9

Installation d'une application .NET

9. Installation d'une application .NET

Lorsqu'une application ou une solution logicielle est conçue à l'aide de *Visual Studio.NET* , celle-ci doit être installée suivant certaines règles afin qu'elle puisse s'exécuter adéquatement sur le poste client.

- Le poste client doit héberger la plate-forme .NET ;
- Les assemblages privés doivent être copiés aux endroits appropriés ;
- Les assemblages publics doivent être enregistrés au sein du *GAC*.

De plus, il peut être sympathique à l'utilisateur que des raccourcis soient créés sur le bureau ou au sein de son menu `Démarrer → Programmes`. Ainsi, une application ou une solution logicielle peut être installée sur le système du client à l'aide de plusieurs techniques.

Gestion des assemblages et du GAC

Un assemblage peut aisément référencer un second assemblage. Ainsi, une application Windows *(EXE)* peut référencer et utiliser les modèles d'objets exposés par une bibliothèque de classes *(DLL)*. Du fait, la plateforme .NET prévoit des indicateurs de portée *(protected et protected friend)* permettant à des classes déclarées au sein d'assemblages d'hériter de classes de base déclarées au sein de autres assemblages. Lors de leur exécution, cependant, les assemblages référencés par un autre assemblage doivent pouvoir être physiquement localisés par ce dernier avant de pouvoir être utilisés.

Référencement d'un assemblage

Lorsque vous ajoutez une référence au sein de votre projet *Visual Studio.NET*, l'assemblage concerné peut être localisé à partir du `GAC` ou à partir d'un endroit quelconque sur le disque dur. Le `GAC` *(Global Assembly Cache)* est un endroit virtuel au sein duquel sont stockés divers assemblages aisément localisables par l'ensemble des applications *.NET*. Le `GAC` est un répertoire au sein duquel sont enregistrés des références sur des assemblages peu importe leur emplacement sur le disque. Généralement, les assemblages inscrits au sein du `GAC` sont ceux qui doivent être réutilisés par différentes applications. N'inscrivez-pas au sein du `GAC` un assemblage qui sera référencé par une seule application au risque de surcharger inutilement ce répertoire central d'assemblages.

Lorsqu'un assemblage référence un second assemblage absent du `GAC`, il débute la recherche en examinant dans le répertoire au sein duquel lui-même se trouve. Ainsi, une technique simple permettant à un assemblage d'être utilisé et d'être reconnu par un exécutable est de se trouver dans le même dossier que celui-ci.

Si l'assemblage ne peut être localisé au sein de ce répertoire, l'assemblage recherche ensuite au sein d'un sous-dossier qui porterait le même nom que l'assemblage recherché. Ainsi, si l'assemblage recherché se nomme `Librairie.dll`, l'assemblage sera recherché au sein du sous-dossier nommé `Librairie`.

Notez que le processus de localisation d'un assemblage fait partiellement abstraction de l'extension du fichier. Celui-ci peut en effet porter l'extension .DLL ou .EXE sans nuire à son référencement.

Attribution d'un nom fort à un assemblage

Avant d'être enregistré au sein du GAC dans le but de le rendre aisément localisable par les autres applications *.NET*, un assemblage doit posséder un nom fort *(Strong Name)*. Un nom fort permet d'assurer :

- l'**unicité** de l'assemblage en y adjoignant un numéro d'identification unique (GUID) codé sur 128-bits ainsi que l'ensemble des informations concernant l'espace de nom et la version de cet assemblage. Ainsi, plusieurs assemblages du même nom pourront aisément être différenciés par la plate-forme *.NET* et ce, même s'ils sont stockés au sein de fichiers du même nom. De plus, diverses versions du même assemblage pourront également être différenciés afin d'empêcher une application d'utiliser une version antérieure ou ultérieure du composant pour laquelle elle a été originalement bâtie.

- l'**authenticité** d'un assemblage en y adjoignant une signature numérique *Authenticode* permettant d'éviter toute altération illicite de son code. Une paire de clé d'encryption est intégrée au sein de l'assemblage afin que toute recompilation nécessite cette même paire de clé. Si une application tente d'utiliser un assemblage qui a été altéré et qui ne possède plus la même signature que lors de la compilation, l'exécution de l'application s'interrompra pour des raisons de sécurité.

Pour **générer un nom fort** pour un assemblage, démarrez l'utilitaire SN.EXE à partir de l'invite de commande de *Visual Studio.NET* afin de créer d'abord une paire de clés pour l'assemblage. Vous trouverez l'invite de commande au sein du menu Démarrer → Programmes → Microsoft Visual Studio.NET → Visual Studio.NET Tools en lui précisant la ligne de commande suivante :

```
sn.exe  /k  c:\fichierCles.snk
```

Dans l'exemple précédent, la paire de clés sera créée au sein du fichier *c:\fichierCles.snk*. Notez que l'extension du fichier n'a pas d'importance et que l'extension **.snk* n'est utilisée que par standard. Une fois la paire de clés générée, ne perdez pas le fichier qui les contient puisqu'il est unique et que seul ce dernier vous permettra de compiler à nouveau l'assemblage sans que de conflits de version ne surviennent avec les applications existantes utilisant déjà cet assemblage. Il peut donc s'avérer sage de stocker ce fichier avec la balance du code source du projet.

Une fois la paire de clés générée, vous devez lier l'assemblage au fichier qui les contient afin qu'il soit utilisé lors de la compilation du projet pour signer numériquement le code. Pour ce faire, il suffit de modifier le fichier AssemblyInfo.vb du projet en y ajoutant l'attribut AssemblyKeyFile et le chemin complet du fichier contenant les clés :

```
<Assembly: AssemblyTitle("")>
<Assembly: AssemblyDescription("")>
<Assembly: AssemblyCompany("")>
<Assembly: AssemblyProduct("")>
<Assembly: AssemblyCopyright("")>
<Assembly: AssemblyTrademark("")>
<Assembly: CLSCompliant(True)>
<Assembly: AssemblyKeyFile("c:\fichierCles.snk")>
```

Désormais, les clés seront utilisées pour signer l'assemblage lorsque celui-ci sera compilé.

Enregistrement d'un assemblage au sein du GAC

Le fait qu'un assemblage soit compilé avec un nom fort en assure l'unicité et l'authenticité. Tout fichier doit ainsi se voir attribuer un nom fort avant de pouvoir être enregistré au sein du GAC *(Globally Assembly Cache)*. L'endroit où se trouve le fichier qui sera enregistré au sein du GAC n'importe pas. Ainsi, le GAC peut référencer des fichiers éparpillés au sein de plusieurs répertoires. Cependant, il peut être préférable de déposer votre assemblage au sein du répertoire *%windir%*\Microsoft.NET\Framework\v1.#.####\ si vous désirez pouvoir y ajouter aisément une référence au sein de *Visual Studio.NET*. Notez que c'est au sein de ce répertoire que se trouvent les assemblages de la plate-forme *.NET*.

Pour **enregistrer un assemblage au sein du GAC**, démarrez l'utilitaire GacUtil.exe à partir de l'invite de commande de *Visual Studio.NET*. Vous trouverez l'invite de commande au sein du menu Démarrer → Programmes → Microsoft Visual Studio.NET → Visual Studio.NET Tools. Précisez-y la ligne de commande suivante :

```
gacutil.exe  /i  c:\dossier\assemblage.dll
```

Dans l'exemple précédent, les composants du fichier assemblage.dll situé dans le dossier c:\dossier\ seront enregistrés au sein du GAC et seront désormais aisément accessibles pour l'ensemble des applications *.NET*.

L'utilitaire GacUtil.exe prend en charge les arguments de ligne de commande suivants :

Argument	Description
/i	Installe au sein du GAC le fichier précisé en paramètre.
/if	Installe au sein du GAC le fichier précisé en paramètre et force son enregistrement si une inscription du même fichier existe déjà.
/ir	
/u	Désinstalle du GAC l'assemblage précisé en paramètre. Ne précisez pas le chemin complet du fichier mais plutôt le nom de l'assemblage à désinstaller. Exemple : gacutil.exe /u monAssemblage Puisque ce procédé désinstalle l'ensemble des versions de cet assemblage, il peut être nécessaire de préciser la version de l'assemblage que l'on désire désinstaller. Exemple : /u monAssemblage, Version=1.1.0.0, Culture=fr
/uf	Désinstalle du GAC l'assemblage précisé en paramètre. La désinstallation de l'assemblage est forcé même si l'assemblage est référencé par une application installée. Exemple : gacutil.exe /uf monAssemblage Il est également possible de préciser la version ou la culture.
/ur	
/l	Liste le contenu du GAC.
/lr	
/cdl	Supprime le contenu de la cache de téléchargement.
/ldl	Liste le contenu de la cache de téléchargement.
/nologo	N'affiche pas l'entête de l'utilitaire GacUtil.exe.
/silent	Supprime toute forme d'affichage provoqué par l'utilitaire GacUtil.exe.

Redirection d'assemblages à l'aide d'un fichier *.config*

L'ensemble des applications .NET fait une grande utilisation des fichiers de configuration. Les fichiers de configuration sont des fichiers texte au format XML qu'il est possible de modifier à l'aide de l'environnement de développement *Visual Studio.NET* ou simplement à l'aide d'un éditeur de texte comme *Bloc-Notes*. Il existe une pluralité de fichiers de configuration *(machine.config, web.config, etc.)* qui possèdent tous la même utilité : configurer différents paramètres d'une ou de plusieurs applications. Un fichier de configuration permet entres autres la redirection d'assemblages fort utile pour mettre à jour une application.

Voici la structure minimale d'un fichier de configuration pour une application Windows :

```
<configuration>
   <runtime>

      <!-- Insérez les diverses configurations ici  -->

   </runtime>
</configuration>
```

Le fichier de configuration d'une application doit se situer dans le même répertoire que l'application. Le nom du fichier de configuration est le nom de l'application assorti d'une extension `.config`. Par exemple, une application appelée `monApp.exe` peut être associée à un fichier de configuration appelé `monApp.exe.config`.

Étude de cas

Disons que vous concevez une application dont l'assemblage principal se nomme `monApplication.exe` et que ce dernier référence un assemblage nommé `monAssembly.dll` compilé à l'aide de l'identificateur de version 1.0.0.0. Peu après avoir déployé avec succès votre application sur l'ensemble des postes du parc informatique de votre entreprise en incluant les deux assemblages précédemment nommés, un utilisateur vous contacte afin de vous mentionner que des erreurs se glissent lors d'un traitement spécifique. Vous examinez le code source de votre application, repérez l'erreur au sein de l'assemblage satellite nommé `monAssembly.dll` et la corrigez. Maintenant, vous devez mettre à jour votre application sur l'ensemble des postes du parc informatique.

Quoique vous pourriez désinstaller l'application sur l'ensemble des postes afin d'en installer la nouvelle version, vous pourriez plus simplement remplacer l'assemblage `monAssembly.dll` existant par le nouveau possédant le code corrigé. Quoique cette technique semble plus simple, un problème majeur se pose : l'assemblage principal `monApplication.exe` rejettera systématiquement le nouvel assemblage satellite puisque lors de sa compilation, `monApplication.exe` a pris en note les informations de l'assemblage satellite référencé et puisque le nouvel assemblage corrigé n'expose pas les mêmes informations d'unicité et ce, même s'il possède le même nom. Or vous devez spécifier à l'assemblage principal d'utiliser le nouvel assemblage satellite corrigé sans devoir recompiler `monApplication.exe`. Vous devez donc rediriger sa référence pour un assemblage vers un nouvel assemblage.

C'est à ce moment que le fichier de configuration vient à la rescousse.

Pour rediriger un assemblage, il suffit de créer un fichier de configuration pour l'assemblage en créant un fichier texte possédant le même nom que l'assemblage principal suivi de l'extension *.config*. Dans l'exemple précédent, il suffirait de créer le fichier `monApplication.exe.config`. Le fichier de configuration doit être créé au sein du même répertoire que l'application visée.

Le fichier de configuration devra ensuite posséder l'élément `<dependentAssembly>` permettant d'identifier un assemblage référencé par l'application. Le fichier de configuration possédera autant de blocs `<dependentAssembly>` que d'assemblages qui doivent être identifiés afin d'être configurés :

```
<configuration>
   <runtime>
      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

       <dependentAssembly>
         <assemblyIdentity name="monAssembly"
                           publicKeyToken="32ab4ba45e0a69a1"
                           culture="fr" />
       </dependentAssembly>

      </assemblyBinding>
   </runtime>
</configuration>
```

Une fois l'assemblage identifié à l'aide de l'élément `<assemblyIdentity>`, il est possible d'ajouter un élément `<bindingRedirect>` permettant de rediriger la référence vers un nouvel assemblage. Le nouvel assemblage doit différencier du précédent par son identificateur de version.

```
<configuration>
   <runtime>
      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

       <dependentAssembly>
         <assemblyIdentity name="monAssembly"
                           publicKeyToken="32ab4ba45e0a69a1"
                           culture="fr" />
         <bindingRedirect oldVersion="1.0.0.0
                          newVersion="2.0.0.0"/>
       </dependentAssembly>

      </assemblyBinding>
   </runtime>
</configuration>
```

A partir de ce moment, l'application ne cherchera plus à établir une référence sur la version 1.0.0.0 de l'assemblage nommé `monAssembly` mais tentera plutôt d'établir une référence sur la version 2.0.0.0.

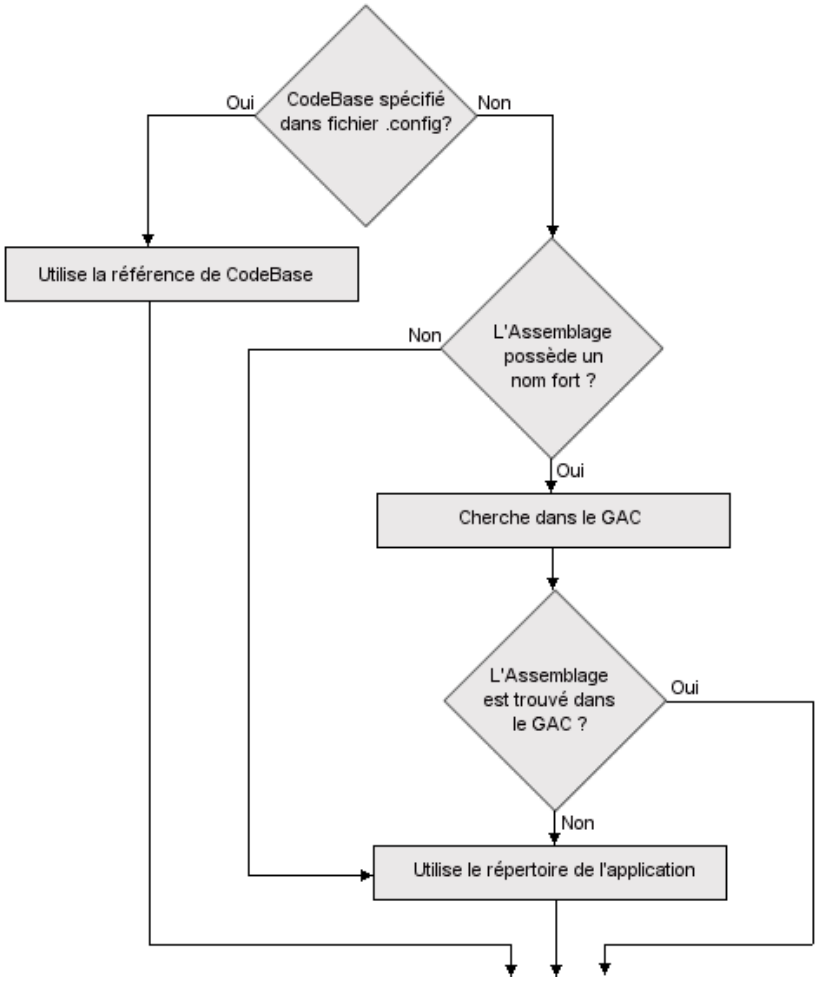
Voici la liste et description de l'ensemble des éléments pouvant figurer dans un fichier de configuration pour une application Windows :

Élément	Description
`<assemblyBinding>`	Contient des informations sur la redirection des versions des assemblages et sur l'emplacement de ces derniers.
`<assemblyIdentity>`	Contient des informations d'identification d'un assemblage.
`<bindingRedirect>`	Redirige une version d'un assemblage vers une autre.
`<codeBase>`	Spécifie où le *runtime* peut trouver un assemblage.
`<dependentAssembly>`	Encapsule la stratégie de liaisons et l'emplacement de chaque assemblage.
`<developmentMode>`	Spécifie si le *runtime* recherche les assemblages dans les répertoires spécifiés par la variable d'environnement `DEVPATH`.
`<gcConcurrent>`	Spécifie si le *runtime* exécute l'opération garbage collection.
`<probing>`	Spécifie les sous-répertoires dans lesquels le *runtime* effectue sa recherche lorsqu'il charge les assemblages
`<publisherPolicy>`	Spécifie si le *runtime* applique la stratégie de l'éditeur.
`<qualifyAssembly>`	Spécifie le nom complet de l'assemblage qui doit être chargé dynamiquement lorsqu'un nom partiel est utilisé.
`<runtime>`	Contient des informations sur les liaisons d'assemblage et sur le comportement de l'opération garbage collection.

L'exemple suivant montre comment spécifier les sous-répertoires de base de l'application dans lesquels le *runtime* devra rechercher les assemblages à l'aide de l'élément `<probing>`. On se rappellera que par défaut les assemblages sont recherchés au sein du répertoire même de l'application ou au sein du GAC.

```
<configuration>
   <runtime>
      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
         <probing privatePath="bin;bin2\subbin;bin3"/>
      </assemblyBinding>
   </runtime>
</configuration>
```

Finalement, une application déterminera le fichier précis de l'Assemblage à utiliser selon plusieurs critères dont la logique de décision est illustrée ci-dessous. Il ne faut toutefois jamais oublié que l'identificateur de version et la signature de l'Assemblage le cas échéant prédomine sur le chemin au sein duquel l'Assemblage peut être trouvé. Ainsi, même si le fichier peut être localisé, il sera rejeté si la version ou la signature ne correspondent pas à moins, évidemment, que le fichier de configuration de l'application n'en prévoit le contraire.



Ainsi, l'Assemblage référencé est d'abord localisé à l'aide de l'attribut `<codeBase>` si existant au sein du fichier de configuration, ensuite au sein du `GAC` puis finalement au sein du répertoire de l'application ainsi que l'ensemble des répertoires référencés par l'attribut `<probing>`.

Installation à l'aide d'un fichier de traitement en lot

Il est possible de procéder à la copie de l'ensemble des fichiers nécessaires pour l'exécution d'une solution logicielle à l'aide d'un fichier de traitement en lot *(Batch file)*. Quoique cette technique ne soit pas la plus conviviale pour permettre à l'utilisateur d'installer l'application, elle peut souvent dépanner lors de l'installation de fichiers complémentaires mais, de surcroît, permet dans un cadre pédagogique de bien comprendre l'ensemble des opérations nécessaires à l'installation d'une solution logicielle.

Un logiciel d'installation *(setup)* doit opérer deux tâches principales :

- **Copier les fichiers à l'endroit correspondant**. Généralement, les fichiers principaux d'une application *EXE* seront copiés dans un répertoire portant le nom de l'application créé dans `Program Files`. Les assemblages *DLL* privés à l'application seront également copiés au sein de ce répertoire. Les assemblages *DLL* publics nécessitant un enregistrement au sein du `GAC` devraient quant à eux être copiés au sein du répertoire `%windir%\Microsoft.NET\ Framework\v1.x.xxxx` afin d'être aisément accessible par les autres applications .NET.

- **Enregistrer les assemblages nécessaires au sein du `GAC`**. L'utilitaire `gacutil.exe` devra être lancé sur les assemblages nécessitant un enregistrement au sein du `GAC`.

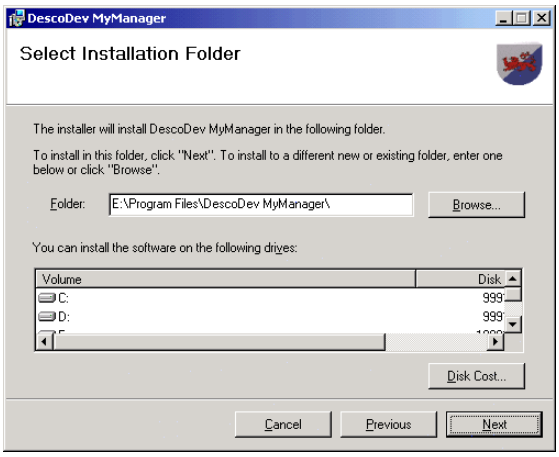
Prenons par exemple une application nommée `monApp` nécessitant trois assemblages : `monApp.exe`, `maLibrairie.dll` et `maLibGac.dll` qui, cette dernière, doit être enregistrée au sein du `GAC`. Déposez ces fichiers à racine du lecteur `c:\` puis créez un fichier de traitement en lot en créant un nouveau fichier texte et renommez-le `installation.bat`. Assurez-vous que ce fichier se trouve dans le même répertoire que l'ensemble des assemblages nécessaires à votre application. Ouvrez le fichier avec `Bloc-Notes` et inscrivez-y les commandes suivantes :

```
copy c:\monApp.exe  "c:\Program Files\monApp.exe"
copy c:\maLibrairie.dll "c:\Program Files\maLibrairie.dll"

cd "%windir%\microsoft.net\framework\v1.?.????"

copy c:\maLibGac.dll maLibGac.dll

gacutil.exe /i maLibGac.dll
```

La commande `copy` permet la copie des fichiers à l'endroit précisé puis `gacutil.exe` est utilisé afin d'enregistrer un assemblage au sein du `GAC`. Remarquez comment la commande `CD` est utilisée afin de changer de répertoire en faisant abstraction du numéro de version de la plateforme.NET installée.

Installation à l'aide de *Windows Installer*

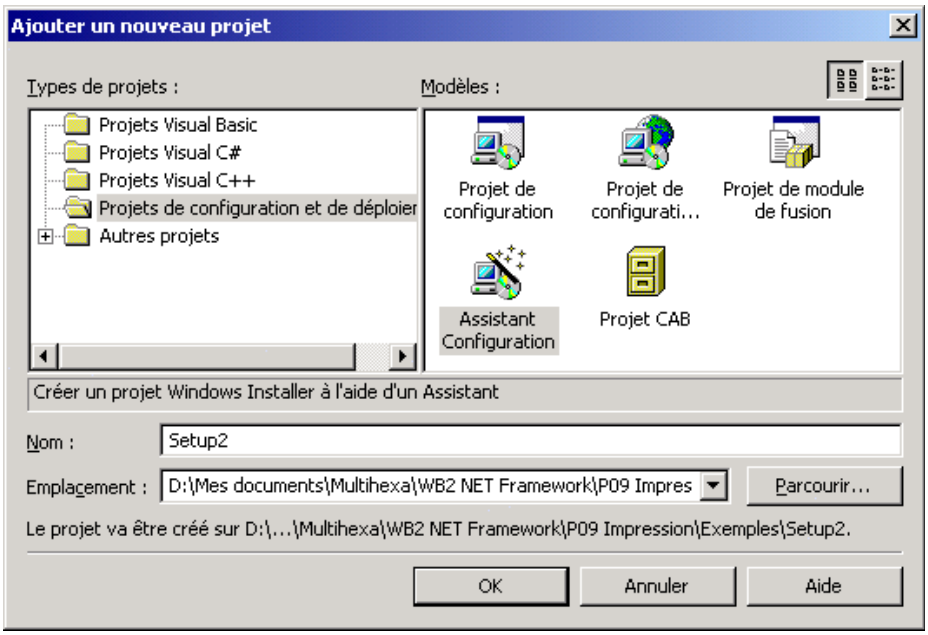
Visual Studio.NET incorpore les fonctionnalités permettant au développeur de concevoir ses logiciels d'installation à l'aide de *Windows Installer*. Cette technique est largement préférable à celle utilisant des fichiers de traitement en lot puisque l'ensemble des étapes de l'installation du logiciel peut aisément être personnalisée par l'utilisateur du fait que celle-ci lui est présentée à l'aide d'interfaces graphiques.



Notez que le logiciel d'installation *Windows Installer* permettra également la vérification du numéro de série, l'exécution d'actions personnalisées permettant entre autre la création de bases de données lors de l'installation, etc.

La présence de *Windows Installer* sur le poste destinataire est nécessaire pour que vos logiciels d'installation soient pris en charge par *Windows*. Nous examinerons plus loin comment installer *Windows Installer* sur les postes exécutant *Windows 9x* et *NT4*. Windows 2000 et les versions ultérieures incorporent *Windows Installer*.

Le logiciel d'installation pour une application existante doit être créé en tant que nouveau projet au sein de la même solution que l'application devant être installée. Ainsi, ouvrez la solution contenant le projet de l'application à distribuer et activez le menu `Fichier → Ajouter un projet → Nouveau projet`. Au sein de la boîte de dialogue vous invitant à préciser le type de projet à ajouter, sélectionnez la catégorie des **Projets de configuration et de déploiement**.



Les modèles de projets de configuration disponibles sont les suivants. Utilisez chacun d'eux selon les besoins spécifiques de votre logiciel d'installation.

Modèle de projet	Description
Projet de configuration	Création de système d'installation afin de déployer une application sur un système Windows.
Projet de configuration Web	Création de système d'installation afin de déployer une application Web sur un hébergeur Internet.
Projet de module de fusion	Création de modules localisables pouvant ensuite être incorporés au sein d'un projet de configuration.
Projet CAB	Création de fichiers CAB encapsulant les fichiers compressés de l'application.
Assistant Configuration	Création de tout type de projet parmi ceux précédemment énuméré avec l'aide d'une assistant.

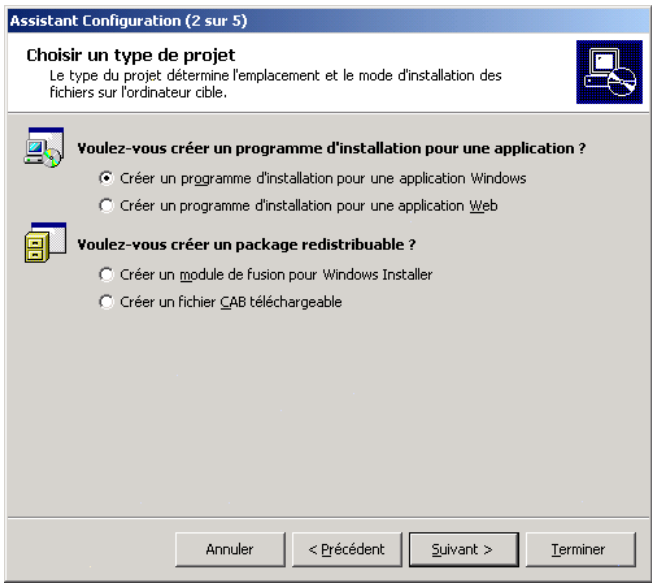
Vous préférerez le plus souvent utiliser l'assistant Configuration afin de créer vos systèmes d'installation puisque celui-ci permet de régler rapidement certains aspects fastidieux de la création d'un projet de configuration.

Nous examinerons d'avantage les projets de configuration Windows et les projets de module de fusion puisque l'utilisation des autres modèles de projet de configuration dépasse largement la portée du présent document.

Programme d'installation pour une application *Windows*

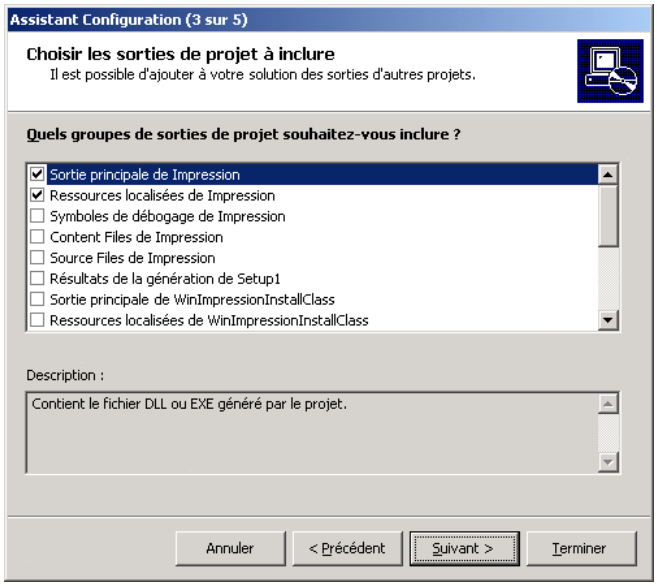
Nous utiliserons l'assistant Configuration afin de créer notre programme d'installation.

Cet assistant permet de régler rapidement certains aspects fastidieux de la création d'un projet de configuration. Aucune des étapes de l'assistant ne sont irrémédiables et il est toujours ultérieurement possible de modifier les comportements du système d'installation définis à l'aide de l'assistant.



Débutez en spécifiant le type de programme d'installation que vous désirez créer. En l'occurrence, nous créerons un **programme d'installation pour une application Windows**.

Appuyez sur le bouton `Suivant` pour continuer.



Choisissez ensuite les groupes de sortie que vous désirez inclure au sein du logiciel d'installation. Les éléments spécifiés seront installés sur le poste client lors de l'exécution du logiciel d'installation.

Notez que l'assistant ne vous permet que d'ajouter des groupes de sortie provenant des projets de la même solution à laquelle votre projet de configuration appartient.

Sélectionnez les groupes de sortie désirés puis appuyez sur le bouton `Suivant` pour continuer.

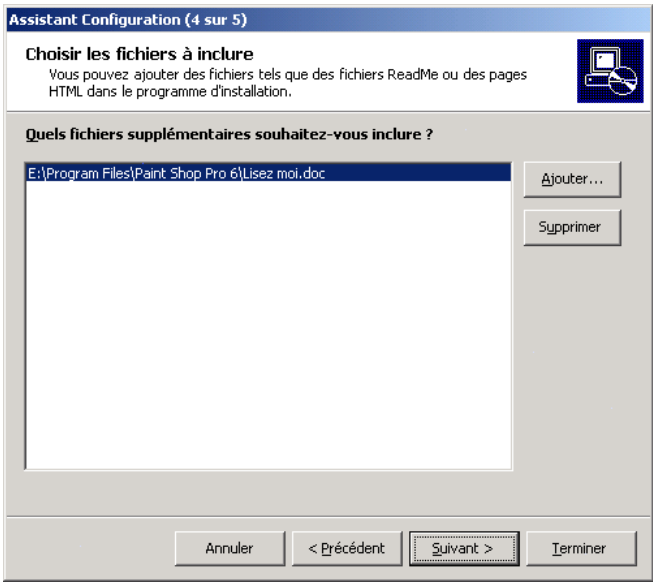
Les différents groupes de sortie sont expliqués à la page suivante.

Groupes de sortie	Description
Sortie principale	Fichiers résultants de la compilation d'un projet (EXE ou DLL).
Ressources localisées	Fichiers ressources (DLL) contenant les chaînes de caractères en différents langages localisés par votre application. Incorporez ce groupe de sortie si vous avez utilisé la propriété `Localizable` au sein d'un formulaire de votre application.
Symboles de débogage	Fichiers de déboguage (PDB) pouvant permettre à un autre développeur de déboguer votre application.
Content Files	Tous les fichiers contenus au sein du répertoire de travail de votre projet.
Sources Files	Fichiers sources constituant votre projet.
Résultats de la génération	Fichiers générés par la création d'un autre logiciel d'installation.

Spécifiez finalement les fichiers supplémentaires que vous désirez ajouter au programme d'installation en appuyant sur le bouton `Ajouter`. Il s'agit là d'un moment privilégié pour ajouter les fichiers du type *ReadMe*, les fichiers de base de données ou les fichiers d'aide.

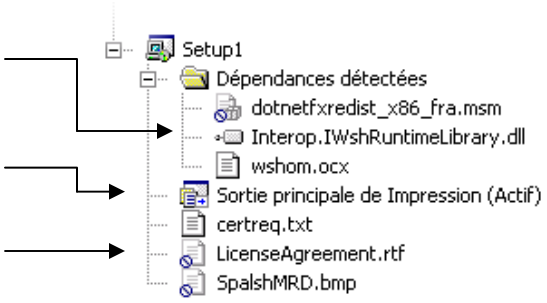
Notez que tout fichier pourra ultérieurement être ajouté au projet même si celui-ci avait préalablement été omis lors de l'exécution du présent assistant.

Appuyez sur le bouton `Suivant` afin de continuer puis terminer.



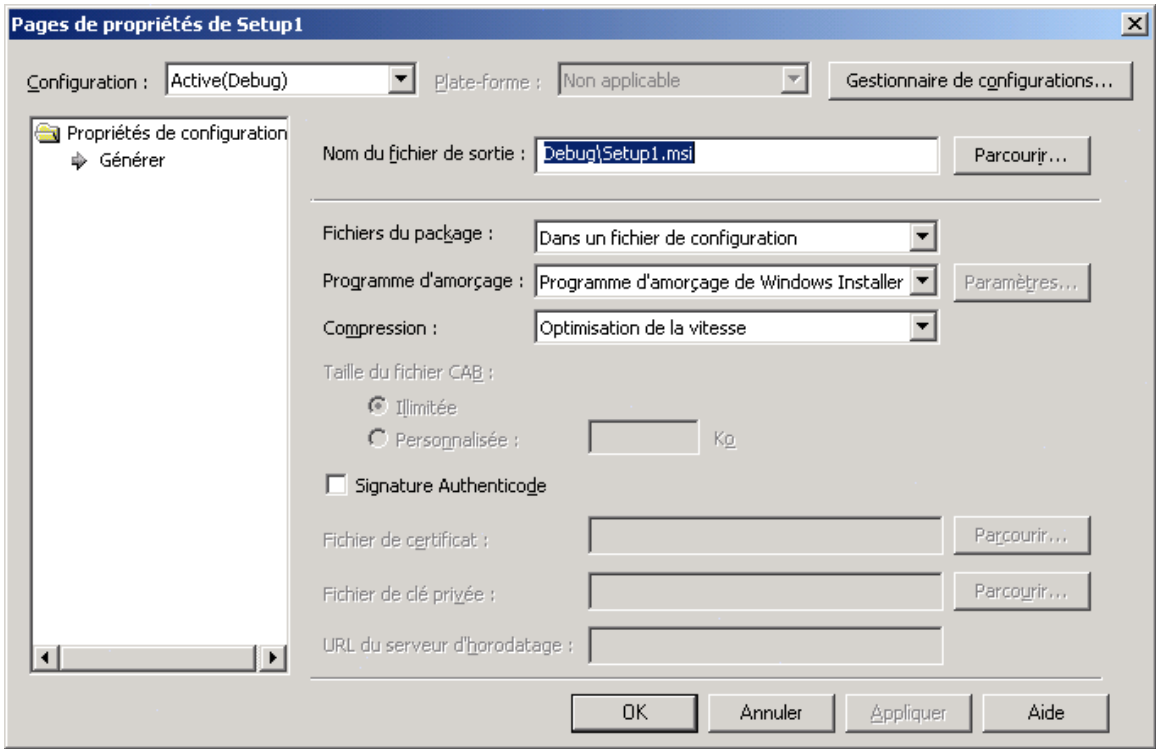
L'environnement de travail de *Windows Installer* s'affiche finalement. Portez d'abord attention au nouveau projet ajouté à la solution existante et aux fichiers qu'il contient. Notez que la liste des fichiers constituant votre logiciel d'installation peut évidemment différer de la liste présentée ci-dessous dépendamment de l'application que vous avez décidé de déployer.

Fichier(s) et module(s) de fusion dont dépend directement l'application.
Sortie(s) principale(s).
Autre(s) fichier(s) ajouté(s) au logiciel d'installation.



Propriétés du projet de configuration

Le projet de configuration expose une multitude de propriétés qu'il est possible de configurer par le biais de la boîte des propriétés et certaines autres qu'il est possible de configurer par le biais d'une page de propriétés. Pour afficher les propriétés du projet de configuration, sélectionnez le projet au sein de l'explorateur de solution et, à l'aide du bouton droit de la souris, activez le menu contextuel **Propriétés**. La page de propriétés suivante s'affichera alors :



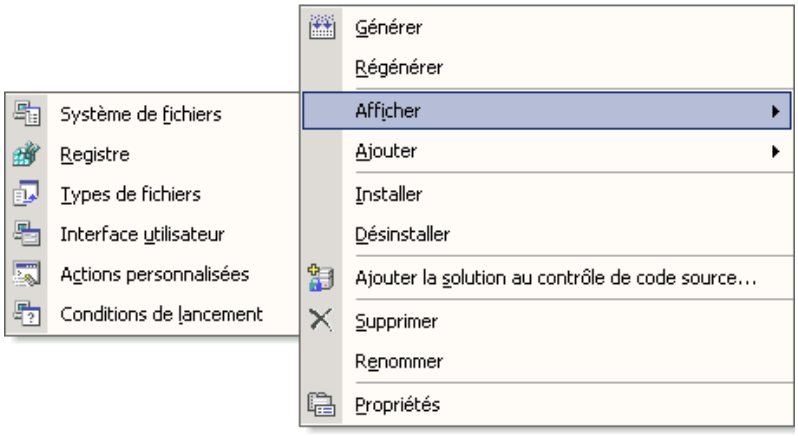
La présente page de propriétés permet de spécifier les propriétés définissant le comportement du compilateur lors de la création du logiciel d'installation.

Propriété	Description
Nom du fichier de sortie	Chemin et nom du fichier qui sera généré lors de la génération du logiciel d'installation. Le chemin peut être absolu ou relatif au chemin du projet en cours. Le fichier doit posséder l'extension MSI.
Fichiers du package	Mode d'encapsulation des fichiers du package. Les fichiers peuvent être compilés au sein du système d'installation MSI, compressés au sein d'un fichier CAB à part ou simplement copiés sans qu'il ne leur soit appliqué aucune compression
Programme d'amorçage	Mode d'encapsulation du programme d'amorce. Celui-ci peut provoquer l'installation à partir de fichiers situés sur le Web ou à partir des fichiers inclus au sein du logiciel d'installation MSI.
Compression	Type de compression à appliquer aux fichiers du logiciel d'installation. Celui-ci peut optimiser la vitesse d'exécution du logiciel d'installation au détriment de la taille du fichier final ou vice versa.
Signature Authenticode	Permet de signer numériquement le logiciel d'installation afin que l'utilisateur puisse en confirmer l'authenticité. La signature est créée à l'aide d'un certificat d'application ou d'une paire de clés générée à l'aide de l'utilitaire `SN.EXE` fourni avec Visual Studio.NET.

Le projet de configuration possède d'autres propriétés qu'il nous est possible de configurer et, bizarrement, celles-ci se trouvent disponibles par le biais de la fenêtre des propriétés. Sélectionnez donc le projet de configuration au sein de l'explorateur de solutions et activez la **fenêtre des propriétés** à l'aide du menu contextuel ou à l'aide de la touche F4. Voici les principales propriétés du projet de configuration :

Propriété	Description
AddRemoveProgramsIcon	Spécifie l'icône à afficher aux côtés de l'application au sein de la fenêtre *Ajout/Suppression de programmes*. Remarquez qu'il est possible de sélectionner un icône à partir d'un fichier ICO, EXE ou DLL. Le fichier doit être ajouté au projet de configuration afin de pouvoir fournir l'icône.
Author	Spécifie le nom de l'auteur de l'application qui sera affiché entre autres au sein de la fenêtre *Ajout/Suppression de programmes*.
Description	Spécifie une description de l'application qui sera affichée entre autres au sein de la fenêtre *Ajout/Suppression de programmes*.
DetectNewerInstalledVersion	Spécifie si l'installation procède à la vérification de l'existence de versions antérieures du même logiciel et en averti l'utilisateur le cas échéant.
Localization	Spécifie le jeu de caractères à utiliser lors de l'affichage des boîtes de dialogue de l'installation.
Manufacturer	Spécifie le nom du manufacturier de l'application qui sera affiché entre autres au sein de la fenêtre *Ajout/Suppression de programmes*.
ManufacturerUrl	Adresse permettant de rejoindre le manufacturier de l'application.
RemovePreviousVersions	Spécifie si le logiciel d'installation doit automatiquement supprimer toute version antérieure du même logiciel.
SupportPhone	Numéro de téléphone permettant de rejoindre le service de la compagnie qui sera affiché entre autres au sein de la fenêtre *Ajout/Suppression de programmes*.
SupportUrl	Adresse Url permettant de rejoindre le service de la compagnie. L'utilisateur peut activer cet Url au sein de la fenêtre *Ajout/Suppression de programmes*.
Title	Titre du système d'installation tel qu'il sera affiché lors de l'installation.
Version	Spécifie la version actuelle de l'application sous la forme #.#.#.#.

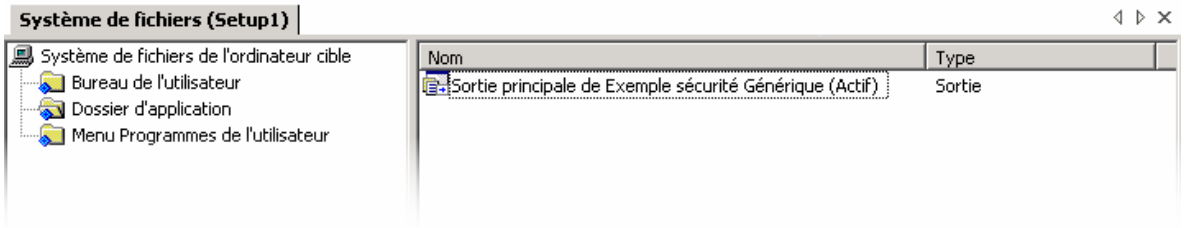
Un grand nombre de différentes autres configurations sont possibles et sont regroupées au sein de grandes sections thématiques. Vous pouvez accéder à ces différentes sections à l'aide du menu `Affichage → Éditeur` ou à l'aide du menu contextuel `Afficher` en appuyant du bouton droit de la souris sur le projet de configuration au sein de l'Explorateur de solutions.



Section système de fichiers

La section `Système de fichiers` vous permet de déterminer l'endroit à lequel les différents fichiers nécessaires au logiciel devront être copiés sur le disque de l'utilisateur. Cette section vous permet également de préciser les raccourcis qui devront être créés dans le menu `Démarrer → Programmes` ou sur le bureau de l'utilisateur.

La fenêtre suivante s'affiche lorsque vous activez la section `Système de fichiers` à l'aide du menu `Affichage → Éditeur` ou à l'aide du menu contextuel `Afficher` en appuyant du bouton droit de la souris sur le projet de configuration au sein de l'Explorateur de solutions.



Le panneau de gauche dresse la liste des dossiers spéciaux disponibles sur le poste de l'utilisateur. On qualifie de spéciaux ces répertoires puisqu'ils représentent relativement un dossier dont le chemin absolu peut changer d'un poste utilisateur à un autre. Par exemple, le dossier System d'un poste *Windows 98* sera, par défaut, situé sur le `c:\windows\system` tandis qu'il se situera sur le `c:\winnt\system32` sous un *Windows 2000*. Ainsi, les dossiers spéciaux permettent de copier des fichiers au sein de ces répertoires changeant sans se soucier de la version du poste destinataire.

Les trois dossiers spéciaux listés sont ceux le plus fréquemment exploités par un logiciel d'installation. Cependant, il est possible d'**ajouter des dossiers spéciaux** en activant le menu `Action → Ajouter un dossier spécial` également accessible par menu contextuel sur le nœud racine de l'arbre composant le panneau de gauche.

Voici la liste et description des dossiers spéciaux disponibles :

Dossier spécial	Description
Fichiers communs	Dossier où sont stockés des fichiers partagés. Généralement dans *c:\program files\fichiers communs|common files.*
Police	Dossier où sont stockées et enregistrées les polices de caractères. Généralement dans *c:\winnt|windows\fonts.*
Program Files	Dossier où sont stockés les programmes. Généralement dans *c:\program files.*
System	Dossier où sont stockés les fichiers du système d'exploitation. Généralement dans *c:\winnt|windows\system|system32* .
Données de l'application de l'utilisateur	Dossier où sont stockés des fichiers d'applications. Généralement dans *c:\documents and settings\utilisateur\ application data.*
Bureau de l'utilisateur	Dossier où sont stockés les fichiers du bureau. Généralement dans *c:\documents and settings\utilisateur\bureau|desktop.*
Favoris de l'utilisateur	Dossier où sont stockés les fichiers du bureau. Généralement dans *c:\documents and settings\utilisateur\favoris|favorites.*

Documents utilisateur	Dossier où sont stockés les documents de l'utilisateur. Généralement dans *c:\documents and settings\utilisateur\ Mes documents*|*My Documents.*
Programmes de l'utilisateur	Menu `Démarrer → Programmes|Programs` de l'utilisateur.
Envoyer vers de l'utilisateur	Élément de menu `Envoyer vers` de l'utilisateur lorsque celui-ci active ce menu contextuel sur un fichier ou un raccourci.
Démarrer de l'utilisateur	Menu `Démarrer` de l'utilisateur.
Démarrage de l'utilisateur	Dossier où sont stockés des fichiers à exécuter au démarrage de la session utilisateur. Généralement dans *c:\documents and settings\utilisateur\menu démarrer/programmes/démarrage.*
Modèles de l'utilisateur	Dossier où sont stockés les fichiers modèles de l'utilisateur. Généralement dans *c:\documents and settings\utilisateur\ modèles.*
Windows	Dossier où sont stockés les applications du système d'exploitation. Généralement dans *c:\winnt*|*windows* .
Global Assembly Cache	Dossier où sont stockés les assemblages partagés par les applications *.NET*. Généralement dans *c:\winnt*|*windows\assembly* .

Notez que le dossier `Dossier de l'application` n'est pas accessible par le menu mais constitue toutefois un dossier spécial qu'il est impossible de supprimer. Il s'agit en effet du dossier représentant l'endroit qui sera précisé par l'utilisateur lors de l'exécution du logiciel d'installation. La majorité de vos fichiers d'application devraient être copiés au sein de ce dossier.

Il est également possible de **créer des dossiers personnalisés** à l'intérieur des dossiers spéciaux prédéfinis par *Visual Studio.NET* en activant le menu `Action → Ajouter un dossier` après avoir sélectionné le dossier ou dossier spécial sous lequel vous désirez créer le nouveau répertoire. Ainsi, le fait de créer un dossier personnalisé nommé `Bin` sous le dossier spécial `Dossier de l'application` créera, lors de l'installation sur un système exécutant *Windows XP* , le dossier `c:\program files\monApplication\Bin\` .

Lorsque vous sélectionnez un dossier spécial ou un dossier personnalisé, les propriétés suivantes sont disponibles au sein de la fenêtre des propriétés :

Propriété du dossier	Description
(Name)	Nom du dossier personnalisé devant être créé. Cette propriété n'est disponible qu'en lecture dans le cas d'un dossier spécial.
AlwaysCreate	Spécifie s'il faut créer le dossier lors de chaque installation et ce même s'il est vide.
Condition	Spécifie une condition qui doit être évaluée par `True` lors de l'installation afin que le dossier soit créé. Consultez la section des conditions couverte plus loin pour plus de détails à ce sujet.
Transitive	Spécifie si le logiciel d'installation doit réévaluer la propriété `Condition` à chaque fois que l'application est installée ou désinstallée.

Après avoir sélectionné le dossier destinataire, il est possible d'**ajouter un nouveau fichier**, une nouvelle sortie d'un projet membre de la présente solution ou un assemblage général de .NET en activant le menu `Action → Ajouter` correspondant.

Propriété du fichier	Description
Condition	Spécifie une condition qui doit être évaluée par `True` lors de l'installation afin que le fichier soit copié. Consultez la section des conditions couverte plus loin pour plus de détails à ce sujet.
Dependencies	Liste des sorties et assemblages .NET dont dépend la présente sortie principale. Disponible dans le cas d'une sortie principale seulement.
Exclude	Permet d'exclure le fichier du logiciel d'installation lorsque vaut `True`. Le fichier n'est pas supprimé du projet de configuration. Cette propriété est utile pour fournir une image BMP en écran de démarrage sans que le fichier ne soit installé sur le poste destinataire.
Folder	Spécifie le dossier dans lequel le fichier sera copié.
Hidden	Permet de spécifier l'activation de l'attribut `Hidden` du fichier lors de sa copie sur le poste destination.
Permanent	Spécifie que le fichier ne devra pas être désinstallé lors du processus de désinstallation de l'application.
ReadOnly	Permet de spécifier l'activation de l'attribut `ReadOnly` du fichier lors de sa copie sur le poste destination.
Register	Permet de spécifier le type d'enregistrement du fichier auprès de la base de registre. Permet entre autres d'enregistrer des composants COM ou des polices de caractères.
SharedLegacy	Permet de spécifier que les logiciels utilisant ce fichier doivent être comptés au sein de la base de registre afin de s'assurer que le fichier n'est pas supprimé tant qu'un logiciel en a besoin. Généralement utilisé pour des composants de classes COM.
System	Permet de spécifier l'activation de l'attribut `System` du fichier lors de sa copie sur le poste destination.
TargetName	Nom du fichier lors de sa copie sur le poste destinataire. Ce nom peut différer du nom original du fichier. La propriété n'est disponible dans le cas d'un fichier seulement.
Vital	Permet de spécifier si le fichier est vital au fonctionnement de l'application. Si la propriété vaut `True`, l'échec de la copie de ce fichier sur le poste destinataire provoque l'échec de l'installation.

Notez que toute sortie principale ou tout fichier DLL devrait généralement posséder la valeur `True` à la propriété `Vital` puisque ces fichiers sont généralement nécessaires au bon fonctionnement de l'application.

De plus, toute base de données format fichier tel que les bases de données *Access* ou *DBase* devraient posséder la valeur `True` à la propriété `Permanent` afin d'éviter de supprimer la base de données seulement parce que l'application est désinstallée afin d'être remplacée par une version plus récente.

Finalement, les fichiers DLL ou OCX exposants un interface COM devraient être enregistrés au sein de la base de registres à l'aide de la propriété `Register` tandis que les fichiers TLB ne devraient en aucun cas être enregistrés de la sorte sous peine de causer une erreur lors de l'installation.

Après avoir sélectionné le dossier destinataire, il est possible de **créer un raccourci** vers un fichier ou une sortie principale existante en activant le menu `Action → Créer un raccourci`. Cette fonctionnalité sera utilisée entre autres afin de créer un raccourci sur le bureau de l'utilisateur ou au sein du menu `Démarrer → Programmes`.

Voici les propriétés d'un raccourci :

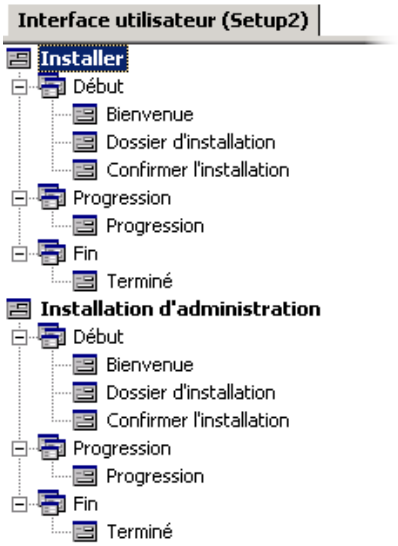
Propriété du raccourci	Description
(Name)	Nom du raccourci tel qu'il sera affiché sous l'icône l'illustrant.
Arguments	Ligne de commande à ajouter lors du lancement de l'exécutable pointé par le raccourci.
Description	Description du fichier pointé par le raccourci. S'affiche en info-bulle lorsque l'utilisateur laisse son pointeur de souris au-dessus de l'icône du raccourci.
Folder	Dossier dans lequel le raccourci sera créé.
Icon	Icône qui illustrera le raccourci qui sera créé.
ShowCmd	Dimensions de la fenêtre qui accueillera l'exécutable pointé par le raccourci *(Minimizée, maximizée ou normale)*.
Target	Chemin et nom du fichier pointé par le raccourci.
Transitive	Spécifie si le logiciel d'installation doit réévaluer la propriété `Condition` à chaque fois que l'application est installée ou désinstallée.
WorkingFolder	Dossier de travail au sein duquel l'exécutable pointé par le raccourci sera démarré.

Si après différentes générations de votre logiciel d'installation un fichier, une sortie principale ou un raccourci s'affiche soulignée d'un trait bleu c'est généralement parce que le fichier source a changé de répertoire ou qu'il n'existe plus. Dans le cas d'un raccourci, l'erreur peut également être causée par la suppression ou la modification de l'icône servant à illustrer le raccourci.

Section interface utilisateur

La section `Interface utilisateur` vous permet de personnaliser les boîtes de dialogue qui seront affichées à l'utilisateur lorsqu'il exécutera votre logiciel d'installation. L'ensemble des boîtes de dialogue ne peut pas être complètement personnalisées mais vous pouvez tout de même choisir des images d'entête, des textes de protection des droits d'auteur, modifier l'ordre des boîtes de dialogues et en ajouter de nouvelles. Plus loin cependant, vous verrez comment créer vos boîtes de dialogues personnalisées à l'aide des actions personnalisées.

La fenêtre ci-contre s'affiche lorsque vous activez la section `Interface utilisateur` à l'aide du menu `Affichage → Éditeur` ou à l'aide du menu contextuel `Afficher` en appuyant du bouton droit de la souris sur le projet de configuration au sein de l'Explorateur de solutions.

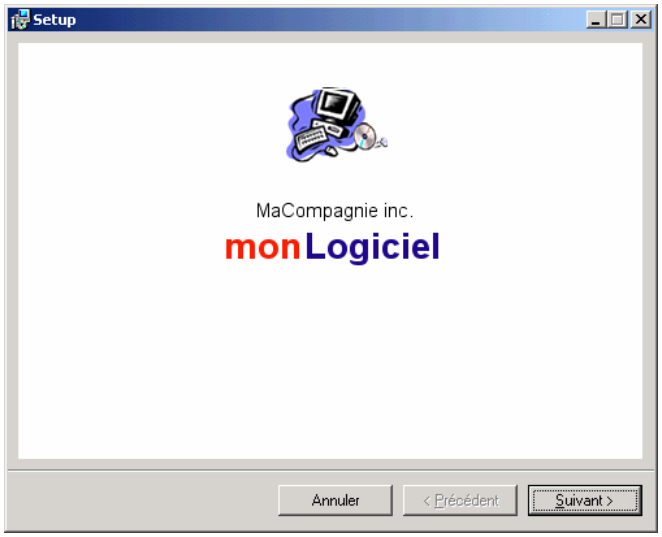


La fenêtre dresse la liste des boîtes de dialogue constituant par défaut tout nouveau projet de configuration.

Sélectionnez une interface utilisateur afin que la fenêtre des propriétés n'en affiche ses propriétés.

Il est possible d'**ajouter d'autres interfaces** en sélectionnant le menu `Action → Ajouter une boîte de dialogue` ou à l'aide du menu contextuel sur la section désirée.

Boîte de dialogue	Description
Bienvenue	Écran de bienvenue.
Dossier d'installation	Boîte de dialogue permettant à l'utilisateur de préciser le dossier au sein duquel il désire installer l'application.
Confirmer l'installation	Boîte de dialogue invitant simplement l'utilisateur à confirmer l'exécution de l'installation.
Progression	Boîte de dialogue affichant la progression de l'installation à l'aide d'une barre de progression facultative.
Terminé	Boîte de dialogue affichant à l'utilisateur que l'installation a réussie.
Cases à cocher	Boîte de dialogue contenant des cases à cocher personnalisables.
Cases d'option	Boîte de dialogue contenant des cases d'option personnalisables.
Zones de texte	Boîte de dialogue contenant des zones de texte personnalisables.
Contrat de licence	Boîte de dialogue affichant un contrat de licence à partir d'un document **.rtf*.
Lisez-moi	Boîte de dialogue affichant un *Lisez-moi* à partir d'un document **.rtf*.
Écran de démarrage	Écran de démarrage affichant une image personnalisée.
Informations client	Boîte de dialogue permettant de saisir le nom de l'utilisateur, nom de compagnie et un numéro de série.
Inscrire un utilisateur	Boîte de dialogue permettant à l'utilisateur de lancer un exécutable lui permettant de s'inscrire en ligne.



La boîte de dialogue **Ecran de démarrage** permet d'afficher une image à l'utilisateur au démarrage de l'installation.

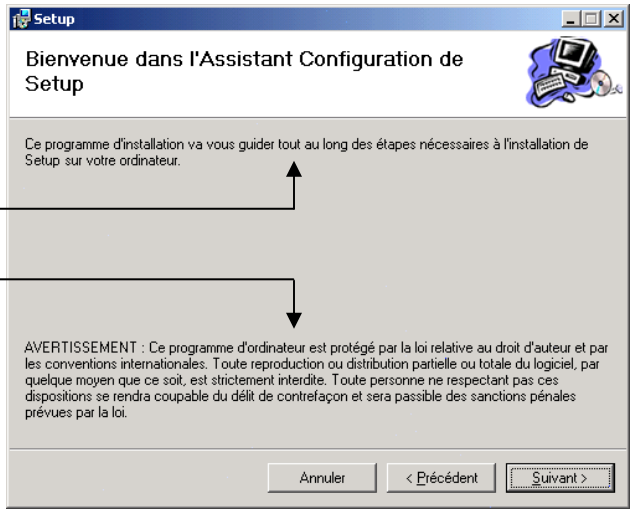
Spécifiez le fichier devant fournir l'image à l'aide de la propriété `SplashBitmap` de la boîte de dialogue.

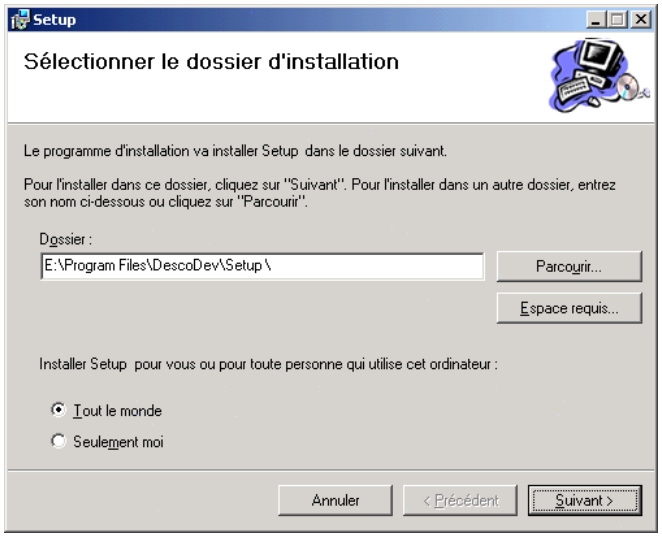
L'image doit être un *BMP* ou un *JPG* idéalement de 475 x 315 pixels.

La boîte de dialogue **Bienvenue** permet d'afficher un mot de bienvenue à l'utilisateur.

La propriété `WelcomeText` permet de spécifier le message de bienvenue affiché immédiatement sous l'entête.

La propriété `CopyrightWarning` permet d'afficher un avertissement de droits d'auteurs au sein de la boîte de dialogue. Notez qu'il est également possible d'afficher un contrat de licence en ajoutant la boîte de dialogue `Contrat de licence` présentée plus loin.





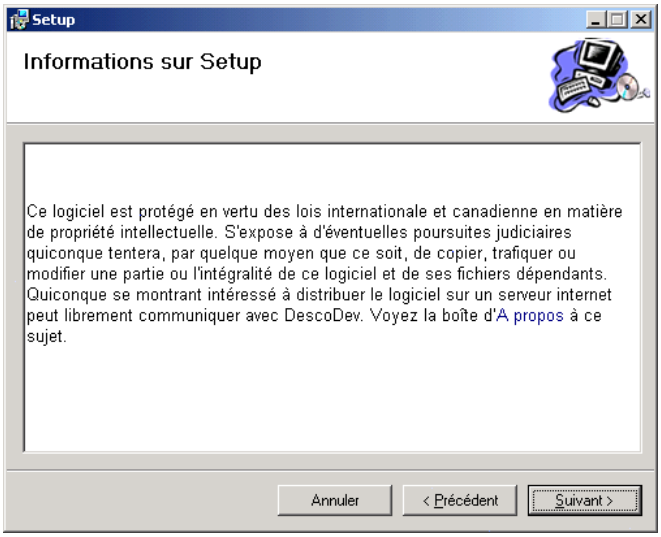
La boîte de dialogue **Dossier d'installation** permet à l'utilisateur de spécifier le dossier au sein duquel il désire voir l'application s'installer.

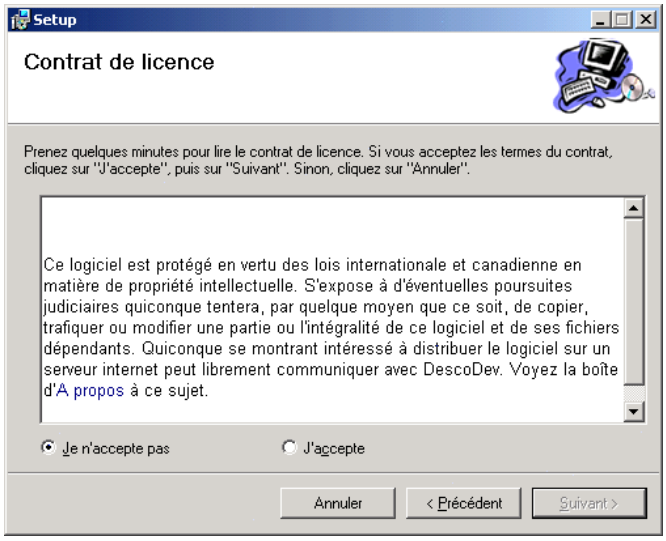
L'utilisateur peut également choisir s'il désire voir l'application être disponible par l'ensemble des utilisateurs du poste ou seulement par lui-même.

La boîte de dialogue **Lisez-moi** permet d'afficher un texte quelconque à l'utilisateur.

Pour spécifier le contenu du texte, il suffit de créer un document texte formaté RTF et d'en spécifier le chemin complet à la propriété `ReadmeFile`.

Il est possible de créer un document texte formaté RTF à l'aide de *Microsoft Word* en s'assurant de sauvegarder le document sous le format approprié.





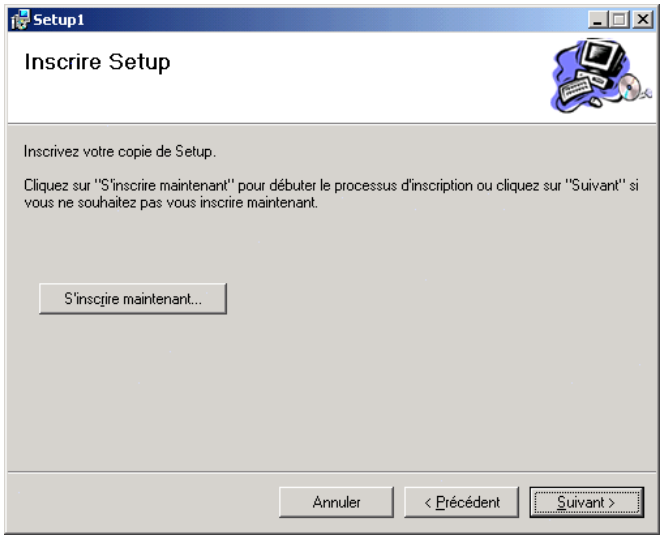
La boîte de dialogue **Contrat de licence** permet d'afficher un contrat de licence à l'utilisateur. Cette boîte de dialogue se distingue de la boîte de dialogue *Lisez-moi* du fait que cette dernière ne prévoit pas de cases d'option *J'accepte* et *Je refuse*.

Le logiciel d'installation se termine si l'utilisateur n'accepte pas le contrat de licence.

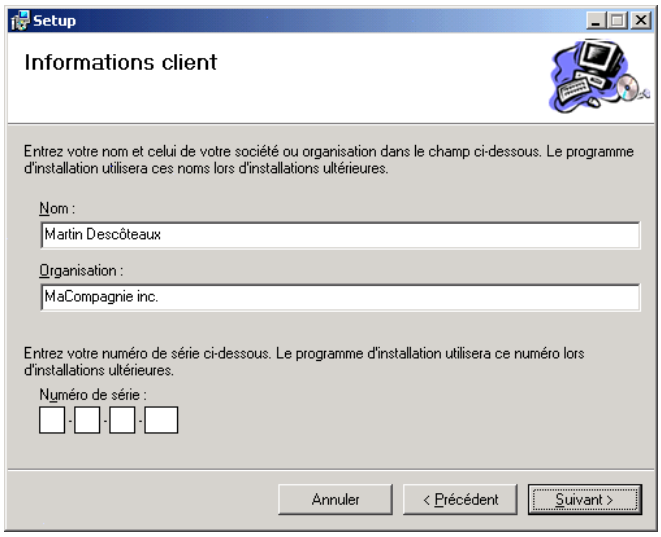
Pour spécifier le contenu du contrat de licence, il suffit de créer un document texte formaté RTF et d'en spécifier le chemin complet à la propriété `LicenseFile`.

La boîte de dialogue **Inscrire un utilisateur** permet d'inviter l'utilisateur à inscrire en ligne sa copie de l'application.

Pour inscrire un utilisateur en ligne, vous devez créer un exécutable supplémentaire que vous spécifierez à la propriété `Executable`. Lorsque l'utilisateur appuiera sur le bouton S'inscrire maintenant, cet exécutable sera lancé et les arguments facultatifs précisés à la propriété `Arguments` lui seront passés en ligne de commande. Notez qu'il est très simple d'inscrire un utilisateur en ligne en concevant un exécutable utilisant un service Web que vous aurez préalablement créé.



La boîte de dialogue **Informations client** permet d'inviter l'utilisateur à saisir son nom et le nom de son organisation.

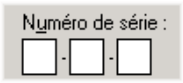


Si vous affectez la valeur `True` à la propriété `ShowSerialNumber`, la boîte de dialogue invitera également l'utilisateur à saisir son numéro de série. Le numéro de série ainsi saisi pourra immédiatement être validé par un algorithme de vérification intégré à Windows Installer ou pourra être ultérieurement récupéré à l'aide d'actions personnalisées. Cette dernière option sera étudiée plus loin au sein du présent chapitre.

Si vous décidez d'inviter l'utilisateur à saisir un numéro de série, ce dernier devra correspondre au format défini à l'aide de la propriété `SerialNumberTemplate`. Cette propriété n'a aucune incidence sur l'installation si la propriété `ShowSerialNumber` possède la valeur `False`.

La propriété `SerialNumberTemplate` permet de spécifier le format de saisie du numéro de série entre chevrons. Chaque groupe de caractères peuvent être divisés à l'aide d'un trait-d'union, ce qui aura pour conséquence d'afficher à l'utilisateur autant de zones de texte que de groupes de caractères. Par exemple, le format suivant générera l'affichage ci-contre :

```
<###-###-###>
```



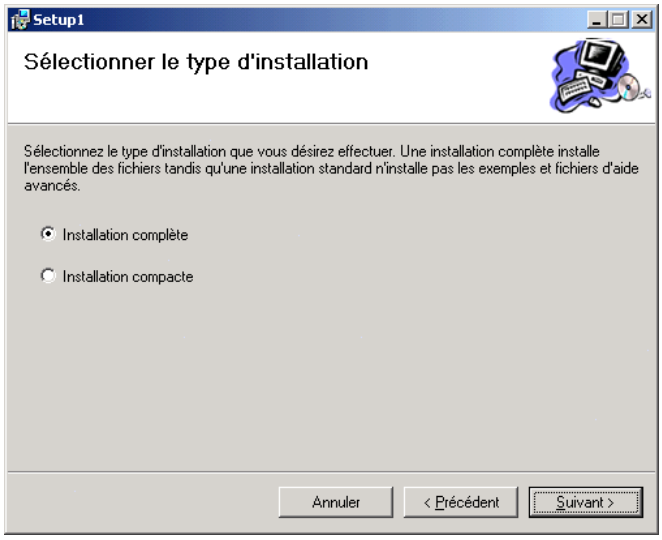
Le format de saisie peut contenir les symboles suivants :

Symbole	Description
#	Exige un chiffre qui ne sera pas vérifié par l'algorithme de validation.
?	Exige un caractère alphanumérique qui ne sera pas vérifié par l'algorithme de validation.
^	Exige une lettre majuscule qui ne sera pas vérifié par l'algorithme de validation.
%	Exige un chiffre qui sera vérifié par l'algorithme de validation.

Si vous utilisez tout autre caractère que ceux-ci, ce caractère sera pré-inscrit dans la boîte de dialogue et sera affiché à l'utilisateur tel quel sans qu'il ne puisse être modifié.

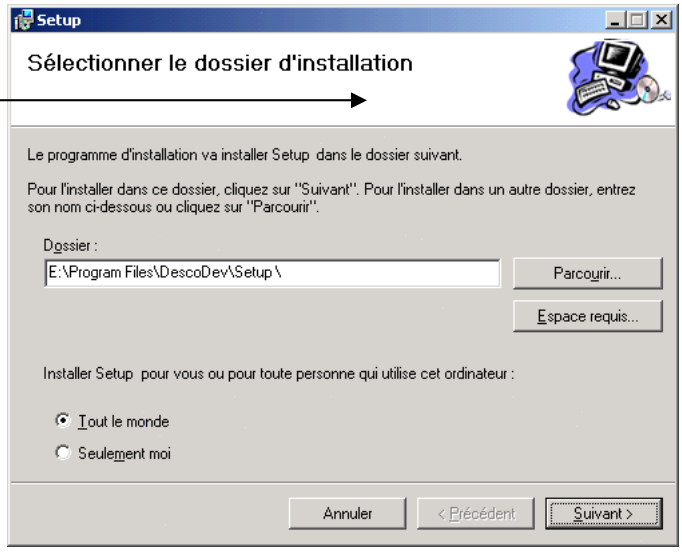
L'algorithme de validation utilisé par *Windows Installer* stipule que tout numéro de série dont la somme des chiffres vérifiés est divisible par sept sans laisser aucun reste est valide. Cet algorithme est appliqué indépendamment pour chacun des groupes de caractères.

SerialNumberTemplate	Numéro valide	Numéro invalide
<##-%%#-%%%>	11-521-232	11-441-333
<##-%#%#-%#%%>	11-3141-2157	11-3141-2156
<#%#-%%#-%%%>	171-431-734	111-111-111
<#%^-%%#-^%%>	17A-771-A77	777-777-777



Les boîtes de dialogue **Cases d'options**, **Cases à cocher** et **Zones de texte** permettent d'inviter l'utilisateur à saisir diverses informations que vous pouvez personnaliser. Ces informations pourront ensuite être traitées à l'aide des conditions d'installation des fichiers ou à l'aide des actions personnalisées. Ces options seront étudiées plus loin au sein du présent chapitre.

Les propriétés `EditProperty`, `CheckboxProperty` et `ButtonProperty` permettront de connaître la valeur spécifiée par l'utilisateur au sein de chacun des contrôles.



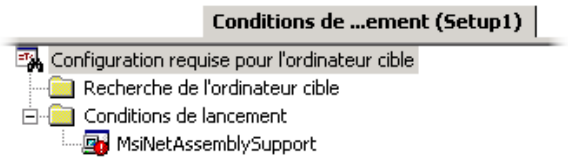
L'ensemble des interfaces possèdent la propriété `BannerBitmap` permettant de préciser une image en entête de la boîte de dialogue. *Windows Installer* utilisera une image par défaut si vous n'en précisez aucune.

L'image doit être un *BMP* ou un *JPG* idéalement de 500 x 75 pixels. N'utilisez pas une image de plus petites dimensions car *Windows Installer* la disposera au centre de l'entête au risque d'afficher le texte de l'entête par-dessus l'image.

Section conditions de lancement

La section `Conditions de lancement` vous permet de conditionner le lancement de votre logiciel d'installation en spécifiant une condition qui devra nécessairement être évaluée positivement afin que l'installation se complète. De la même manière, il est possible d'appliquer des conditions à la copie et l'installation de chacun des fichiers composant le logiciel d'installation.

La fenêtre ci-contre s'affiche lorsque vous activez la section `Conditions de lancement` à l'aide du menu `Affichage → Éditeur` ou à l'aide du menu contextuel `Afficher` en appuyant du bouton droit de la souris sur le projet de configuration au sein de l'Explorateur de solutions.



La fenêtre dresse la liste des conditions de lancement et vous permet d'en ajouter de nouvelles. Il est possible d'**ajouter une condition de lancement** en sélectionnant le menu `Action → Ajouter une condition de lancement` ou à l'aide du menu contextuel sur le nœud `Conditions de lancement`.

Remarquez qu'une condition de lancement est présente par défaut : *MsiNetAssemblySupport*. Cette condition s'assure que le poste de travail cible possède la plate-forme .NET afin que l'installation et l'exécution de votre application s'effectuent correctement.

Chaque condition de lancement possède les propriétés suivantes :

Propriété	Description
(Name)	Nom de la condition permettant de l'identifier au sein de la liste des conditions de lancement.
Condition	Expression à évaluer lors de l'installation. L'installation échoue si la condition est évaluée négativement.
Message	Message à afficher à l'utilisateur si l'installation ne peut s'exécuter correctement dû à l'échec de cette condition de lancement.

Toute expression pouvant être attribuée à la propriété `Condition` d'une condition de lancement peut également être attribuée à la propriété `Condition` d'un fichier. Ainsi, la condition de copie et d'installation de ce fichier sera évaluée individuellement des autres fichiers et, au cas où l'expression est évaluée négativement, aucun message ne sera affiché à l'utilisateur mais le fichier ne sera pas copié. Cette technique est utile afin de créer des types d'installation *(complète, compacte, etc.)* ou pour choisir entre plusieurs fichiers fortement dépendants de la version du système d'exploitation cible.

Toute expression pouvant être attribuée aux propriété `Condition` d'une condition de lancement ou de copie d'un fichier peut incorporer plusieurs éléments séparés par les opérateurs logiques AND, OR, NOT et XOR. Exemple :

Condition = `VersionNT > 0 AND AdminUser = True`

Condition	Description
AdminUser	Vaut True lorsque l'utilisateur qui a lancé l'installation est membre du groupe *Administrateur (Ex: AdminUser = True)*.
ComputerName	Contient le nom du poste *(Ex: ComputerName = "bebitte")*.
LogonUser	Contient le nom de l'utilisateur qui a lancé l'installation.
ServicePackLevel	Contient le numéro de version du dernier *Service Pack* installé *(Ex: ServicePackLevel > 1 AND VersionNT > 400)*.
Version9x	Contient un numéro identifiant le système 9x *(Voir le tableau plus loin)*.
VersionNT	Contient un numéro identifiant le système NT *(Voir le tableau plus loin)*.
WindowsBuild	Contient le numéro de *Build* du système d'exploitation.
SourceDir	Contient le chemin à partir duquel l'installation a été lancée.
TARGETDIR	Contient le chemin choisi par l'utilisateur comme destination de l'installation.
Date	Contient la date courante sous forme de chaîne de caractères au format MM-DD-YYYY.
Time	Contient l'heure courante sous forme de chaîne de caractères au format HH:MM:SS.
Intel	Contient la version du processeur si celui-ci est de famille Intel. *Exemples*: 4 pour un 486, 5 pour un Pentium, 6 pour un Pentium II.
PhysicalMemory	Contient la quantité de RAM disponible exprimée en mégaoctets *(Ex: PhysicalMemory > 256)*.
ScreenY	Contient la résolution de l'écran *(hauteur)* en pixels.
ScreenX	Contient la résolution de l'écran *(largeur)* en pixels.
Installed	Indique si le produit est déjà installé sur le poste.
OutOfDiskSpace	Indique si le disque contient suffisamment d'espace pour accueillir le logiciel.
COMPANYNAME	Contient le nom de compagnie saisi par l'utilisateur lors de l'installation.
ProductID	Contient le numéro de série saisi par l'utilisateur lors de l'installation.
MsiNetAssemblySupport	Indique si la plateforme .NET est installée sur le poste.

Voici les valeurs possibles des propriétés *Version9x*, *VersionNT* et *WindowsBuild* .

Système	Version9x	VersionNT	WindowsBuild
Windows 95	400		
Windows 95 (Gold)	400		950
Windows 98	410		
Windows 98 (Gold)	410		1998
Windows ME	490		
Windows NT4		400	1381
Windows 2000		500	2195
Windows XP		501	2526 *ou plus*
Windows 2003 Server		502	

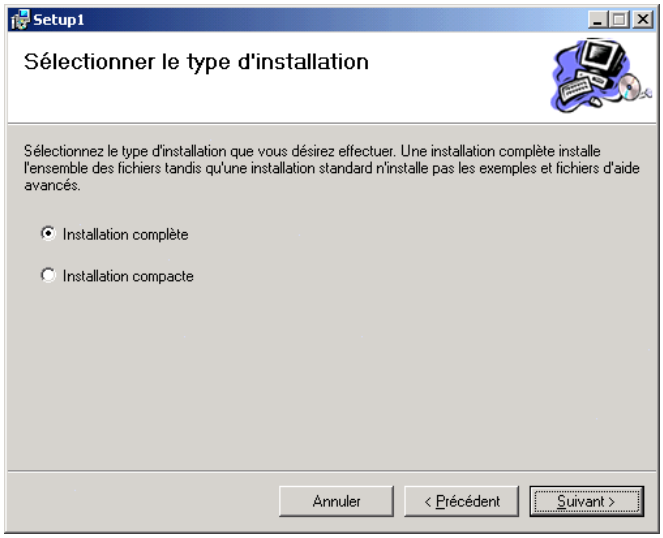
La liste des conditions présentée précédemment n'est pas exhaustive. Vous pourrez obtenir la liste complète des conditions de lancement en consultant l'aide MSDN.

Votre logiciel d'installation peut récupérer les valeurs saisies par l'utilisateur au sein de boîtes de dialogue personnalisées (*Cases à cocher*, *Cases d'option* et *Zones de texte*) dans le but de les utiliser au sein de conditions de lancement. La valeur `Property` retournée par chacune de ces boîtes de dialogues peut être récupérée lorsqu'utilisée entre crochets. Par exemple, si votre logiciel d'installation prévoit l'affichage d'une boîte de dialogue affichant deux cases à cocher et que la propriété `ButtonProperty` vaut `InstallationType`, une condition de lancement pourra récupérer la valeur saisie par l'utilisateur comme suit :

```
[InstallationType] = 1
```

Toute expression pouvant être attribuée à la propriété `Condition` d'une condition de lancement peut également être attribuée à la propriété `Condition` d'un fichier. Ainsi, la condition de copie et d'installation de ce fichier sera évaluée individuellement des autres fichiers et, au cas où l'expression est évaluée négativement, aucun message ne sera affiché à l'utilisateur mais le fichier ne sera pas copié. Cette technique est utile afin de créer des types d'installation *(complète, compacte, etc.)* ou pour choisir entre plusieurs fichiers fortement dépendants de la version du système d'exploitation cible.

L'exemple suivant installe un fichier nommé seulement lors d'une installation complète du logiciel. Une boîte de dialogue de type *Cases d'option* a été ajoutée au logiciel d'installation et sa propriété `ButtonProperty` a été définie à la valeur `InstallationType`. Le premier bouton radio possède l'étiquette `Complete` et le second possède l'étiquette `Compacte`.



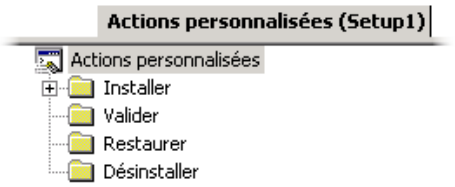
La propriété Condition du fichier devra donc posséder la valeur suivante afin qu'il ne soit installé seulement lors d'une installation complète :

```
[InstallationType] = 0
```

Section actions personnalisées

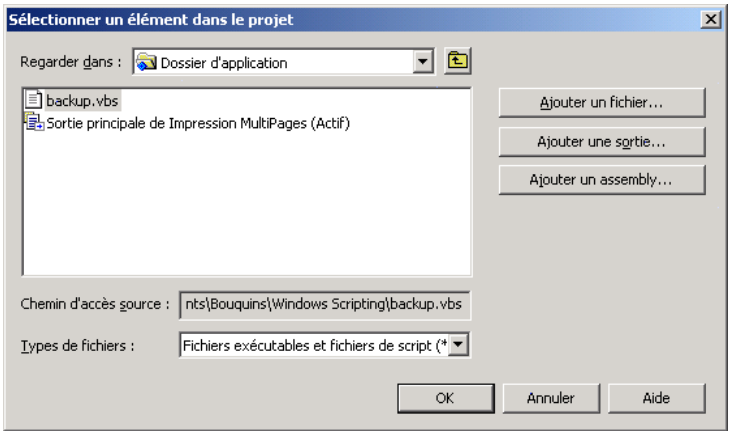
La section `Actions personnalisées` vous permet d'accomplir certaines tâches personnalisées lors de l'installation, la validation, la restauration ou la désinstallation de votre application. À l'aide des actions personnalisées, il sera possible à votre logiciel d'installation de lancer l'exécution de scripts *Windows Script Host*, d'exécutables EXE standard ou .NET ainsi que de classes *Windows Installer* sous forme de librairie DLL.

La fenêtre ci-contre s'affiche lorsque vous activez la section `Actions personnalisées` à l'aide du menu `Affichage → Éditeur` ou à l'aide du menu contextuel `Afficher` en appuyant du bouton droit de la souris sur le projet de configuration au sein de l'Explorateur de solutions.



La fenêtre dresse la liste des actions personnalisées catégorisées selon le moment à lequel elles doivent être activées par *Windows Installer*. Il est possible d'**ajouter une action personnalisée** en sélectionnant le menu `Action → Ajouter une action personnalisée` ou à l'aide du menu contextuel sur la section désirée.

Lorsque vous ajoutez une action personnalisée, vous serez invité à spécifier le fichier contenant le code à exécuter lorsque l'action sera lancée.



Le code de l'action personnalisée peut être contenu au sein d'un fichier **.vbs* ou **.js* s'il s'agit d'un script *Windows Script Host*. Dans ce cas, assurez-vous que le fichier est exclu de l'installation en configurant la propriété `Exclude` du fichier à la valeur `True`.

Puisque la création de scripts *Windows Script Host* ne prend pas en charge la nouvelle syntaxe *Visual Basic.NET*, nous préférerons ajouter des actions personnalisées utilisant un EXE ou une classe `Installer` pouvant tous deux préalablement être conçus à l'aide de *Visual Studio.NET*.

Le code de l'action personnalisée peut être contenu au sein d'un fichier exécutable EXE standard ou .NET. Dans ce cas, ajoutez le fichier au sein du projet et assurez-vous qu'il est inclu dans l'installation en configurant la propriété `Exclude` du fichier à la valeur `False`.

Finalement, vous pouvez utiliser une sortie principale générée par la compilation d'un projet appartenant à la même solution que le projet de configuration en ajoutant la sortie principale à la liste de fichiers constituant l'installation puis en la sélectionnant comme code à exécuter lors du lancement de l'action personnalisée. Dans le cas d'une librairie contenant une classe `Installer`, vous devrez vous assurer d'inclure le fichier en configurant la propriété `Exclude` du fichier à la valeur `False`. Les classes `Installer` seront brièvement étudiées plus loin.

Par exemple, pour procéder à la **création d'une action personnalisée** qui lancera l'affichage du site Web officiel de votre compagnie au sein du navigateur Internet du client :

- Ajoutez un nouveau projet au sein de la solution en cours à l'aide du menu `Fichier → Ajouter un projet → Nouveau projet`. Au sein de la boîte de dialogue affichant les modèles de projets, choisissez le modèle `Application Windows` et le langage Visual Basic. Nommez votre projet `OuvrirWeb`.

- Ajoutez un nouveau `Module` au sein de ce projet et inscrivez-y le code suivant :

```
Public Sub Main()
    Process.Start("http://www.votrecompagnie.com")
End Sub
```

- Sélectionnez le projet `OuvrirWeb` au sein de l'explorateur de solutions et, à l'aide du bouton droit de la souris, activez le menu contextuel `Propriétés` afin d'accéder aux pages de propriétés de ce projet. Dans la liste déroulant `Objet de démarrage`, choisissez l'élément `Sub Main`.

- Générez le projet `OuvrirWeb` à l'aide du menu `Générer → Générer OuvrirWeb`.

- Sélectionnez désormais votre projet de configuration au sein de l'explorateur de solution et assurez-vous de faire afficher l'éditeur des actions personnalisées. Sous le nœud `Installer`, créez une action à l'aide du menu `Action → Ajouter une action personnalisée`.

- Dans la boîte de dialogue `Sélectionner un élément dans le projet`, double-cliquez sur `Dossier d'application` et appuyez sur le bouton `Ajouter une sortie`. Sélectionnez l'élément `Sortie principale de OuvrirWeb` au sein de la liste des sorties disponibles au sein de la solution en cours.

- Lorsque l'action personnalisée s'affichera au sein de la liste des actions de votre projet de configuration, sélectionnez-la et assurez-vous que la propriété `InstallerClass` possède la valeur `False`. Vous êtes maintenant prêt à générer votre solution et à démarrer l'installation.

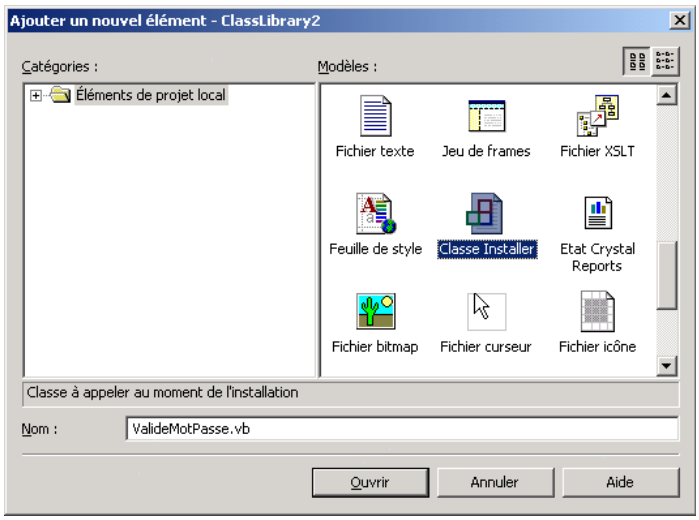
Voici la description des propriétés des actions personnalisées :

Propriété	Description
Arguments	Ligne de paramètres pouvant être fournis à l'exécutable qui sera lancé. Non-disponible lorsque l'action est un script ou une librairie DLL.
Condition	Condition de lancement de l'action personnalisée *(voir section précédente)*. L'action ne sera pas lancée si la condition n'est pas rencontrée.
CustomActionData	Informations supplémentaires pouvant être fournis au script ou à la librairie dans le format `/propriété=[valeur]` .
EntryPoint	Nom de la procédure d'entrée de la librairie DLL. Non-disponible pour les autres types d'actions personnalisées puisque ceux-ci possèdent un point d'entrée bien spécifique.
InstallerClass	Spécifie si la classe est une classe dérivant de la classe `Installer`.

Pour procéder à la **création d'une classe Installer** afin d'y stocker le code devant s'exécuter lors du lancement de l'action personnalisée. L'action suivante permettra de récupérer le numéro de série saisi par l'utilisateur et de le soumettre à vérification supplémentaire à celle effectuée par l'algorithme de validation de *Windows Installer*.

- Ajoutez un nouveau projet au sein de la solution en cours à l'aide du menu `Fichier → Ajouter un projet → Nouveau projet`. Au sein de la boîte de dialogue affichant les modèles de projets, choisissez le modèle `Bibliothèque de classes` et le langage Visual Basic. Nommez votre projet `ValideNoSerie`.

- Supprimez la classe automatiquement ajoutée à votre projet et ajoutez un nouvel élément de type `Classe Installer` tel que démontré ci-contre.



- Remarquez que la classe créée dérive de la classe `System.Configuration.Install.Installer` et qu'elle possède un conteneur à composants vous permettant d'y déposer des composants de la boîte à outils.

- Localisez le masquage de la procédure `Install` au sein de l'élément *(Overrides)* de la liste déroulante de l'éditeur de code et inscrivez-y le code suivant. Nous reviendrons plus loin sur la description des éléments de ce code.

```
Public Overrides Sub Install(ByVal stateSaver As ... )

   If Context.Parameters.Item("key") <> "123-12345" Then
      Throw New InstallException("Numéro de série invalide!")
   End If

End Sub
```

- Sélectionnez désormais votre projet de configuration au sein de l'explorateur de solution et assurez-vous de faire afficher l'éditeur des actions personnalisées. Sous le nœud `Installer`, créez une action à l'aide du menu `Action → Ajouter une action personnalisée`.

- Dans la boîte de dialogue `Sélectionner un élément dans le projet`, double-cliquez sur `Dossier d'application` et appuyez sur le bouton `Ajouter une sortie`. Sélectionnez l'élément `Sortie principale de ValideNoSerie` au sein de la liste des sorties disponibles au sein de la solution en cours.

- Assurez-vous que la propriété `InstallerClass` possède la valeur `False` et que la propriété `CustomActionData` possède la valeur `/key=[ProductId]`. Vous êtes maintenant prêt à générer votre solution et à démarrer l'installation.

Les diverses valeurs récoltées par *Windows Installer* auprès de l'utilisateur exécutant l'installation peuvent être transférées à la classe `Installer` à l'aide de la propriété `CustomActionData`. Les valeurs peuvent provenir de propriétés de l'installation telles que celles testées par les conditions de lancement ou peuvent provenir des zones de textes ou autres contrôles présents au sein des boîtes de dialogue de l'installation.

Ainsi, si le code de votre classe `Installer` nécessite de connaître le nom de l'utilisateur exécutant l'installation, la propriété `CustomActionData` posséderait la valeur suivante :

```
/nom=[LogonUser]
```

Consultez la section *Conditions de lancement* présentée précédemment afin de prendre connaissance de la liste des propriétés les plus fréquemment utilisées.

Si le code de votre classe `Installer` nécessite de connaître la valeur saisie par l'utilisateur au sein d'une zone de texte d'une boîte de dialogue spécifique, la propriété `CustomActionData` devra transférer le nom de la zone de texte. Le nom de la zone de texte est défini par la propriété `EditProperty` au sein de la section *Interfaces utilis*ateur tel que présenté précédemment.

```
/texte=[EditPropertyDeLaZoneTexte]
```

Finalement, la classe `Installer` peut récupérer les valeurs des différentes propriétés précisées au sein de `CustomActionData` en utilisant l'objet `Context` intrinsèque à la classe `Installer`. Cet objet prévoit une collection nommée `Parameters` contenant l'ensemble des paramètres envoyés par *Windows Installer*.

```
If Context.Parameters.Item("nom") <> "valeur" Then

End If
```

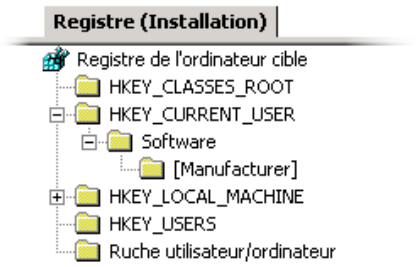
Finalement, la classe `Installer` peut **provoquer l'arrêt de l'installation** en cours simplement en soulevant une exception à l'aide de l'instruction `Throw` :

```
Public Overrides Sub Install(ByVal stateSaver As ... )

   If Context.Parameters.Item("key") <> "123-12345" Then
       Throw New InstallException("Numéro de série invalide!")
   End If

End Sub
```

Section registre

La section `Registre` vous permet d'ajouter des clés et valeurs personnalisées à la base de registre du poste utilisateur cible.

La fenêtre ci-contre s'affiche lorsque vous activez la section `Registre` à l'aide du menu `Affichage → Éditeur` ou à l'aide du menu contextuel `Afficher` en appuyant du bouton droit de la souris sur le projet de configuration au sein de l'Explorateur de solutions.



Notez que les nœuds libellés *[Manufacturer]* porteront le nom que vous aurez spécifié à la propriété `Manufacturer` du projet de configuration.

Il est possible d'importer la liste des clés et valeurs à créer lors de l'exécution de *Windows Installer* à partir d'un fichier REG préalablement créé à l'aide de l'éditeur de registre en sélectionnant le nœud racine nommé `Registre de l'ordinateur cible` avant d'activer le menu `Action → Importer` ou le menu contextuel sur ce même nœud. Vous serez ensuite invité à parcourir les dossiers de votre poste afin de sélectionner le fichier REG contenant l'arborescence de registre à créer lors de l'installation.

Pour créer un fichier REG à l'aide de l'éditeur de registre, démarrez `RegEdit` à partir du menu `Démarrer → Exécuter` de *Windows* puis localisez le nœud que vous désirez exporter au sein du fichier. Activez ensuite le menu `Registre → Exporter un fichier du registre` et spécifiez le nom et l'endroit du fichier à créer. Notez que l'ensemble des sous-clés et des valeurs contenues sous la clé que vous aurez sélectionnée sera exporté.

Vous pouvez créer manuellement l'arborescence de registre désirée en ajoutant les clés et valeurs devant la constituer. Pour **ajouter une nouvelle clé**, sélectionnez la clé sous laquelle la nouvelle clé devra être créée puis activez le menu `Action → Nouveau → Clé` ou le menu contextuel.

Lorsque vous sélectionnez une clé, les propriétés suivantes sont disponibles au sein de la fenêtre des propriétés. Notez qu'aucune propriété n'est disponible concernant les clés système.

Propriété de la clé	Description
(Name)	Nom de la clé devant être créée.
AlwaysCreate	Spécifie s'il faut créer la clé lors de chaque installation et ce même si elle est vide.
Condition	Spécifie une condition qui doit être évaluée par `True` lors de l'installation afin que la clé soit créée. Consultez la section des conditions couverte précédemment pour plus de détails à ce sujet.
DeleteAtUninstall	Spécifie si la clé et ses sous-clés doivent être supprimées lors de la désinstallation de l'application.
FullPath	Retourne le chemin complet de la clé dans le registre *(lecture seulement)*.
Transitive	Spécifie si le logiciel d'installation doit réévaluer la propriété `Condition` de cette clé à chaque fois que l'application est installée ou désinstallée.

Pour **ajouter une nouvelle valeur**, sélectionnez la clé sous laquelle la nouvelle valeur devra être créée puis activez le menu `Action → Nouveau` ou le menu contextuel puis, au sein du menu, choisissez l'un des types suivants de valeurs :

Type de valeur	Description
Valeur de chaîne	Chaîne de caractères.
Valeur de la chaîne d'environnement	Chaîne de caractères contenant une référence vers une variable d'environnement devant être interprétée avant que la valeur ne soit retournée. Exemple : `%systemroot%\dossier`.
Valeur binaire	Données binaires de toutes formes que vous devrez exprimer octets par octets à l'aide des caractères ASCII correspondants.
Valeur DWORD	Valeur numérique entière sur 32-bits.

Lorsque vous sélectionnez une valeur, les propriétés suivantes sont disponibles :

Propriété de la valeur	Description
(Name)	Nom de la valeur devant être créée.
Condition	Spécifie une condition qui doit être évaluée par `True` lors de l'installation afin que la clé soit créée. Consultez la section des conditions couverte précédemment pour plus de détails à ce sujet.
Transitive	Spécifie si le logiciel d'installation doit réévaluer la propriété `Condition` de cette valeur à chaque fois que l'application est installée ou désinstallée.
Value	Spécifie les données à attribuer à la valeur lors de l'installation.
ValueType	Retourne le type de la valeur *(lecture seulement)*.

Section types de fichiers

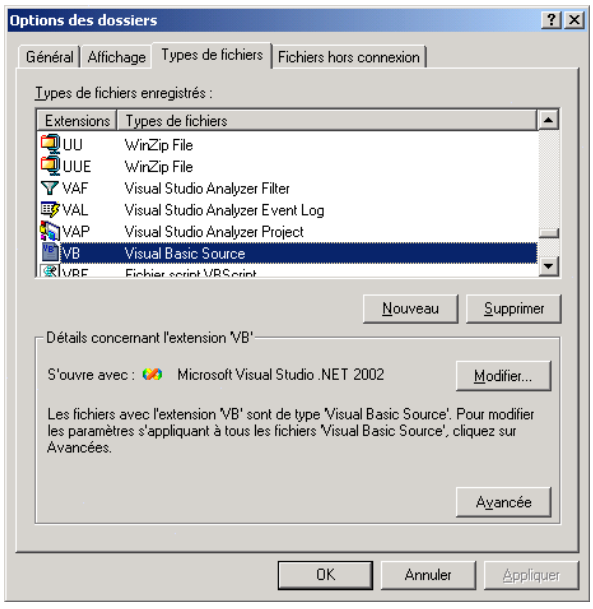
La section `Types de fichiers` vous permet de créer des associations entre votre application et une extension de fichiers personnalisée. Ainsi, une fois votre application installée, les fichiers possédant l'extension personnalisée se verront attribuer l'icône que vous leur aurez assigné et pourront lancer automatiquement l'exécution de votre application lorsqu'ils seront activés par l'utilisateur.

Pour **ajouter un nouveau type de fichier**, sélectionnez le nœud racine de l'arborescence puis activez le menu `Action → Ajouter un type de fichier` ou le menu contextuel. Un nouveau type de fichier s'affichera alors sous le nœud racine ainsi qu'une action `Open` sur laquelle nous reviendrons plus loin. Notez que celui-ci sera souligné en bleu – ceci indiquant une erreur qui provoquera l'échec de la création du logiciel d'installation – jusqu'à ce que l'ensemble des propriétés minimales soit spécifié.

Lorsque vous sélectionnez un type de fichier, les propriétés suivantes sont disponibles :

Propriété du type	Description
(Name)	Nom du type de fichiers. Ce nom est utilisé pour afficher le fichier à l'utilisateur au sein de l'éditeur des types de fichiers *(voir plus bas)*.
Command	Spécifie le fichier EXE de votre application qui sera exécuté lorsque ce type de fichiers sera activé par l'utilisateur. Cet exécutable doit prévoir le code nécessaire pour prendre en charge un tel type d'exécution.
Description	Spécifie une chaîne de caractères constituant une description détaillée du type de fichiers.
Extensions	Listes des extensions concernées par ce type de fichiers. Séparez les différentes extensions les unes des autres par un point-virgule (;).
Icon	Spécifie l'icône qui sera utilisé pour illustrer l'ensemble des fichiers concernés par les extensions listés par la propriété `Extensions`.
MIME	Spécifie un type MIME associé aux fichiers de ce type.

L'utilisateur pourra modifier les informations relatives à un type de fichier en activant le menu `Outils` de l'Explorateur de fichiers de *Windows* puis en sélectionnant l'onglet `Types de fichiers` au sein de la boîte de dialogue telle qu'affichée ci-contre.



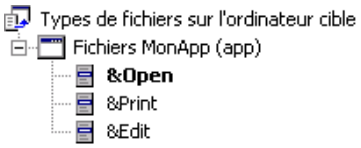
Les associations de fichiers sont stockées dans la clé `HKEY_CLASSES_ROOT` de la base de registre du poste utilisateur cible.

Les actions définissent les éléments affichés à l'utilisateur lorsque celui-ci active le menu contextuel sur un fichier du type spécifié.

Dans l'exemple ci-contre affichant le menu contextuel disponible lorsque l'utilisateur sélectionne un fichier PDF d'*Adobe Acrobat*, remarquez la présence de deux actions possibles situées dans le haut complètement du menu : `Ouvrir` et `Imprimer`.

Remarquez également qu'une action s'affiche en gras : il s'agit de l'action par défaut, action qui sera exécutée si l'utilisateur double-clique sur le fichier.

Pour **ajouter une nouvelle action** à un type de fichier personnalisé, sélectionnez le type de fichier pour lequel l'action devra être créée puis activez le menu `Action → Ajouter une action` ou le menu contextuel.



Lorsque plusieurs actions sont ajoutées pour un même type de fichiers comme illustré ci-contre, il est possible de modifier l'ordre dans lequel elles s'afficheront au sein du menu contextuel du fichier en activant le menu `Action → Monter` ou `Action → Descendre`. Il est également possible de **définir l'action par défaut** à l'aide du menu `Action → Par défaut`.

Lorsque vous sélectionnez une action, les propriétés suivantes sont disponibles :

Propriété de l'action	Description
(Name)	Nom de l'action telle qu'elle sera identifiée au sein du menu contextuel du fichier. Créez un raccourci en précédent la lettre désiré d'une esperluette (&).
Arguments	Spécifie les arguments de la ligne de commande qui seront transférés à l'exécutable spécifié par la propriété `Command` du type de fichiers. Le symbole %1 sera toujours remplacé par le nom et le chemin complet du fichier activé avant d'être transmis à l'exécutable associé.
Verb	Nom du verbe correspondant à l'action.

Pour permettre à votre application de récupérer l'argument passé par l'action, vous pouvez utiliser la commande `System.Environment.GetCommandLineArgs` qui retourne un tableau de chaînes de caractères dont chacun des éléments représente un argument de la ligne de commande. Notez que le premier élément du tableau contient toujours le nom et le chemin complet de votre exécutable.

Créer un raccourci activant la désinstallation de l'application

Quoique l'utilisateur puisse utiliser le panneau de configuration pour désinstaller l'application, il peut parfois être pratique de créer un raccourci au sein du menu `Démarrer → Programmes` du poste de l'utilisateur afin de lui permettre de réparer ou de désinstaller l'application.

Lorsque l'application est installée sur le poste client, les informations de désinstallation sont stockées au sein des clés de registre contenues sous la clé suivante :

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall
```

Une clé est créée à cet endroit et porte comme nom la valeur stockée au sein de la propriété `ProductCode` de votre projet de configuration. Généralement, ce code de produit devrait être constitué d'un *GUID* unique généré automatiquement par *Visual Studio.NET*. Or, une sous-clé nommée `UninstallString` contient l'application à exécuter lors de la désinstallation de l'application ainsi que le(s) paramètre(s) à lui passer en ligne de commande. Lors de la création de logiciels d'installation à l'aide de *Windows Installer*, cette clé possède la valeur `MsiExec.exe /I` suivi immédiatement du code de produit de l'application à désinstaller.

Par exemple, si vous ouvrez un projet de configuration et sélectionnez le projet au sein de l'explorateur de solutions, les propriétés du projet s'afficheront dans la fenêtre des propriétés. Localisez la propriété `ProductCode` et prenez en note le numéro qui y s'affiche, incluant les accolades le cas échéant. Si la valeur de cette propriété vaut par exemple `{3CB9294A-91F7-4FF0-8176-4493AB06D8D8}`, la commande permettant de désinstaller votre application est donc la suivante :

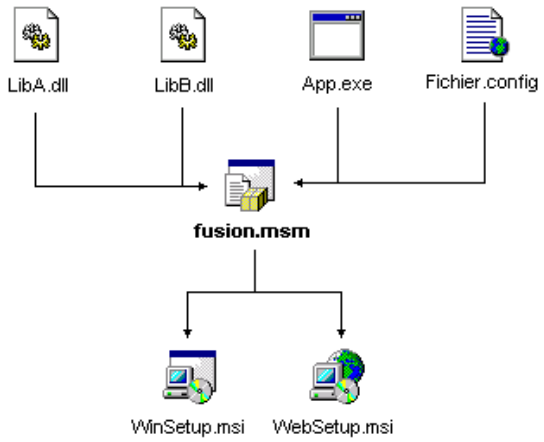
```
MsiExec.exe /I{3CB9294A-91F7-4FF0-8176-4493AB06D8D8}
```

Il ne vous plus donc à créer un raccourci permettant l'exécution de cette commande. Pour ce faire, ajoutez le dossier spécial `System` à l'aide de l'éditeur `Systèmes de fichiers`. Ensuite, ajoutez-y un nouveau fichier et localisez le fichier *`%system%`*`\msiexec.exe`. Le concepteur de projet de configuration vous soulignera une erreur en prétendant que le fichier est protégé et qu'il ne peut être copié. Faite fi de cette erreur et créez un raccourci pointant vers ce fichier. Dans les propriétés du raccourci, localisez la propriété `Arguments` et précisez-y la ligne de commande `/I{3CB9294A-91F7-4FF0-8176-4493AB06D8D8}`.

Lors de la création du logiciel d'installation, le fichier `msiexe.exe` sera automatiquement exclu du projet mais le raccourci permettant l'exécution de cet utilitaire sera tout de même créé et celui-ci précise la ligne de commande nécessaire à `msiexec.exe` pour désinstaller l'application spécifié.

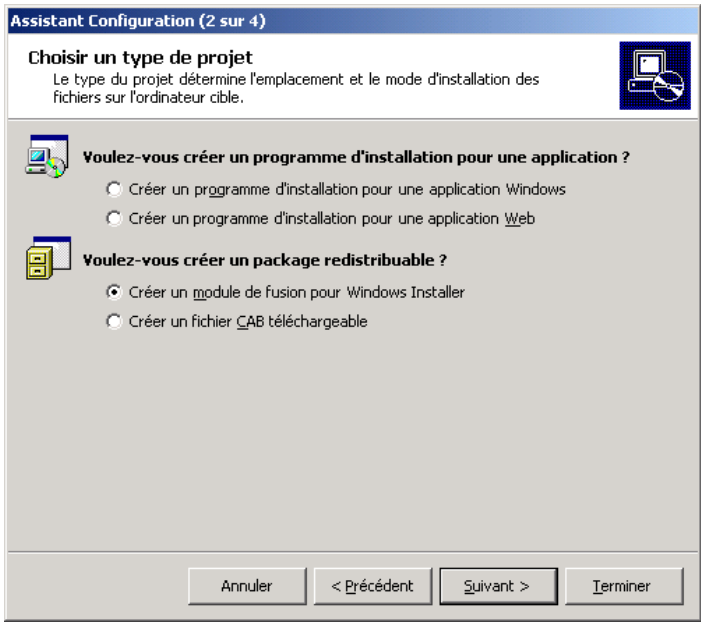
Projet de module de fusion

Le projet de module de fusion ne constitue pas en soit un logiciel d'installation qui pourra être exécuté sur le poste du client. Il s'agit plutôt d'un rassemblement de plusieurs assemblages ou fichiers au sein d'une même entité physique qui pourront ensuite être intégrés en bloc au sein d'un tiers projet de configuration pour applications *Windows* ou Web. Le projet de module de fusion est utile lorsque vous comptez déployer plusieurs applications qui intègrent un jeu commun d'assemblages.



Pour créer un projet de module de fusion, démarrez un nouveau projet et au sein de la boîte de dialogue vous invitant à préciser le type de projet à ajouter, sélectionnez la catégorie des **Projets de configuration et de déploiement**.

Si vous utilisez l'assistant Configuration afin de créer le projet de module de fusion, débutez en spécifiant le type de programme d'installation que vous désirez créer. En l'occurrence, créez un **module de fusion pour Windows Installer**.



Appuyez sur le bouton `Suivant` pour continuer. Vous serez alors invité à ajouter les assemblages et fichiers qui devront constituer le module de fusion. Notez que tout fichier pourra ultérieurement être ajouté au projet même si celui-ci avait préalablement été omis lors de l'exécution du présent assistant.

Appuyez sur le bouton `Suivant` afin de continuer puis terminer.

Une interface semblable à celle affichée dans la section des fichiers lors de la création d'un programme d'installation *Windows* s'affichera et vous permettra de créer les dossiers ajouter des dossiers spéciaux au sein desquels vous déposez ensuite les fichiers constituant le module de fusion. Générez ce dernier à l'aide du menu `Générer → Générer` *`NomProjet`*. Le module de fusion créé portera l'extension `.MSM`.

Installation de la plate-forme.NET sur le poste client

Installation de *Windows Installer* sur le poste client

Nous