Internationalisation de vos applications SharePoint 2007

Sommaire

Introduction

La plateforme SharePoint, incluant les produits Windows SharePoint Services 3.0 et Microsoft Office SharePoint Server 2007, a largement amélioré l'internationalisation des ressources dans sa dernière version, facilitant ainsi le portage d'applications dans différentes langues. Par exemple, les définitions de site et les fonctionnalités peuvent être créées indépendamment de la langue et les ressources de traduction gérées séparément.

Le but de cet article est de vous présenter les possibilités offertes par SharePoint pour permettre le support multilingue de vos applications, depuis les fichiers de définition XML jusqu'aux développements de composants tels que les WebParts.

SharePoint est une application .NET

Ce simple constat nous conduit au fait suivant : les sites SharePoint étant en ASP.NET 2.0, ils partagent donc les techniques d'internationalisation proposées par le Framework .NET.

Ce système repose sur un ensemble de fichiers de ressources (fichiers XML, un par langue) contenant une liste de couple clé/valeur attribuant à une clé une traduction donnée.

Vous pourrez utiliser ces fichiers de ressources :

- En les déployant dans le répertoire « App_GlobalResources » de votre application web
- En les embarquant (« embedded ») dans les assemblies ou sous forme de fichiers satellites
- Puis par déclaration dans vos pages (ASPX, Master Page) ou par code :
    - <asp:Label ID="Label1" runat="server" Text="<%$ Resources:ResFile,ResKey %>" />
    - string translation = HttpContext.GetGlobalResourceObject("ResFile", "ResKey");
    - stirng translation = Resources.ResFile.ResourceManager.GetString("ResKey");

Je ne rentrerai pas dans les détails, les techniques étant déjà largement documentées, par exemple dans les tutoriaux du site www.asp.net.

Les ressources de traduction de SharePoint

Il existe plusieurs emplacements contenant des fichiers de ressources dans le répertoire de SharePoint, le fameux « 12 Hive » (par défault « C:\Program Files\Common Files\Microsoft Shared\Web server extensions\12 » sur un système d'exploitation anglais). Chaque emplacement induit une utilisation particulière des ressources et donc leur cadre d'application. C'est ce que je vais vous détailler dans ce chapitre.

12\Resources

Ce répertoire contient l'ensemble des fichiers de ressources pouvant être utilisés dans les fichiers de définition de SharePoint comme les définitions de site et les fonctionnalités. Ce sont des fichiers dits « partagés » car disponibles pour toutes les applications et tous les composants.

12\Config\Resources

Ce répertoire contient les fichiers de ressources qui seront recopiés dans le répertoire « App_GlobalResources » des applications web. On y trouve des ressources propres à SharePoint utilisées dans l'interface et les composants, mais il est tout à fait possible d'y ajouter les nôtres.

La recopie s'effectue uniquement de deux manières :

- lors de la création d'une nouvelle application web
- en utilisant la commande « stsadm.exe -o copyappbincontent »

Le principal problème ici c'est la contrainte d'utiliser « stsadm.exe » lorsqu'on désire mettre à jour des applications web existantes. En effet cette commande doit être obligatoirement lancée avec un compte administrateur local et sur chacun des serveurs web frontaux, ce qui est plutôt lourd dans une ferme SharePoint comportant plusieurs serveurs. De plus, il n'existe pas de ciblage de cette

copie : les fichiers de ressources seront ajoutés dans toutes les applications web, pas seulement celles qui en ont besoin.

12\Config\AdminResources

Le fonctionnement de ce répertoire est identique au précédent sauf qu'il recopie les fichiers de ressources uniquement dans le répertoire « App_GlobalResources » de l'application hébergeant le site d'administration centrale.

12\TEMPLATE\Features\[Nom de la feature]\Resources

Les fonctionnalités SharePoint sont constituées d'un répertoire comprenant un fichier descriptif « Feature.xml ». Y sont définis les différents attributs de la fonctionnalité (titre, description, portée, …) mais aussi les fichiers qui la composent. En créant un sous-répertoire « Resources » il est possible de proposer un ensemble de fichiers de ressources qui n'auront alors de portée que la fonctionnalité courante. Ces fichiers devront s'appeler « Resources.<Culture>.resx ».

Quelques informations supplémentaires

Tout d'abord, une différence notable par rapport au fonctionnement standard de la localisation en ASP.Net est à signaler: vous devrez redéfinir toutes les clés de traduction dans votre fichier localisé ! Par exemple, si vous aviez 10 termes dans le fichier de base, vous devrez avoir les 10 dans votre version française. Si ce n'est pas le cas vous obtiendrez le nom de la ressource, par exemple « $Resources:ResFile,ResKey ; ». En effet, une première vérification sera effectuée pour savoir si le fichier existe pour la culture courante, si c'est le cas il cherchera la traduction, qu'elle soit présente ou pas. Si le fichier n'existe pas ce sera alors le fichier par défaut qui sera utilisé.

Enfin, pour qu'une mise à jour des fichiers de ressources soit effective au niveau des sites web, il vous faudra effectuer un redémarrage des services web (« iisreset ») car les fichiers sont mis en cache.

Où et comment utiliser ces ressources

Tous les fichiers de ressources déployées dans le répertoire « App_GlobalResources » vous serviront pour supporter l'internationalisation de vos interfaces et composants web : pages ASPX, pages maîtres, contrôles utilisateurs, WebParts, …

Les autres (le répertoire « 12\Resources » et le répertoire « Resources » des fonctionnalités) permettent de paramétrer les fichiers de définition, mais vous verrez aussi qu'il est possible de les utiliser directement dans vos développements grâce à l'API SharePoint.

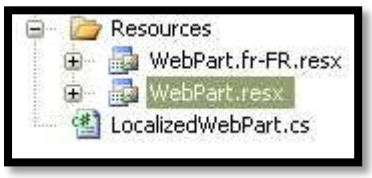
Voici quelques exemples pour illustrer ces différents cas.

WebPart

Avec la WebPart « SharePoint », c'est-à-dire celle disponible dans le modèle objet SharePoint (« Microsoft.SharePoint.WebPartPages.WebPart ») il existe la méthode « LoadResource » permettant de traduire les attributs des propriétés. Cette méthode peut être surchargée afin d'implémenter sa propre gestion des ressources.

Voici le code d'une WebPart très simple dont les propriétés, affichées dans le panneau de configuration de la WebPart, sont traduites *via* l'appel aux ressources embarquées (vous trouverez le code source dans le projet fourni avec l'article).

Dans cet exemple j'ai ajouté mes fichiers de ressources dans le répertoire « Resources » de mon projet Visual Studio :



Vous pouvez remarquer dans les propriétés de ces fichiers qu'ils sont bien marqués comme embarqués (« embedded ») :

Visual Studio crée automatiquement une classe qui nous permet d'utiliser plus facilement les ressources que l'on pourra utiliser dans la méthode LoadResource().

```
public override string LoadResource(string id)
{
    return Resources.WebPart.ResourceManager.GetString(id);
}
```

Petit rappel : seul le fichier de ressource par défaut est inclus dans la DLL principale, les autres génèreront des fichiers satellites (1 DLL par langue) qu'il faudra donc veiller à bien déployer aussi !

Pages Maîtres et pages ASPX

Pour les pages maîtres et les pages ASPX il suffit d'utiliser les ressources situées dans le répertoire « App_GlobalResources » et la déclaration directe au sein de la page. Cette méthode est classique et souvent utilisée dans les sites web ASP.Net.

Voici un fragment de page utilisant ce procédé :

```
<a href="javascript:TopHelpButtonClick('NavBarHelpHome')"
    id="TopHelpLink"
    title="<%$Resources:wss,multipages_helplinkalt_text%>"
    runat="server">
    <img
        border="0" src="/_layouts/images/helpicon.gif"
        alt="<%$Resources:wss,multipages_helplinkalt_text%>"
        runat="server" />
    </a>
```

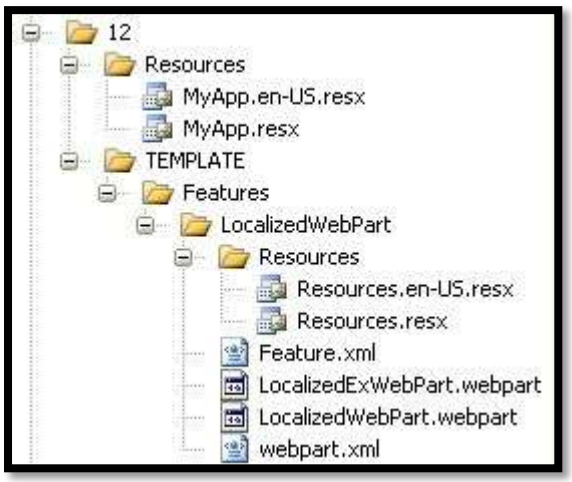
Fonctionnalité

Les fonctionnalités sont un des piliers de la nouvelle version de SharePoint. Celles-ci permettent d'ajouter de nouveaux éléments au sein de SharePoint tels que des workflows, des types de contenu, des liens personnalisés dans l'interface, des événements, des définitions de liste…

Le fichier de définition « Feature.xml » est un document XML contenant la description de la fonctionnalité ainsi que les références à ses éléments (manifests). Comme expliqué auparavant, ce fichier de description utilisera par défaut les fichiers « 12\TEMPLATE\Features\[Nom de la feature]\Resources\Resources.< culture>.resx » (portée limitée à la fonctionnalité uniquement). Il est possible aussi d'utiliser les fichiers du répertoire « 12\Resources » en spécifiant le nom du fichier lors de la référence à la ressource (principe de ressources partagées).

Nous pouvns utiliser nos ressources afin de spécifier le titre et la description d'une fonctionnalité, mais aussi certaines valeurs dans les fichiers manifest. Voici un petit exemple de fonctionnalité installant les fichiers de définition de webparts eux aussi localisés.

Tout d'abord voici la structure des fichiers déployés dans SharePoint :



Le fichier « Feature.xml » a un titre et une description adaptés à la langue du site courant :

```
<?xml version="1.0" encoding="utf-8" ?>
<Feature xmlns="http://schemas.microsoft.com/sharepoint/"
         Id="f4aeb7ad-dcd0-43d3-a91e-ab05ba2bd0ec"
         Title="$Resources:FeatureTitle;"
         Description="$Resources:MyApp,FeatureDescription;"
         Scope="Site"
         Hidden="FALSE"
         RequireResources="TRUE"
         DefaultResourceFile="_Res"
         Creator="Gaetan Bouveret, http://www.sharepointofview.fr/gat/"
         >
  <ElementManifests>
    <ElementManifest Location="webpart.xml"/>
  </ElementManifests>
</Feature>
```

Le titre est récupéré depuis le fichier par défaut (« 12\TEMPLATE\FEATURES\LocalizedWebParts\Resources\Resources.<Culture>.resx ») tandis que la description est récupérée depuis le fichier partagé (« 12\Resources\MyApp.<Culture>.resx »).

Enfin les webparts possèderont un titre et une description eux aussi dépendants de la langue du site courant :

```
<webParts>
  <webPart xmlns="http://schemas.microsoft.com/WebPart/v3">
    <metaData>
      <type name="SPLocalization.LocalizedWebPart, SPLocalization, Version=1.0.0.0, Culture=neutral, Public
      <importErrorMessage>$Resources:MyApp:ImportErrorMessage;</importErrorMessage>
    </metaData>
    <data>
      <properties>
        <property name="Title" type="string">$Resources:MyApp,LocalizedWebPartTitle;</property>
        <property name="Description" type="string">$Resources:MyApp,LocalizedWebPartDescription;</property>
      </properties>
    </data>
  </webPart>
</webParts>
```

Attributs supplémentaires

Dans les fichiers de définition de fonctionnalité le nœud XML « Feature » peut comporter des attributs supplémentaires modifiant leur comportement face aux ressources. Voici leur description.

DefaultResourceFile

L'attribut « DefaultResourceFile » permet de spécifier un fichier par défaut situé dans le répertoire « 12\Resources ». Si l'attribut est manquant ou a pour valeur « _Res » ce seront les fichiers de ressources de la fonctionnalité qui seront utilisés (comportement par défaut décrit juste avant).

Ici un exemple de définition avec les fichiers de ressources « MyApp » (MyApp.resx, MyApp.fr-FR.resx, …) présents dans « 12\Resources » :

```
<Feature xmlns="http://schemas.microsoft.com/sharepoint/"
         Id="f4aeb7ad-dcd0-43d3-a91e-ab05ba2bd0ec"
         Title="$Resources:FeatureTitle;"
         Description="$Resources:FeatureDescription;"
         Scope="Site"
         DefaultResourceFile="MyApp"
         >
</Feature>
```

RequireResources

L'attribut « RequireResources » permet d'avoir une dépendance entre une fonctionnalité et des fichiers de ressources. Ainsi la fonctionnalité sera masquée dans l'interface web lorsque les fichiers de ressources ne sont pas présent si la valeur de l'attribut « RequireResources » est à « TRUE ».

```
<?xml version="1.0" encoding="utf-8" ?>
<Feature xmlns="http://schemas.microsoft.com/sharepoint/"
         Id="f4aeb7ad-dcd0-43d3-a91e-ab05ba2bd0ec"
         Title="$Resources:FeatureTitle;"
         Description="$Resources:FeatureDescription;"
         Scope="Site"
         RequireResources="TRUE">
</Feature>
```

Notez que bien que la fonctionnalité peut se retrouver masquée, celle-ci sera toujours activable *via* stsadm.exe ou le modèle objet : elle ne sera invisible que depuis l'interface web (similaire à l'attribut « Hidden »). De plus cette dépendance ne fonctionne que pour les fichiers de ressources de la fonctionnalité pour la culture désirée. Concrètement, pour un site en français, le fichier « [MaFeature]\Resources**Resources.fr-fr.resx** » sera recherché : s'il existe la fonctionnalité sera affichée, dans le cas contraire elle restera masquée. De fait, le fichier de ressource par défaut (celui sans culture) ne sera pas utilisé.

Utilisation de l'API SharePoint

Le modèle objet SharePoint permet la récupération des valeurs des fichiers de ressources situés dans le répertoire « 12\Resources » grâce à la classe SPUtility située dans l'espace de nom « Microsoft.SharePoint.Utilities » et sa méthode « GetLocalizedString »:

C#

```
public static string GetLocalizedString (
        string source,
        string defaultResourceFile,
        uint language
)
```

Ses paramètres sont :

- *source* : la clé du terme localisé sous la forme « $Resources:[ResFileName,]ResKey », « ResFileName » étant facultatif
- *defaultResourceFile* : le fichier de ressource par défaut à utiliser si le fichier n'est pas spécifié, c'est-à-dire sous la forme « $Resources :ResKey »

- *language* : code culture utilisé pour la traduction (LCID). Si ce code est égal à « 0 » alors ce seront les ressources par défaut qui seront utilisées (fichier « MyApp.resx » par exemple)

Vous pouvez ainsi appeler directement les fichiers de ressources situés dans le répertoire « 12\Resources » sans pour autant nécessiter leur déploiement dans les répertoires « App_GlobalResources » des applications web et surtout localiser plus facilement les applications s'exécutant en dehors du contexte web.

Voici un petit exemple d'utilisation en reprenant la WebPart « LocalizedWebPart » et en changeant la méthode « LoadResource »

```
public override string LoadResource(string id)
{
    string res = id;
    // Need at least medium trust or sharepoint permissions
    string resKey = "$Resources:" + id;
    res = SPUtility.GetLocalizedString(resKey, "WebPart", SPContext.Current.Web.Language);
    if (resKey == res) res = SPUtility.GetLocalizedString(resKey, "WebPart", 0);

    return res;
}
```

Le déploiement des fichiers de ressources

Les fichiers de ressources peuvent être packagés au sein de solutions SharePoint de deux manières :

- Soit en les ajoutant directement dans la solution contenant le reste de vos développements. Ils seront alors déployés en même temps que les autres fichiers et cela permettra de proposer le support d'un certain nombre de langues sans nécessiter d'autre installation.
- Soit en créant un pack de langue. Ce pack est une solution classique SharePoint mais ne contenant que des fichiers propres à une langue et elle est rattachée à une autre solution. On peut y placer les fichiers de ressources mais aussi des images, fichiers javascript, feuilles de styles etc… nécessaires au bon fonctionnement pour la langue supportée. Cette solution devra porter le même nom et utiliser le même identifiant unique que le fichier de la solution « parente » et être installée grâce à la commande suivante :
  stsadm –o addsolution –filename [nom de la solution.wsp] –lcid [Code langue]

Le code de localisation ou lcid (Locale ID) doit correspondre à un pack de langue SharePoint déjà installé. Voici quelques exemples de codes :

- 1033 : version anglaise
- 1036 : version française
- 1031 : version allemande

Vous trouverez ici la liste complète des codes disponibles :
http://www.microsoft.com/globaldev/reference/lcid-all.mspx

Voici un petit exemple de pack de langue pour la solution « SPLocalization.wsp » comprenant des fichiers de ressources globaux ainsi que qu'une fonctionnalité « LocalizedWebPart ». Pour réaliser le pack de langue il nous faut lister tous les fichiers qui seront utilisés pour une liste donnée.

- *MyApp.fr-FR.resx* doit être déployé dans le répertoire « 12\Resources » pour être utilisé par différentes définitions
- *resources.fr-FR.resx* qui est une ressource pour la feature « LocalizedWebPart » et devant être déployé dans « 12\TEMPLATE\Features\LocalizedWebPart\Resources »

Voici à quoi va ressembler le fichier « manifest.xml » qui déploiera les fichiers de ressources:

```
<?xml version="1.0" encoding="utf-8" ?>
- <Solution xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" SolutionId="ad5c7320-c267-43e5-8236-
    e58bb37c9364" xmlns="http://schemas.microsoft.com/sharepoint/">
  - <RootFiles>
      <RootFile Location="Resources\MyApp.fr-FR.resx" />
    </RootFiles>
  - <Resources>
      <Resource Location="LocalizedWebPart\Resources\Resources.fr-FR.resx" />
    </Resources>
  </Solution>
```

Les fichiers de ressources propres à la fonctionnalité sont à déclarer grâce aux nœuds « <Resource /> » en spécifiant le chemin depuis le répertoire « 12\TEMPLATE\Features », les autres sont à déclarer grâce aux nœuds « <RootFile /> » avec un chemin relatif au répertoire « 12 ».

Concernant le fichier Diamond Directive File qui va nous permettre de générer le fichier WSP grâce à l'utilitaire « makecab », voici son contenu :

```
1  .OPTION EXPLICIT
2  ;
3  .Set CabinetNameTemplate=SPLocalization.wsp
4  .set DiskDirectoryTemplate=CDROM
5  .Set CompressionType=MSZIP
6  .Set Cabinet=on
7  .Set Compress=on
8  .Set DiskDirectory1=.
9  .Set CabinetFileCountThreshold=0
10 .Set FolderFileCountThreshold=0
11 .Set FolderSizeThreshold=0
12 .Set MaxCabinetSize=0
13 .Set MaxDiskFileCount=0
14 .Set MaxDiskSize=0
15 "Resources\MyApp.fr-FR.resx" "Resources\MyApp.fr-FR.resx"
16 "LocalizedWebPart\Resources\Resources.fr-FR.resx" "LocalizedWebPart\Resources\Resources.fr-FR.resx"
17 manifest.xml
```

Attention de ne pas écraser le fichier de solution original lors de la génération des packs de langues puisque les solutions porteront le même nom. Et si vous n'utilisez pas le même identifiant interne, un message d'erreur vous indiquera qu'une solution existe déjà :

```
M:\fr>stsadm -o addsolution -filename SPLocalization.wsp -lcid 1036

An object of the type Microsoft.SharePoint.Administration.SPSolution
calization.wsp" already exists under the parent Microsoft.SharePoint
ion.SPFarm named "SharePoint_Config".  Rename your object or delete
 object.
SPLocalization.wsp: The Solution installation failed.
```

Pour finir, il ne faudra pas oublier de spécifier le code culture (LCID) lors de l'ajout. Ici par exemple pour la culture 1036 correspond au français de France (fr-FR) :

stsadm –o addsolution –filename SPLocalization.wsp –lcid 1036

Vous obtiendrez alors le pack de langue dans l'administration que vous pourrez déployer :

Conclusion

La localisation de ses applications est souvent une problématique qui nous semble secondaire lors du développement, mais c'est une problématique très importante dans un contexte international.

C'est aussi de manière générale un bon réflexe de sortir ses ressources de traduction de son code afin de rendre vos développements disponibles dans une nouvelle langue sans effort à un moindre coût.

Vous aurez abordé au travers de cet article les points principaux qui vous permettront de mettre en place cette internationalisation au sein de vos projets.

Références supplémentaires

- Localizing a solution : http://msdn.microsoft.com/en-us/library/aa544282.aspx
- Feature Element (schema XML) : http://msdn.microsoft.com/en-us/library/ms436075.aspx
- WebPart.LoadResource Method : http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.webpartpages.webpart.loadresource.aspx
- ASP.NET étape par étape - Tome 5: Internationalisation d'une application ASP.NET : http://ditch.developpez.com/aspnet/tome5/