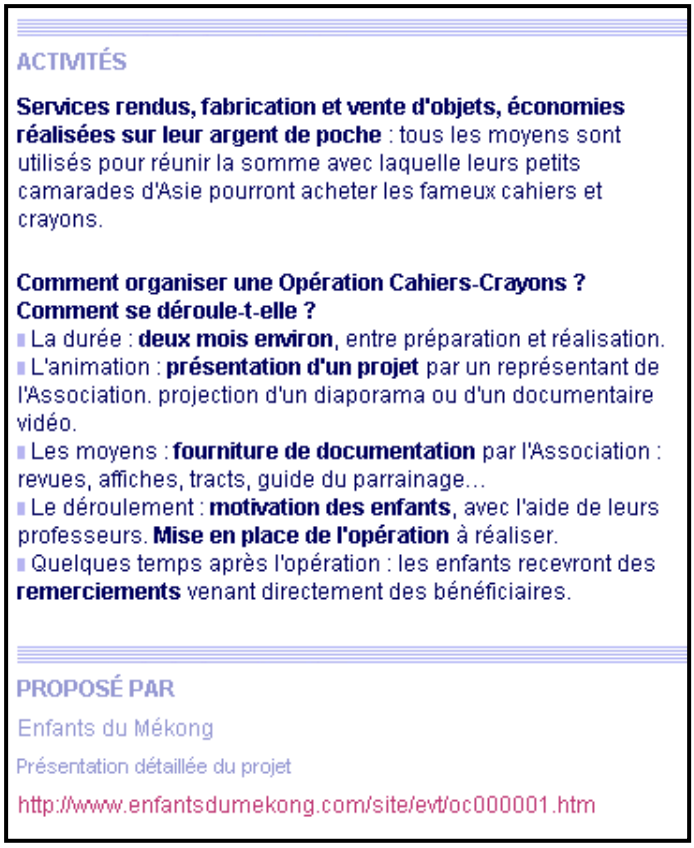
```
   <ActivitesProjet
    Label="Activités envisagées"
    Traitement="NoAffLabel"
    InputOblig=" *"
    TypeInput="TextArea"
    Format="Vertical"
    TypoSpip="Mauve"
    Class_Valeur="txtbleunuit"
   ></ActivitesProjet>
```

```
  <Titre_4_
   Titre="Proposé par"
   Traitement="NoAffInForm"
   />
```

```
   <Partenaires
    Label="Proposé&amp;amp;nbsp;par"
    Traitement="NoAffLabel,UrlRubrique"
    InputOblig=" *"
    TypeInput="SelectM"
    AlignLabels="left"
    DefautTexte="Sélectionnez les partenaires..."
    Table="spip_rubriques AS rubriques"
    ChpId="rubriques.id_rubrique"
    ChpTxt="rubriques.titre"
    ChpValOption="rubriques.titre"
    TableWhere="spip_mots_rubriques AS lien_mot, ↵
                spip_mots AS mots"
    ChpWhere="rubriques.id_parent!=0 ↵
          AND rubriques.id_rubrique=lien_mot.id_rubrique ↵
          AND lien_mot.id_mot=mots.id_mot ↵
          AND mots.titre='Partenaire' ↵
          AND rubriques.statut='publie'"
    FiltreApresImport="liensProjetPartenaires"
   ></Partenaires>
```

```
   <SiteWebProjet
    Label="Présentation détaillée du projet"
    Format="Vertical"
    InputOblig=""
    TypeInput="Texte"
    FiltreFromInput="verifUrl"
    Traitement="UrlSansNom"
    Evnt_onFocus="if(this.value.substring(0,3)=='...') ↵
                  this.value='http://';"
   >... où trouver des informations complémentaires
    </SiteWebProjet>
```





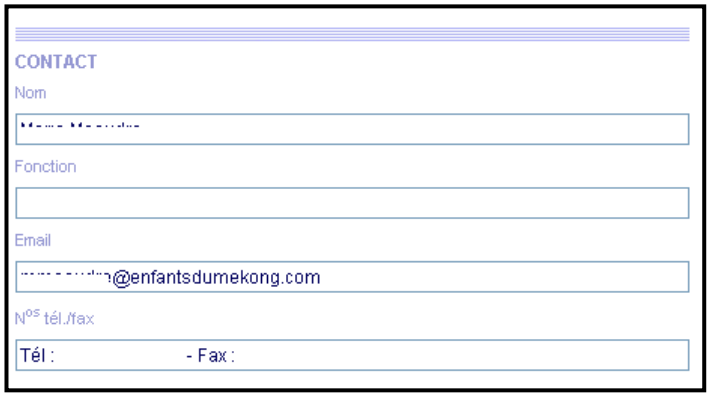
```
<Titre_5_
   Titre="Contact"
   />
```

```
   <NomContact
    Label="Nom"
    Traitement="NoAffLabel"
    InputOblig=""
    TypeInput="Texte"
    Evnt_onFocus="this.value='';"
    >Prénom Nom...
   </NomContact>
```

```
   <FonctionContact
    Label="Fonction"
    Traitement="NoAffLabel"
    InputOblig=""
    TypeInput="Texte"
    ></FonctionContact>
```

```
   <EmailContact
    Label="Email"
    Traitement="NoAffLabel,MailTo"
    InputOblig=""
    TypeInput="Texte"
    ></EmailContact>
```

```
   <TelContact
    Label="N§SUP§os§/SUP§ tél./fax"
    Traitement="NoAffLabel"
    InputOblig=""
    TypeInput="Texte"
    ></TelContact>
```



```
<Titre_6_
   Titre="Indexation (mots-clés)"
   Traitement="NoAff"
   />
   <VsObjectifs
    Label="Objectifs"
    InputOblig=""
    AlignLabels="left"
    TypeInput="SelectM"
    DefautTexte="...autre..."
   Table="spip_appli_index_objectifs"
    ChpId="id_index_objectif"
    ChpTxt="titre"
    ChpValOption="titre"
    Traitement="NoAff"
    ></VsObjectifs>
```

```
   <VsActivites
    Label="Activités"
    InputOblig=""
    AlignLabels="left"
    TypeInput="SelectM"
    DefautTexte="...autre..."
    Table="spip_appli_index_activites"
    ChpId="id_index_activite"
    ChpTxt="titre"
    ChpValOption="titre"
    ChpClasst="classt"
    Traitement="NoAff"
    ></VsActivites>
```

```
   <VsClasses
    Label="Classes"
    InputOblig=""
    AlignLabels="left"
    TypeInput="SelectM"
    DefautTexte="...autre..."
    Table="spip_appli_index_classes"
    ChpId="id_index_classe"
    ChpTxt="titre"
    ChpValOption="titre"
    Traitement="NoAff"
    ></VsClasses>
```

```
   <VsPopulation
    Label="Population visée"
    InputOblig=""
    AlignLabels="left"
    TypeInput="SelectM"
    DefautTexte="...autre..."
    Table="spip_appli_index_populations"
    ChpId="id_index_population"
    ChpTxt="titre"
    ChpValOption="titre"
    Traitement="NoAff"
    ></VsPopulation>
```

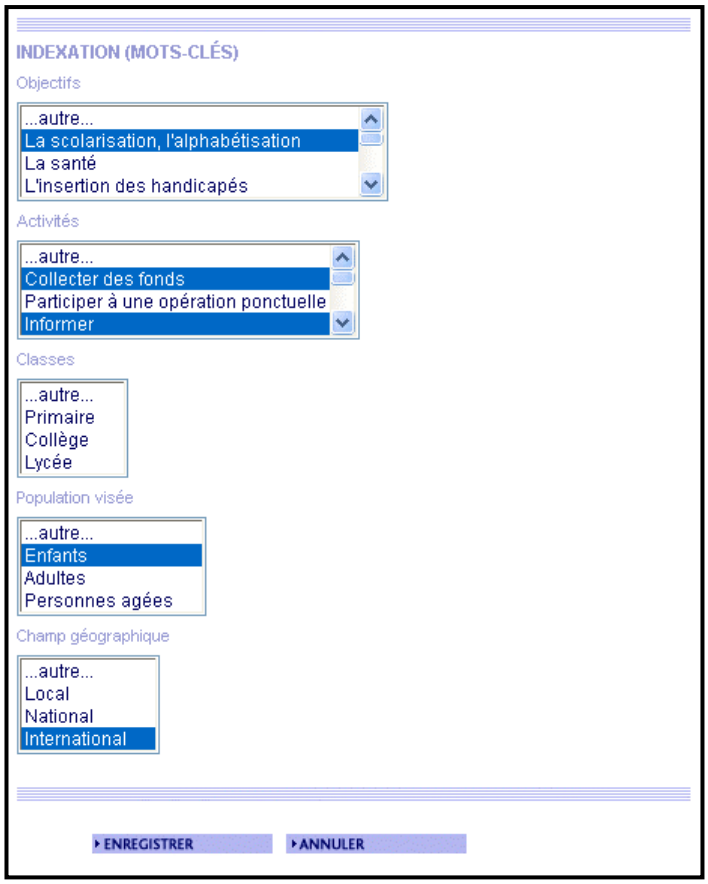


Table MySQL « spip_appli_index_champgeos » (le préfixe des tables Spip est « cc »)

Table *cc_appli_index_champgeos*

←T→		id_index_champgeo	titre
Modifier	Effacer	1	Local
Modifier	Effacer	2	National
Modifier	Effacer	3	International

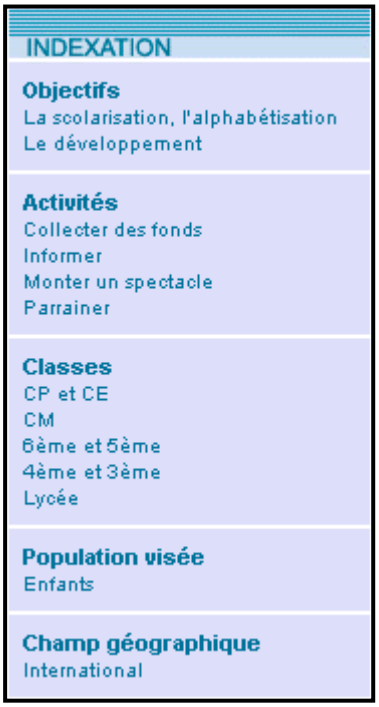
```
  <VsPortee
    Label="Champ géographique"
    InputOblig=""
    AlignLabels="left"
    TypeInput="SelectM"
    DefautTexte="...autre..."
    Table="spip_appli_index_champgeos"
    ChpId="id_index_champgeo"
    ChpTxt="titre"
    ChpValOption="titre"
    Traitement="NoAff"
    ></VsPortee>

 <Titre_99_   Html_Titre="§BR /§§IMG ↵
              SRC='images/filetmauve.gif' WIDTH='100%' ↵
              HEIGHT='9'§ §BR /§ §?=$lienAvAp?§"  />

</DocumentXML>
```

Squelette « rubrique.html » : Affichage des paramètres d'indexation du projet dans un calque séparé

```
.....
.....
<!------------------------- INDEXATION PROJET ------------------------->
<?
// Afficher les paramètres de classement des projets
    if($Entite=='Projet') {
?>
      [(#TEXTE|affIndexation)]
<?
    } // Fin if($Entite=='Projet')
.....
.....
```

Filtre « affIndexation » : Affichage des paramètres d'indexation du projet

```
function affIndexation($data) {

   global $attributsModele;  // Renseigné par l'utilisation du filtre « processXML »

/// Récupérer les valeurs des balises Xml de la fiche
    $valeursFiche = balisesXML($data);
.....
.....
    foreach($valeursFiche as $index=>$value) {

// Ne traiter que les balises "Vs..." non vides
      if(strpos('_'.$index,'Vs')!=1 || !$value) continue;
.....
.....
       $outData .= '
            <td bgcolor="#DDDDF9" class="txtbreves" width="180">
              <span class="txtbrevesbold">'.
                $attributsModele[$index]['Label'].'
              </span><br />
              <small>' ;
       $values = explode(';', $value);
       foreach($values as $value)
         if ($value) {
           $outData .= noNumero($value).'<br/>';
         }

       $outData .= '
              </small>
            </td>
.....
.....
    } // Fin foreach($valeursFiche as $index=>$value)
.....
.....
    return $outData;

 } // Fin function affIndexation($data)
```



Contenu de la fiche « Projet » utilisée pour les images ci-dessus

<**DocumentXML** TypeDocXML="FicheProjet" Handler="FicheXML">

<**NomRubrique**>Cahiers - Crayons</**NomRubrique**>

<**CourteDescription**>Opération pédagogique et humanitaire de collecte de fonds pour offrir à une école du Sud-Est asiatique du matériel scolaire.
§NL§</**CourteDescription**>

<**ObjectifsProjet**>Mobiliser les élèves pour qu'ils récoltent une somme d'argent, si modeste soit-elle, qu'ils enverront à une école du Cambodge, de Thaïlande, du Vietnam, des Philippines ou du Laos…
§NL§
§NL§C'est pour les écoliers l'occasion de {{découvrir}} cette vaste région de notre planète, et de {{mieux connaître des modes de vie}} totalement différents du leur .
§NL§
§NL§Tous {{s'instruisent}}, comprennent mieux ce monde – qui leur appartiendra- d'autres {{réalisent}} simplement {{la chance qu'ils ont}} de pouvoir suivre régulièrement des cours, dans une « vraie » école…
§NL§
§NL§Ils {{restent ensuite souvent en contact}} avec leurs nouveaux amis, par des envois de lettres, de photos de leur région, ou de simples dessins. Et les petits cambodgiens, vietnamiens, thaïlandais, laotiens ou philippins éprouveront la même joie à découvrir cet étrange univers qu'est pour eux l'Europe.</**ObjectifsProjet**>

<**ActivitesProjet**>{{Services rendus, fabrication et vente d'objets, économies réalisées sur leur argent de poche}} : tous les moyens sont utilisés pour réunir la somme avec laquelle leurs petits camarades d'Asie pourront acheter les fameux cahiers et crayons.
§NL§
§NL§{{Comment organiser une Opération Cahiers-Crayons ? Comment se déroule-t-elle ?}}
§NL§- La durée : {{deux mois environ}}, entre préparation et réalisation.
§NL§- L'animation : {{présentation d'un projet}} par un représentant de l'Association. projection d'un diaporama ou d'un documentaire vidéo.
§NL§- Les moyens : {{fourniture de documentation}} par l'Association : revues, affiches, tracts, guide du parrainage…
§NL§- Le déroulement : {{motivation des enfants}}, avec l'aide de leurs professeurs. {{Mise en place de l'opération}} à réaliser.
§NL§- Quelques temps après l'opération : les enfants recevront des {{remerciements}} venant directement des bénéficiaires.</**ActivitesProjet**>

<**Partenaires**>48,Enfants du Mékong</**Partenaires**>

<**SiteWebProjet**>http://www.enfantsdumekong.com/site/evt/oc000001.htm</**SiteWebProjet**>

<**NomContact**> </**NomContact**>

<**FonctionContact**></**FonctionContact**>

<**EmailContact**> @enfantsdumekong.com</**EmailContact**>

<**TelContact**>Tél : - Fax : </**TelContact**>

<**VsObjectifs**>1,La scolarisation, l'alphabétisation;9,Le développement</**VsObjectifs**>

<**VsActivites**>9,Collecter des fonds;2,Informer;3,Monter un spectacle;7,Parrainer</**VsActivites**>

<**VsClasses**>1,CP et CE;2,CM;3,6ème et 5ème;4,4ème et 3ème;5,Lycée</**VsClasses**>

<**VsPopulation**>1,Enfants</**VsPopulation**>

<**VsPortee**>3,International</**VsPortee**>

<editDate>2003-06-12 19:39:41</editDate>
<userName>Nathalie</userName>
<userEmail>nathalie.navarro@juniorsolidarite.org</userEmail>

</**DocumentXML**>