Déploiement et monitoring

Sommaire

1 Introduction

L'étape finale de la conception d'une application Web est le déploiement de celle-ci. Le déploiement consiste à mettre en place l'ensemble de nos fichiers nécessaire au fonctionnement de notre application. Au cours de ce chapitre nous verrons comment créer un installateur automatisé afin de publier notre site Web. Nous étudierons la mise en place d'un service de monitoring permettant de consulter les divers évènements que notre application aura levés ainsi que le procédé de mise en cache.

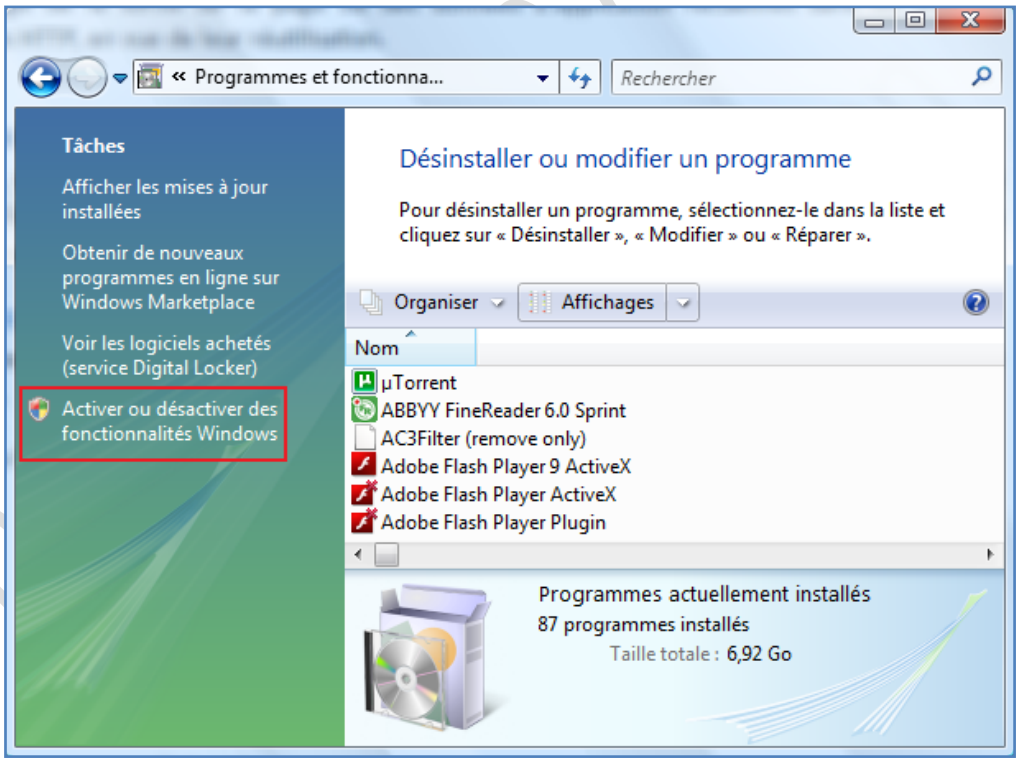
2 Configuration de IIS pour l'environnement ASP.NET

Bien que sous les systèmes Windows Server 2008, IIS soit activé par défaut, certains modules ont besoins d'être ajouté, de plus si vous travaillez sous Windows Vista vous aurez besoin d'installer et d'activer vous-même tous ces services. Nous allons voir ensemble comment effectuer ces opérations.

2.1 Activer IIS

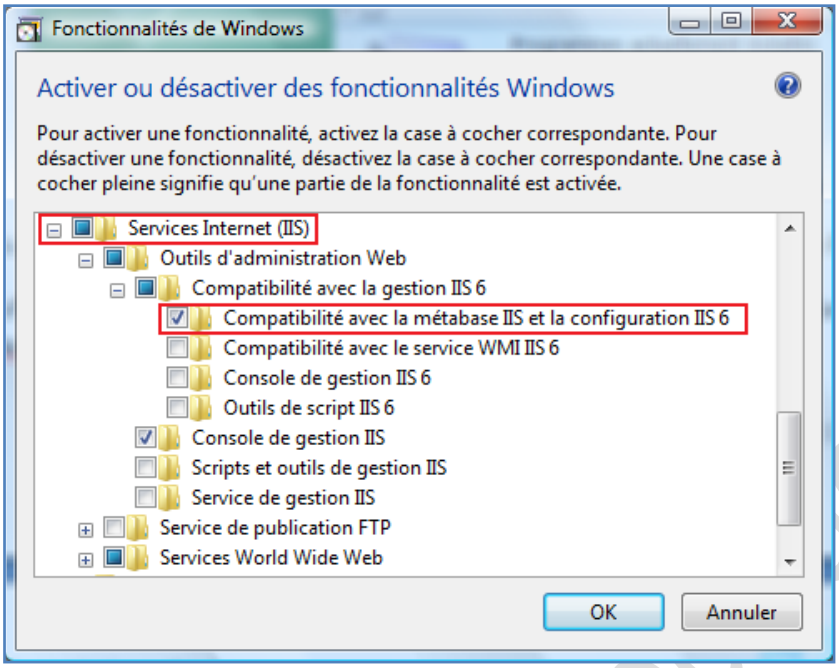
En premier lieu nous allons commencer par installer ce service compris dans toutes les versions de Windows.

Allez dans le *Panneau de configuration* puis choisissez *Programmes et fonctionnalités*. Dans la partie gauche de la fenêtre sélectionnez *Activer ou désactiver des fonctionnalités Windows* :



Une fois la fenêtre ouverte et chargée, cochez *Services Internet (IIS)* ouvrez ensuite l'arborescence :

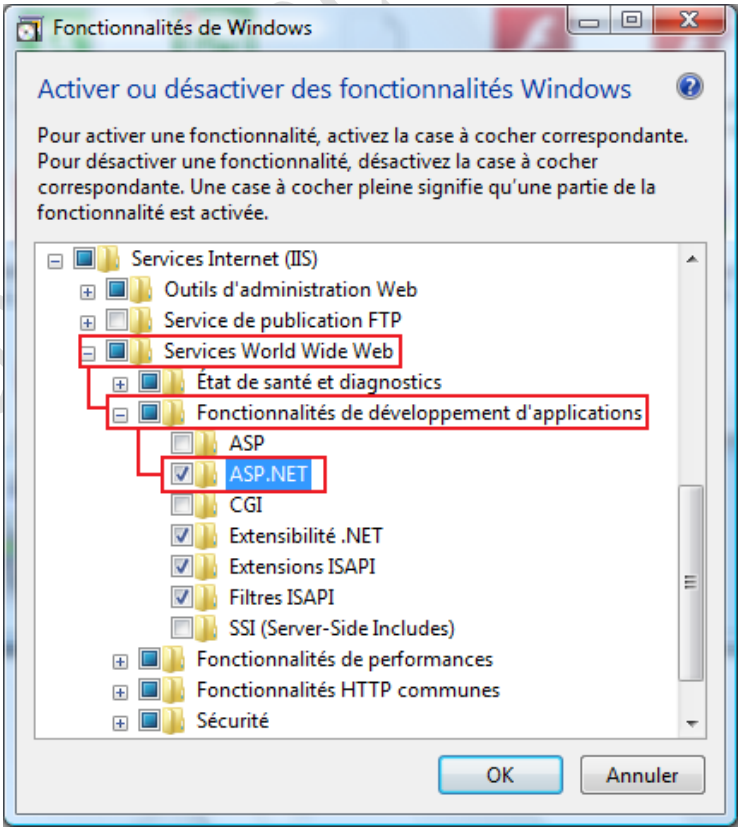
Outils d'administration Web → *Compatibilité avec la gestion IIS 6* → Cochez *Compatibilité avec la métabase IIS et la configuration IIS 6*.



Ne validez pas pour le moment, nous passons maintenant à l'environnement .Net.

2.2 Prise en charge de l'environnement ASP.NET

Toujours dans cette même fenêtre ouvrez l'arborescence de *Services World Wide Web*, puis développez *Fonctionnalités de développement et d'applications*. Cochez *ASP.NET*, d'autres composant se cocherons d'eux même.



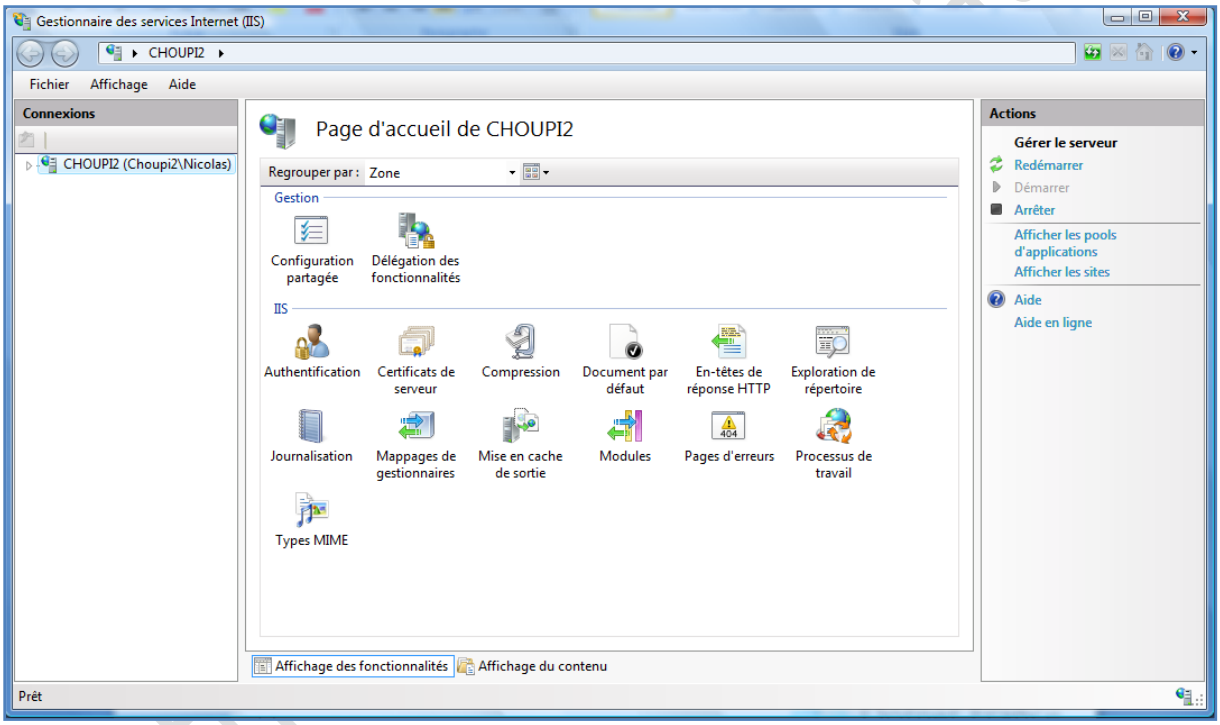
Cliquez sur OK et attendez la fin de l'installation. Soyez patient celle-ci peut durer plusieurs minutes.

2.3 Création d'un répertoire virtuel dédié à votre application

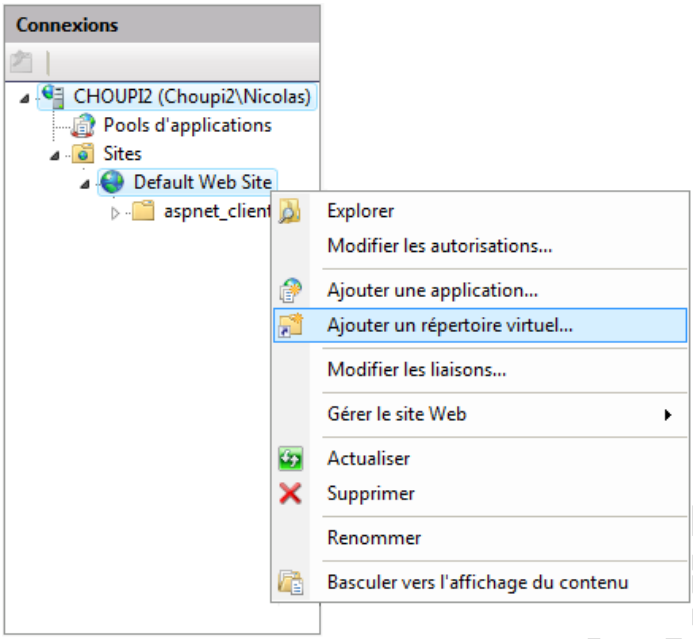
Une fois ces fonctionnalités installées nous allons procéder à la création d'un répertoire virtuel. Si les services Internet (IIS) sont exécutés sur votre ordinateur local, vous pouvez créer une racine virtuelle IIS qui pointe vers les fichiers d'un emplacement situé en dehors de la structure des dossiers IIS par défaut (en général *\Inetpub\wwwroot*). Un répertoire virtuel est utile lorsque vous souhaitez exécuter des pages dans un dossier existant ou dans un site Web de système de fichiers local.

En règle général un répertoire virtuel est automatiquement crée lors du déploiement de notre application Web. Nous allons cependant expliquer comment le faire manuellement.

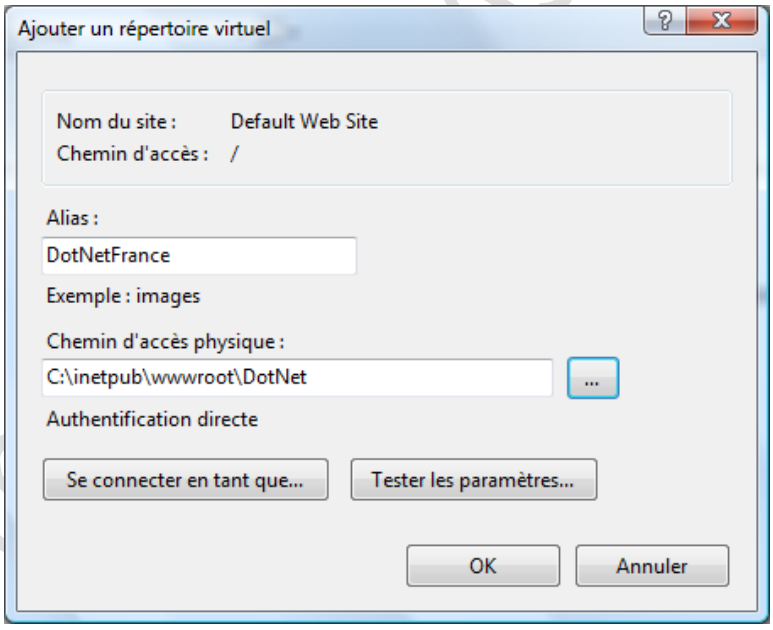
Revenez au *Panneau de configuration* puis sélectionnez *Outils d'administration*. Ouvrez maintenant *Gestionnaire des services Internet (IIS)*. Vous accéder à une interface de gestion telle que le montre l'image ci-dessous :



Dépliez l'arborescence de votre ordinateur dans la partie connexions, faites clic droit sur votre site Web puis *Ajouter un répertoire virtuel…* comme le montre l'exemple ci-dessous.



Une nouvelle fenêtre apparait vous permettant de créer ledit répertoire, il est conseillé de choisir le chemin physique dans *C:\intetpub\wwwroot* chemin par défaut des sites Web de IIS, pour des soucis d'autorisations, puis d'y créer le répertoire de votre choix.



Cliquez sur OK pour finaliser la création de votre répertoire virtuel.

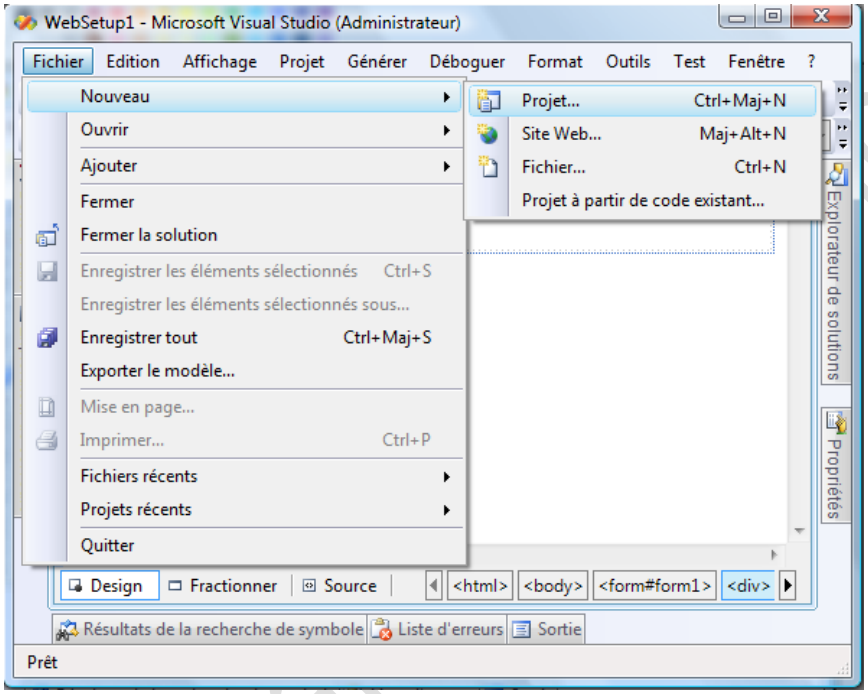
3 Le déploiement d'une application Web

Pour déployer une application, plusieurs étapes sont nécessaires et, en fonction des paramètres à prendre en compte, le temps pour effectuer ce déploiement sera plus ou moins long. Au cours de cette leçon nous verrons comment créer un fichier d'installation automatisé qui nous permettra de déployer notre application Web rapidement.

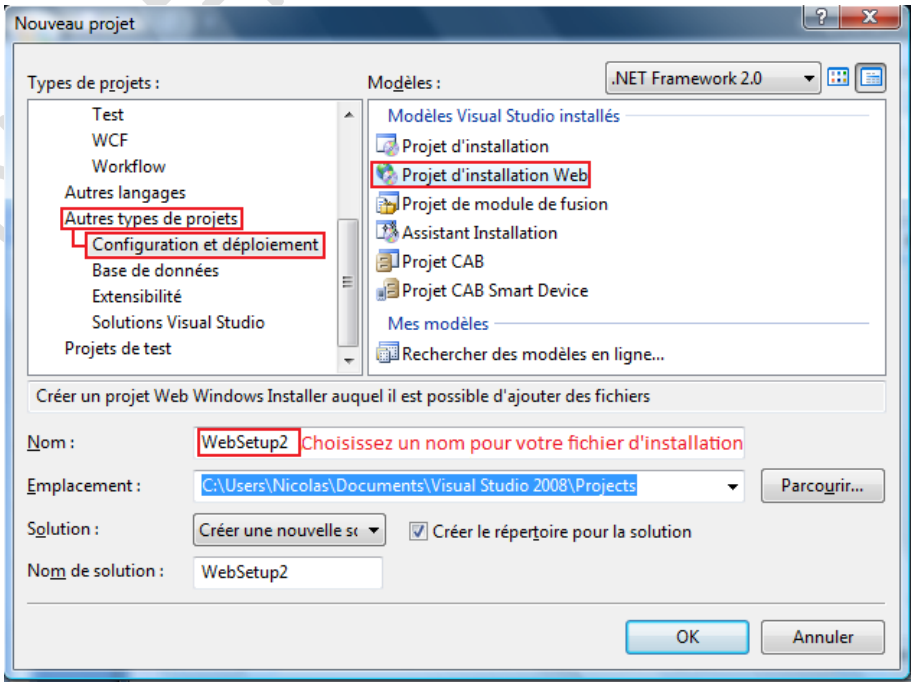
3.1 Déploiement d'une application simple

La création d'un fichier d'installation est une étape relativement facile. Dans le cas présent nous effectuerons un déploiement simple sans base de données à intégrer.

Pour ce premier exemple commencer par créer une application Web comme à votre habitude dans le langage de votre choix (C# ou VB), ne faisant appel à aucune base de données. Un Label et une Textbox dans votre page Default.aspx feront l'affaire. Sauvegardez votre solution puis créez un nouveau projet :

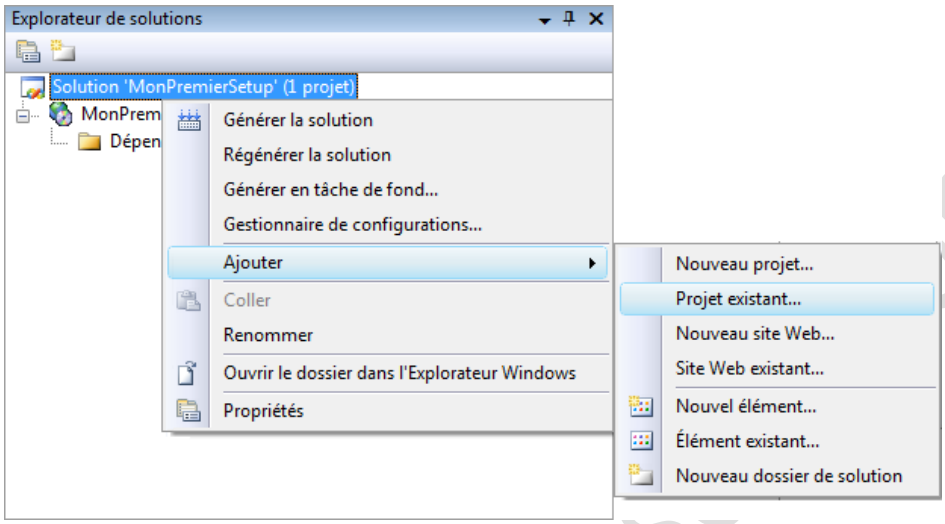


Choisissez *Autres Type de Projet* puis *Configuration et déploiement*. Puis sélectionnez comme modèle *Projet d'installation Web*.

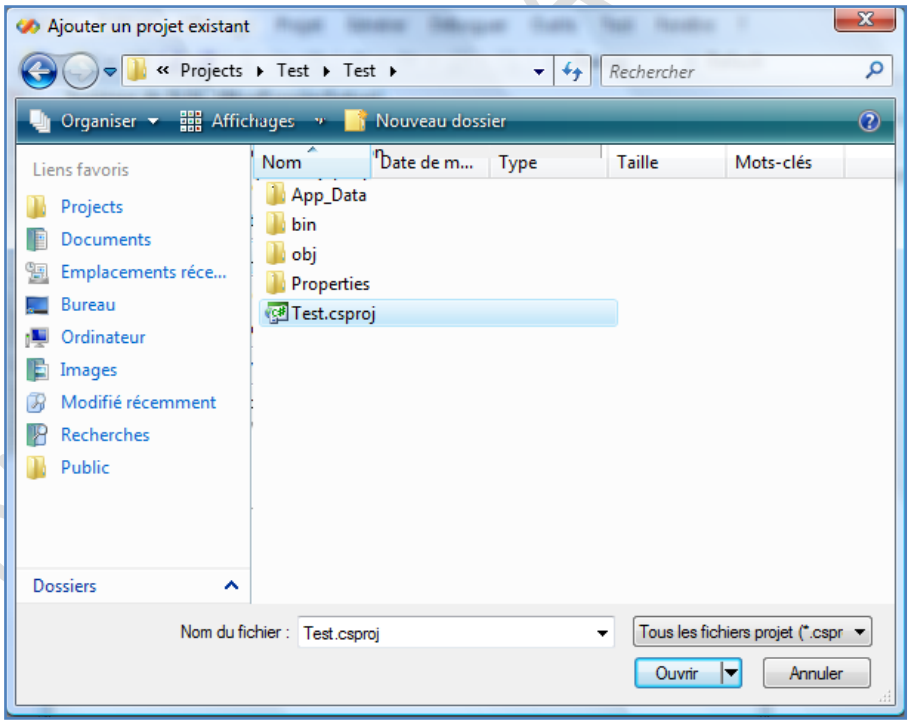


Cliquez sur OK pour générer votre projet de déploiement.

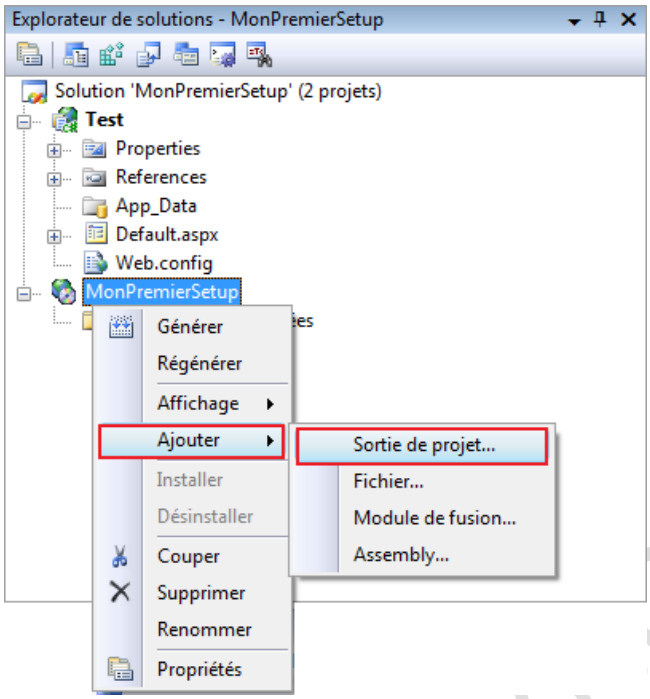
Nous allons maintenant lier notre projet de déploiement à un projet existant. Pour cela, dans l'explorateur de votre solution de projet de déploiement, faites un clic droit sur le nom de votre solution puis faites *Ajouter → Un projet existant*



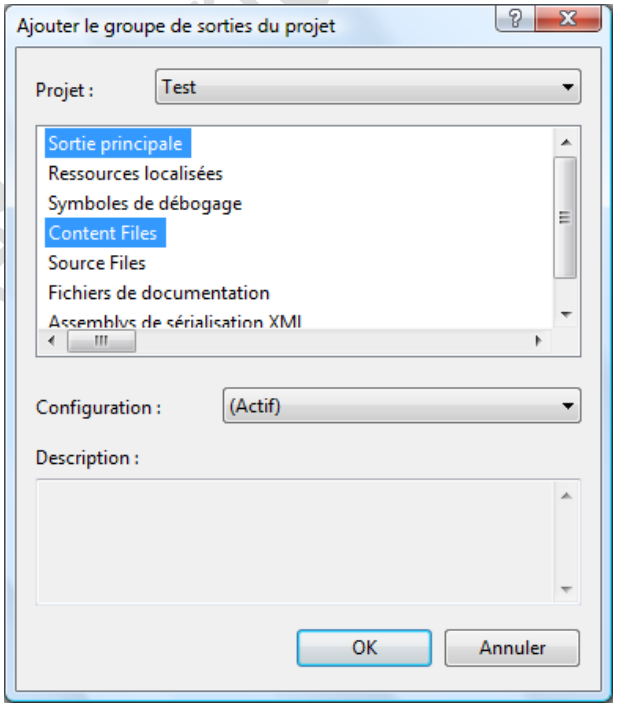
Choisissez le projet que vous voulez déployer en récupérant le ficher *.vbproj* ou *.csproj* de la solution.



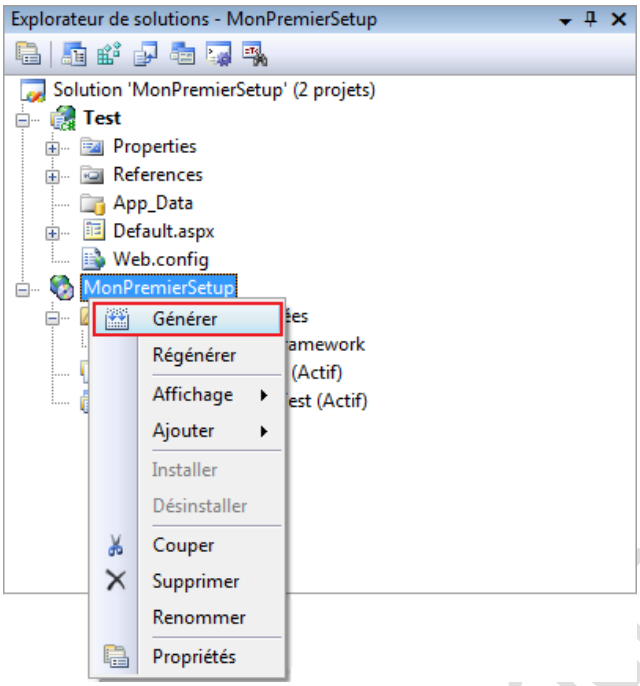
Cliquez sur Ouvrir pour lier votre solution à votre projet de déploiement. Désormais votre application Web apparait dans l'explorateur de solution. Cela dit, rien n'est encore fait, il faut préciser ce que l'on veut que notre fichier d'installation copie sur le serveur IIS. Pour cela faite un clic droit sur le nom de votre Projet d'installation puis *Ajouter → Sortie de fichier…*



Sélectionnez *Sortie principale* et *Content File* pour ajouter les *Assemblies* ainsi que les fichiers sources de votre application. Notez qu'il est important de ne pas modifier votre projet à installer une fois chargé dans votre projet d'installation, sous peine de ne pas pouvoir lier nos bases de données locale (Exemple : fichier.mdf).

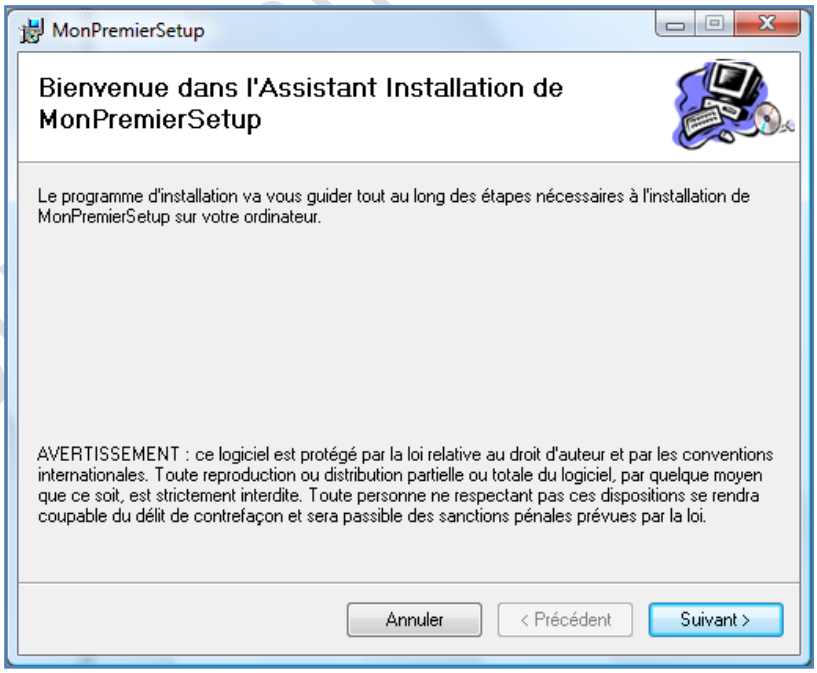


Cliquez sur OK. Il ne reste plus qu'à compiler le tout et générer un fichier d'installation. Faites un clic droit sur votre projet d'installation puis sélectionnez générer.



Vous pouvez désormais tester l'installation de votre application. Pour cela vous refaites un clic droit sur votre projet d'installation et cliquez sur *Installer* qui sera maintenant dégrisé. Vous pouvez tout aussi bien allez récupérer votre fichier d'installation dans votre répertoire de sauvegarde de vos projet et l'exécuter.

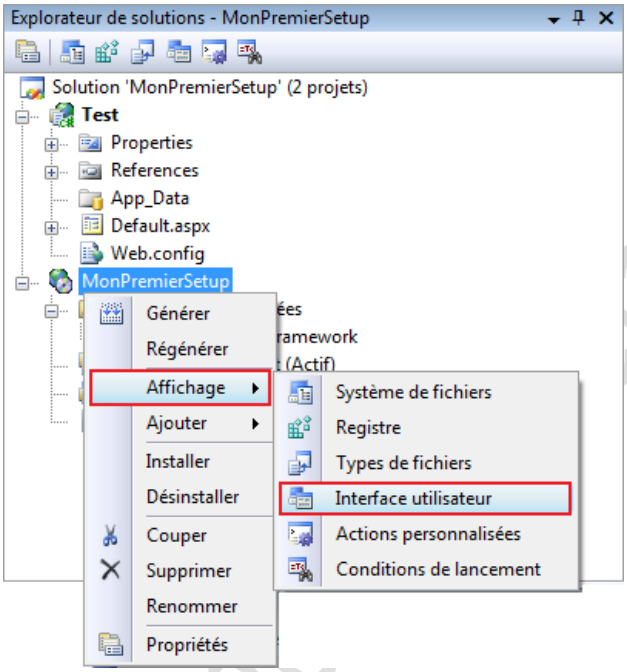
Voici un aperçu de votre assistant d'installation généré :



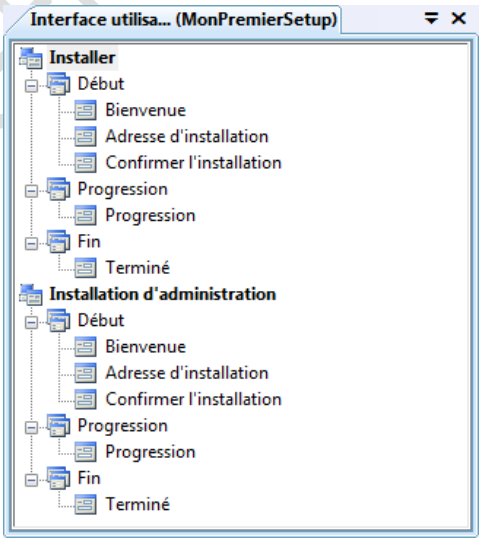
3.2 Personnaliser l'assistant d'installation

Au cours du déploiement de votre application vous pouvez paramétrer l'interface utilisateur afin d'afficher un contrat de licence ou bien demander un numéro de série. Nous allons voir comment créer ces interfaces dans cette partie.

Pour commencer nous allons voir comment afficher l'éditeur d'interface utilisateur. Faites un clic droit sur votre projet de Setup puis Afficher, Interface Utilisateur.



Vous pouvez désormais consulter les différentes parties de la procédure d'installation de votre site Web dont voici un aperçu.



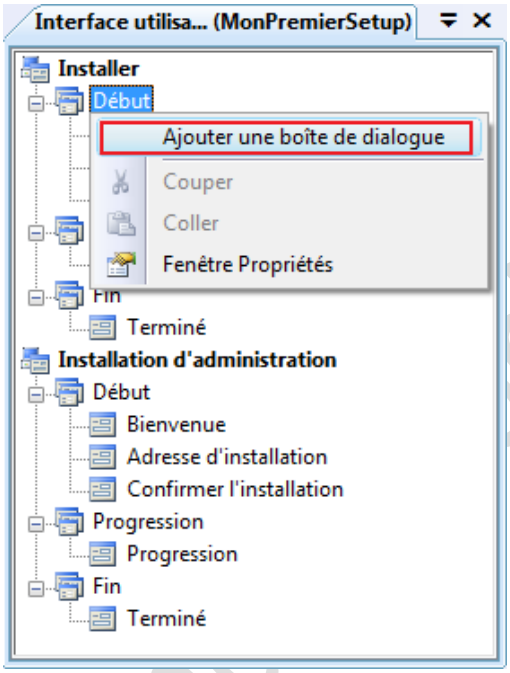
Nous pouvons constater la présence de deux parties distinctes :

- **Installer** : cette partie est celle utilisé pour une installation classique.
- **Installation d'administration** : utilisé pour une installation en mode remote (installation par l'intermédiaire d'un partage réseau).

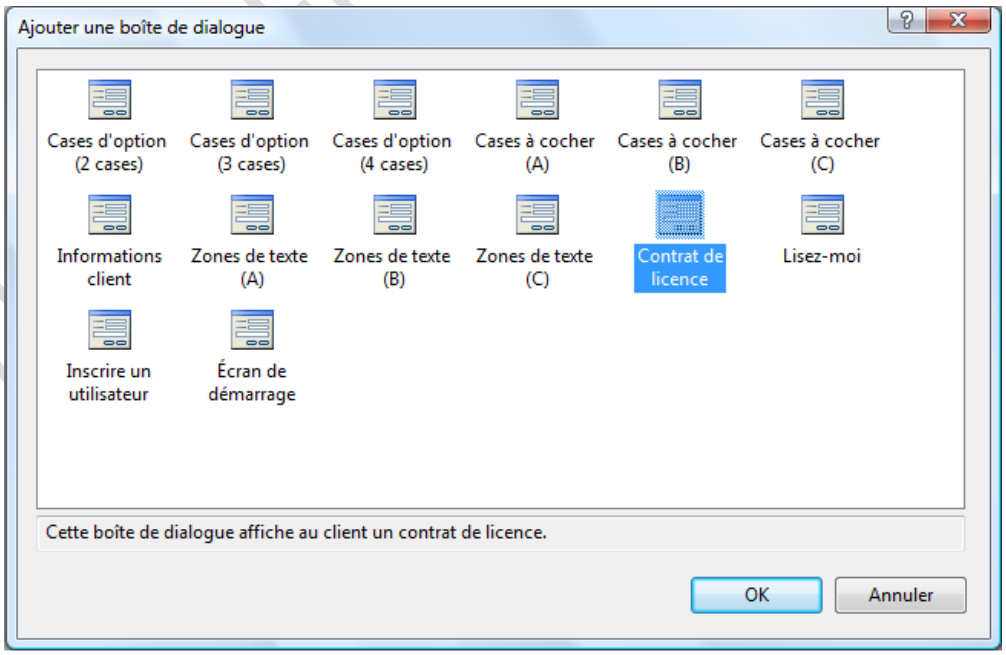
Dans cette partie nous ne verrons que l'installation classique car la plus courante. Ainsi que les fonctionnalités les plus utilisées.

3.2.1 Ajouter un contrat de licence

Un contrat de licence est un écran présentant la licence du logiciel concerné, l'utilisateur est appelé à accepter ou à refuser cette licence. Pour ajouter un contrat de licence faites clic droit sur *Début*, puis *Ajouter une boite de dialogue*.



Une nouvelle fenêtre apparait alors vous demandant de choisir quelle boite de dialogue vous désirez ajouter. Nous choisirons *Contrat de licence*.

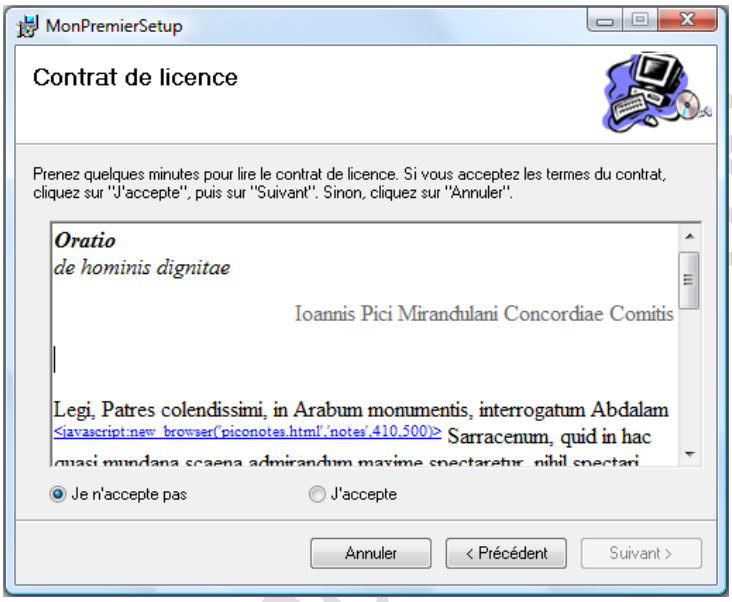


Un contrat de licence se place généralement juste après l'écran de *Bienvenue*, déplacez-le si nécessaire.

Un contrat de licence dispose de trois propriétés :

- **BannerBitmap** : Affiche une bannière cet écran d'installation, prend en paramètre un fichier Bitmap.
- **LicenseFile** : Prend en paramètre un fichier.rtf, fichier texte qui sera affiché dans le conteneur de licence de l'installateur. Les mises en forme de votre texte sont conservées.
- **Sunken** : Attend un booléen, affiche ou non les bordures du conteneur de licence.

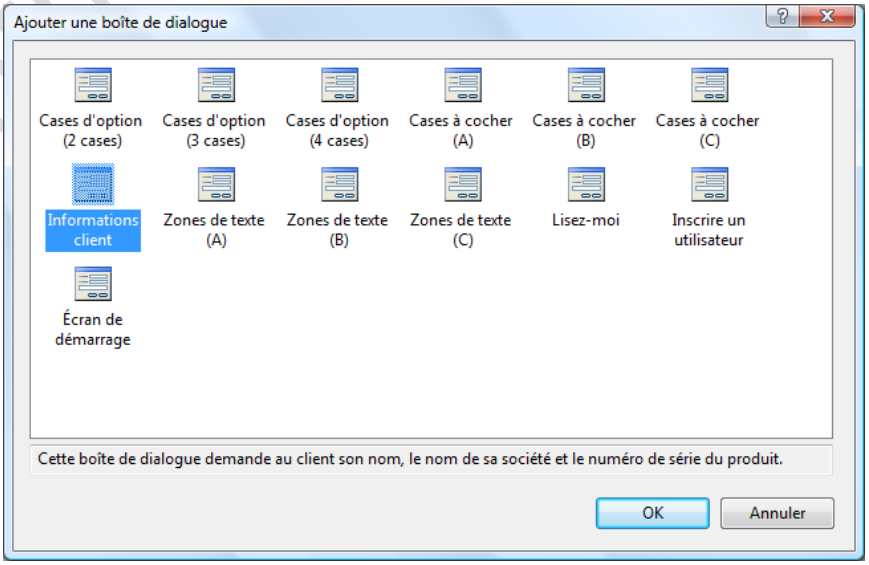
Une fois ces propriétés passées en revus voici un aperçu de notre écran de *Contrat de licence*.



3.2.2 Ajouter une gestion de numéro de série

Ajouter un numéro de série permettra à votre application de se protéger contre le piratage. L'utilisateur doit entrer une clé valide pour pouvoir poursuivre l'installation de votre application. La clé entrée est soumise à un algorithme que nous expliquerons plus tard en détail.

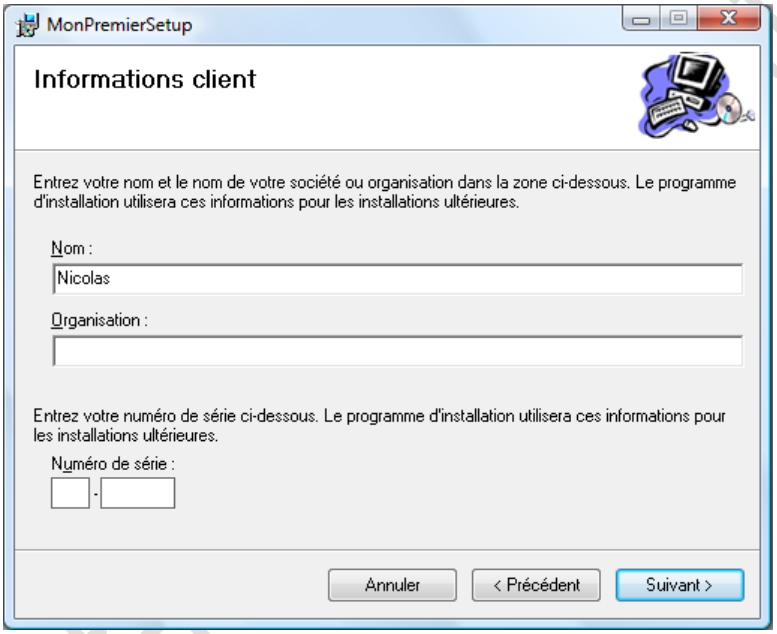
Tout d'abord nous devons ajouter une boite de dialogue *Informations Clients* de la même façon que pour le *Contrat de licence*.



Par convention, il faut mettre cette boite de dialogue juste après le contrat de licence. Voici les propriétés de cette boite de dialogue :

- **BannerBitmap** ;
- **SerialNumberTemplate** : Détermine le modèle que doit suivre l'algorithme pour vérifier la clé de validation ;
- **ShowOrganisation** : Prend un booléen et affiche ou non dans la boite de dialogue des champs demandant à l'utilisateur d'entrer ses informations ainsi que la compagnie pour laquelle il travaille ;
- **ShowSerialNumber** : Prend un booléen et affiche ou non un champ permettant d'entrer le numéro de série (false par défaut).

Voici un aperçu lors de l'installation lorsque la propriété *ShowSerialNumber* est à *True* :



L'algorithme du numéro de série :

Comme vous avez pu le constater nous n'avons jamais entré de numéros de séries valide nous même, et nous n'auront pas à le faire. En effet, c'est grâce à la propriété *SerialNumberTemplate* que nous allons déterminer un modèle que va devoir suivre le numéro de série à entrer.

L'algorithme de validation fonctionne de la façon suivante, il additionne tous les chiffres que vous avez choisis de prendre en compte, il divise le résultat par 7, le reste de la division doit être 0 sinon le numéro de série est refusé.

Voici un tableau donnant les caractères possibles à utiliser :

#	Exige un chiffre qui n'est pas inclus dans l'algorithme de validation.
%	Exige un chiffre qui est inclus dans l'algorithme de validation.
?	Exige un caractère alphanumérique qui n'est pas inclus dans l'algorithme de validation.

^	Exige un caractère majuscule ou minuscule. Les chiffres ne sont pas valides ici.
<	Tout caractère situé à gauche de ce caractère ne s'affiche pas dans la boîte de dialogue.
>	Tout caractère situé à droite de ce caractère s'affiche dans la boîte de dialogue. Obligatoire comme caractère de fin en cas d'utilisation du caractère <.

Exemple :

Prenons le *template* suivant : <#### - %%%% - ^^??>

- Le **–** est un séparateur entre les champs texte, vous pouvez l'encadrer d'espace ou non.
- Les quatre **#** signifient que l'utilisateur devra rentrer quatre chiffres qui seront ignorés par l'algorithme.
- Les quatre **%** signifient que l'utilisateur devra rentrer quatre chiffres qui seront contrôlés par l'algorithme de validation.
- Chacun des deux **^** signifient que l'utilisateur devra rentrer un caractère majuscule ou minuscule.
- Les deux **?** signifient que l'utilisateur devra rentrer deux caractères alphanumériques qui seront ignorés par l'algorithme.

Voici une clé entrée par un utilisateur pour ce *template* : **1234 – 3459 – Am1L**

- ➔ **1**, **2**, **3** et **4** sont bien des chiffres.
- ➔ (**3**+**4**+**5**+**9**) mod 7 = 21 mod 7 = 0 la somme des caractères pris en compte par l'algorithme est bien un multiple de 7.
- ➔ **A** est bien un caractère majuscule et **m** un caractère minuscule.
- ➔ **1** et **L** sont bien deux caractères alphanumériques.

La clé est donc valide.

3.2.3 Les autres boites de dialogue

Vous avez pu constater la présence d'autres boites de dialogue que vous avez la possibilité d'ajouter : Lisez-moi, Inscrire un utilisateur, Case d'option, Zone de texte, etc. Dans les cas précédents nous avons pu constater que les propriétés de ces boites de dialogue été relativement intuitive, et c'est toujours le cas pour la boite de dialogue Lisez-moi par exemple.

Cela dit certaines d'entre elles doivent être liées avec des actions personnalisées. Par exemple vous désirez enregistrer les informations personnelles de l'utilisateur à l'installation afin de personnaliser l'affichage de l'application ou bien configurer l'accès à une base de données en modifiant le fichier web.config.

Dans la section suivante nous allons voir comment créer des actions personnalisées.

3.3 Déploiement d'une application Web Avancée

Nous venons de voir qu'il existe certaines boites de dialogue facile à gérer, ceci étant il est parfois nécessaire d'effectuer un déploiement qui requiert une configuration de la part de l'utilisateur. Par exemple si nous avons plusieurs projets ajoutés nous pouvons lui donner la possibilité de ne choisir qu'un seul d'entre eux.

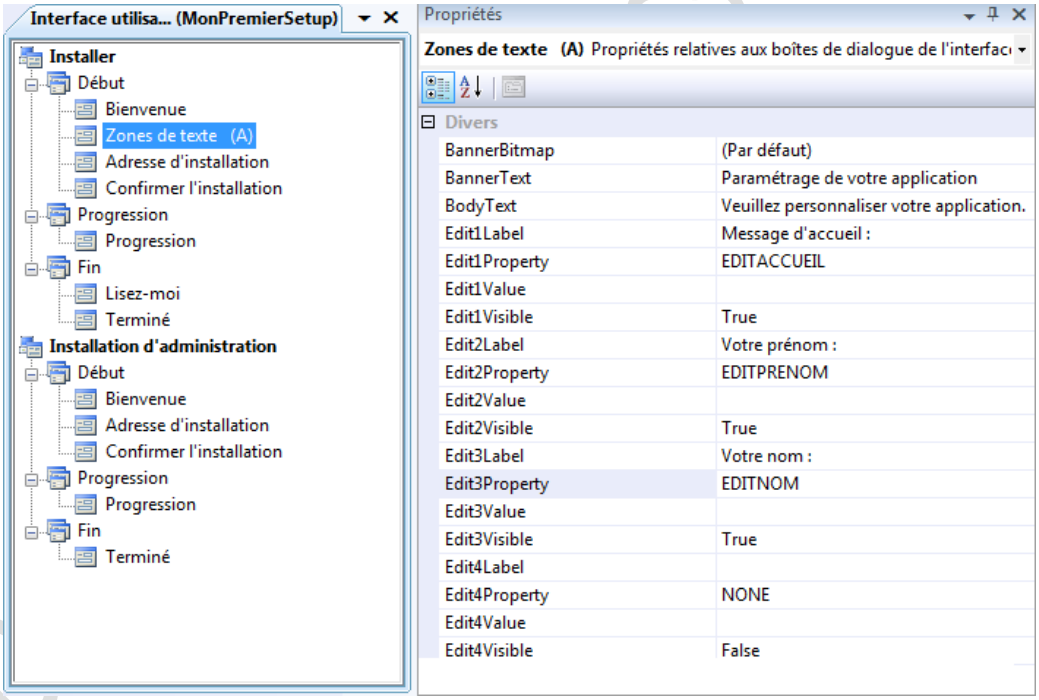
3.3.1 Les actions personnalisées

Les actions personnalisées sont des scripts et ou bien des exécutables que nous aurons crées et dont nous devrons définir le but.

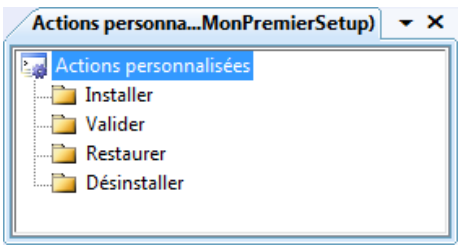
Notre but sera de modifier le fichier web.config en fonction des entrées de l'utilisateur au cours du déploiement. Dans l'exemple que nous allons suivre, nous enregistrerons des variables globales dans le web.config.

Pour commencer nous allons ajouter une boite de dialogue à l'interface utilisateur de type « Zone de texte (A) » juste après « Bienvenue ». Supposons qu'on laisse à l'utilisateur la possibilité de personnaliser le message d'accueil de son application et d'en définir le propriétaire.

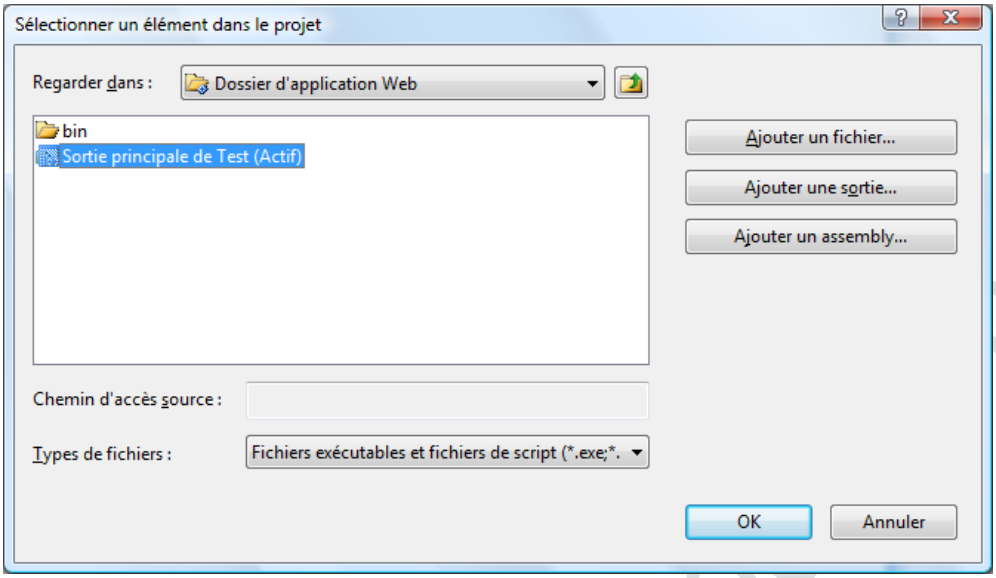
Pour la configuration de la boite de dialogue veuillez suivre l'exemple ci-dessous :



Une fois notre zone de texte ajoutée et configurée nous allons permettre la création d'action personnalisée. Pour cela faite un clic droit sur votre projet de déploiement, puis Afficher, Actions personnalisées. Vous accédez à une interface de ce type :



Sur chacun des dossiers Installer, Valider, Restaurer, Désinstaller, faites un clic droit puis Ajouter une action personnalisée. Sélectionnez la Sortie principale de votre projet (Actif) :



Dans chacune de vos actions personnalisées vous disposez d'une propriété *CustomActionData* qui va nous permettre de créer des variables contenant les entrés de l'utilisateur lors du déploiement utilisable dans le script d'installation (que nous allons voir juste après).

Voici la syntaxe à utiliser pour créer ces variables :

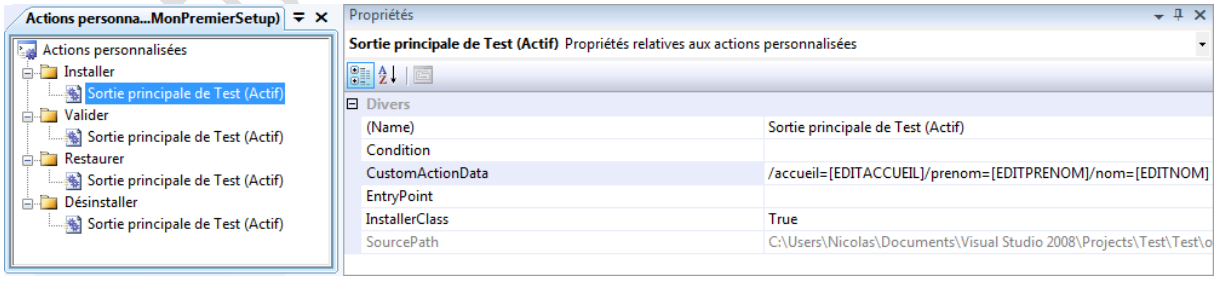
```
/variable1=[NomDeVotreEdit1Property] /variable2=[NomDeVotreEdit2Property]
```

Il ne faut pas oublier de mettre le **/** devant chaque nouvelle variable ainsi qu'un espace entre chacune d'elle, il faut aussi que votre nom d'*EditProperty* soit en majuscule.

Voici ce qu'il vous rentrer dans cette propriété pour les besoins de l'exemple :

```
/accueil=[EDITACCUEIL] /prenom=[EDITPRENOM] /nom=[EDITNOM]
```

N'oubliez pas d'effectuer cette opération pour chaque action personnalisée.



Nous avons maintenant la possibilité de récupérer les entrées utilisateurs et de les gérer dans un script d'installation.

Rappelons que le but de notre exemple est d'ajouter des variables globales à notre fichier web.config. Dans ce but nous allons créer une section dédiée à cette utilisation.

XML du fichier web.config

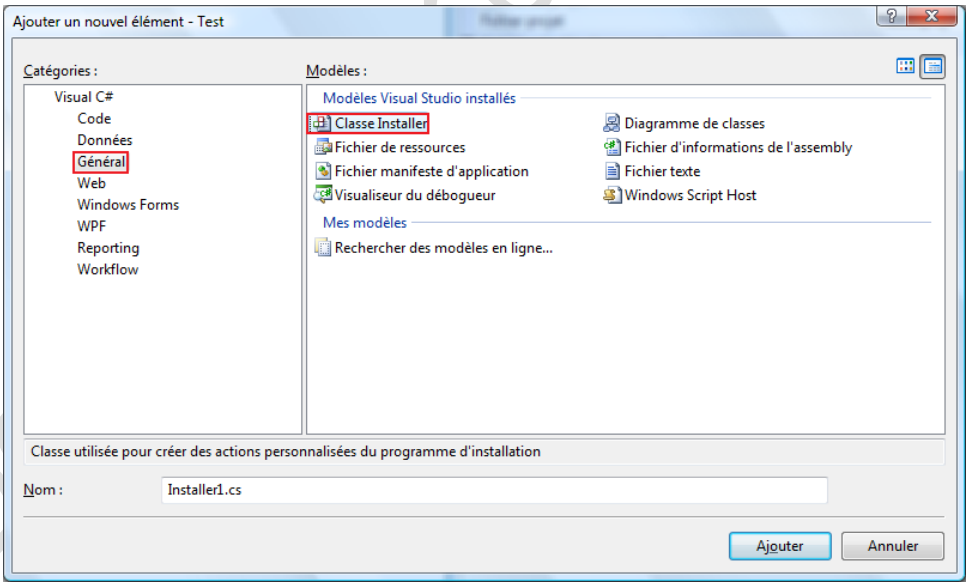
```
<?xml version="1.0"?>
<configuration>

  <appSettings>
    <add key="Accueil" value="" />
    <add key="Prenom" value="" />
    <add key="Nom" value="" />
  </appSettings>

  <system.web>
    <compilation debug="true">
    </compilation>
    <authentication mode="Windows"/>
  </system.web>
</configuration>
```

La partie en gras est celle qui nous interesse, à adapter en fonction de vos besoins pour votre application Web.

3.3.2 La classe *Installer*

La classe *Installer* est une classe que nous allons ajouter à notre projet d'application Web (et non celui de déploiement) qui va nous permettre d'exécuter des actions personnalisées au cours du déploiement de notre application Web.

Pour ajouter une classe *Installer* faite un clic droit sur votre projet d'application Web puis Ajouter → Nouvel élément et choisissez *Classe Installer* puis validez.



Il ne nous reste plus qu'à éditer notre classer *Installer1*. Pour ce faire, nous sommes obligés de surcharger la méthode Install.

Tout d'abord voici les *namespaces* à ajouter :

- `System.IO;`
- `System.ComponentModel;`
- `System.Reflection;`
- `System.Xml.`

Et voici le code de la classe Installer.

C# de Installer1.cs

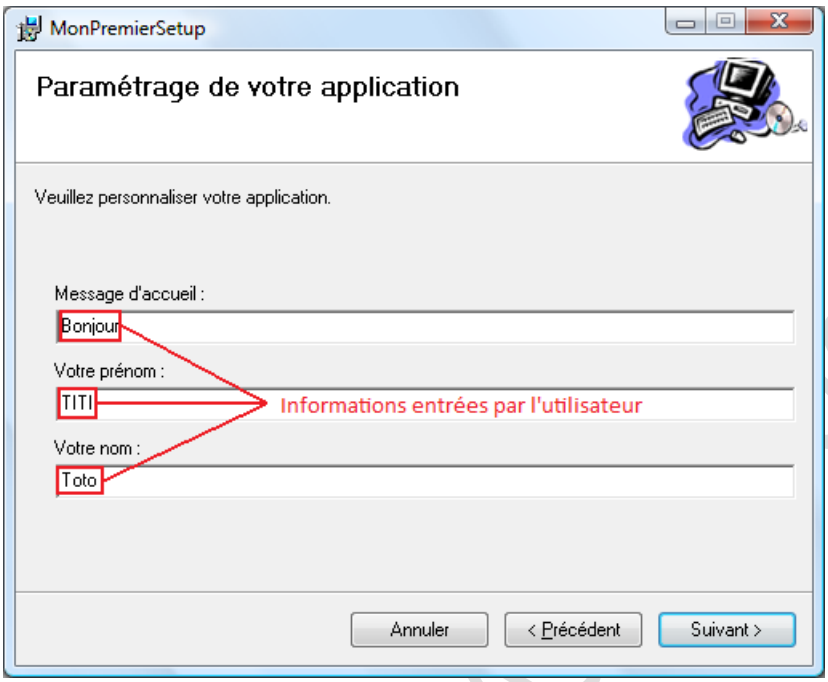
```
public override void Install(IDictionary stateSaver)
{
    base.Install(stateSaver);
    // On commence par récupérer les variables entrées pas l'utilisateur.
    // Notez qu'on utilise les variables déclarées dans les actions personnalisées.
    String txtAccueil = this.Context.Parameters["Accueil"];
    String txtPrenom = this.Context.Parameters["Prenom"];
    String txtNom = this.Context.Parameters["Nom"];
    // Nous allons récupérer l'assembly courrante afin de récupérer le chemin du fichier
     Web.config.
    Assembly Asm = Assembly.GetExecutingAssembly();
    String chemin;
    chemin = Asm.Location.Substring(0, Asm.Location.Length - Asm.GetName().Name.Length - 8) +
              "\\Web.config";
    // Le FileInfo va nous permettre de voir que le fichier Web.config existe bel et bien.
    FileInfo FileInfo1 = new FileInfo(chemin);
    if (!FileInfo1.Exists)
    {
        throw new InstallException("Impossible de trouver le fichier de configuration.");
    }
    // Nous allons charger notre web.config dans un objet XmlDocument afin de le parcourir
    //et de pouvoir le modifier par la suite.
    XmlDocument XmlDocument1 = new XmlDocument();
    XmlDocument1.Load(FileInfo1.FullName);
    // Les booléens suivant vont nous permettre de lever des erreurs personnalisées.
    Boolean FoundAccueil = false;
    Boolean FoundPrenom = false;
    Boolean FoundNom = false;
    // On parcours notre section appSettings de notre web.config contenue dans le
     XmlDocument1.
    foreach (XmlNode NodeFound in XmlDocument1["configuration"]["appSettings"])
    {
        // On recherche les Node ayant pour nom "add" puis en fonction de la valeur de
       l'attribut
        //"key" nous allons insérer les entrées de l'utilisateur adequate dans l'attribut
       "value".
        if (NodeFound.Name == "add")
        {
            if (NodeFound.Attributes["key"].Value == "Accueil")
            {
                NodeFound.Attributes["value"].Value = txtAccueil;
                FoundAccueil = true;
            }
            if (NodeFound.Attributes["key"].Value == "Nom")
            {
                NodeFound.Attributes["value"].Value = txtNom;
                FoundNom = true;
            }
            if (NodeFound.Attributes["key"].Value == "Prenom")
            {
                NodeFound.Attributes["value"].Value = txtPrenom;
                FoundPrenom = true;
            }
        }
    }
    // Si un de nos booléens a sa valeur à false on lève une exception.
    if (!FoundAccueil)
    {
        throw new InstallException("Impossible de trouver la clé Accueil");
    }
    if (!FoundNom)
    {
        throw new InstallException("Impossible de trouver la clé Nom");
    }
    if (!FoundPrenom)
    {
        throw new InstallException("Impossible de trouver la clé Prenom");
    }
    // Pour finir on sauvegarde notre XmlDocument1 sur le web.config
    XmlDocument1.Save(FileInfo1.FullName);
}
```

VB.NET de Installer1.cs

```
Public Overrides Sub Install(ByVal stateSaver As System.Collections.IDictionary)
    MyBase.Install(stateSaver)
    ' On commence par récupérer les variables entrées pas l'utilisateur.
    ' Notez qu'on utilise les variables déclarées dans les actions personnalisées.
    Dim txtAccueil As String = Me.Context.Parameters.Item("Accueil")
    Dim txtPrenom As String = Me.Context.Parameters.Item("Prenom")
    Dim txtNom As String = Me.Context.Parameters.Item("Nom")
    ' Nous allons récupérer l'assembly courrante afin de récupérer le chemin du
    fichier Web.config.
    Dim Asm As System.Reflection.Assembly =
    System.Reflection.Assembly.GetExecutingAssembly
    Dim chemin As String
    chemin = Asm.Location.Substring(0, Asm.Location.Length - Asm.GetName.Name.Length
    - 8) + "\Web.config"
    ' Le FileInfo va nous permettre de voir que le fichier Web.config existe bel et
    bien.
    Dim FileInfo As System.IO.FileInfo = New System.IO.FileInfo(chemin)
    If Not FileInfo.Exists Then
        Throw New InstallException("Impossible de trouver le fichier de
        configuration !")
    End If
    ' Nous allons charger notre web.config dans un objet XmlDocument afin de le
    parcourir
    'et de pouvoir le modifier par la suite.
    Dim XmlDocument As New System.Xml.XmlDocument
    XmlDocument.Load(FileInfo.FullName)
    Dim Node As System.Xml.XmlNode
    ' Les booléens suivant vont nous permettre de lever des erreurs personnalisées.
    Dim FoundAccueil As Boolean = False
    Dim FoundPrenom As Boolean = False
    Dim FoundNom As Boolean = False
    ' On parcours notre section appSettings de notre web.config contenue dans le
    XmlDocument1.
    For Each Node In XmlDocument.Item("configuration").Item("appSettings")
        ' On recherche les Node ayant pour nom "add" puis en fonction de la valeur
        de l'attribut
        '"key" nous allons insérer les entrées de l'utilisateur adequate dans
        l'attribut "value".
        If Node.Name = "add" Then
            If Node.Attributes.GetNamedItem("key").Value = "Accueil" Then
                Node.Attributes.GetNamedItem("value").Value = txtAccueil
                FoundAccueil = True
            End If
            If Node.Attributes.GetNamedItem("key").Value = "Prenom" Then
                Node.Attributes.GetNamedItem("value").Value = txtPrenom
                FoundPrenom = True
            End If
            If Node.Attributes.GetNamedItem("key").Value = "Nom" Then
                Node.Attributes.GetNamedItem("value").Value = txtNom
                FoundNom = True
            End If
        End If
    Next Node
    ' Si un de nos booléens a sa valeur à false on lève une exception.
    If Not FoundAccueil Then
        Throw New InstallException("Impossible de trouver la clé Acceuil")
    End If
    If Not FoundPrenom Then
        Throw New InstallException("Impossible de trouver la clé Prenom")
    End If
    If Not FoundNom Then
        Throw New InstallException("Impossible de trouver la clé Nom")
    End If
    ' Pour finir on sauvegarde notre XmlDocument1 sur le web.config
    XmlDocument.Save(FileInfo.FullName)
End Sub
```

Testons maintenant notre projet de déploiement, faites Générer puis Installer.

Voici ce que vous devriez avoir juste après l'écran de « Bienvenue ».



Finissez l'installation.

Allez dans votre répertoire d'installation de votre application Web (généralement c'est *C:\inetpub\wwwroot\MonPremierSetup*) puis ouvrez avec WordPad le web.config de votre application Web.

Vous pourrez constater que les valeurs entrées lors de l'installation on bien été enregistrées :

XML de web.config section appSettings

```
<add key="Accueil" value="Bonjour" />
<add key="Prenom" value="TITI" />
<add key="Nom" value="Toto" />
```

Ce genre de script peut avoir de multiple utilisation, par exemple vous pouvez ainsi configurer l'accès à une base de données.

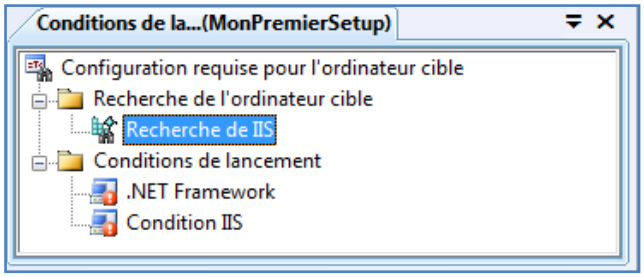
En poussant vos recherches vous pourrez essayer de créer votre base de données lors de l'installation de votre application.

3.3.3 Les conditions de lancement

Lors d'un déploiement sur une machine hôte, il peut arriver que cette machine ne remplisse pas les conditions nécessaires pour un déploiement sans entrave. Par exemple, la machine hôte ne dispose pas de la version adéquate de IIS pour le bon fonctionnement de votre application. Afin de palier à ce genre de problème, ASP.NET nous permet de définir des conditions de lancement de notre installation en ajoutant des entrées registre.

Pour gérer ces conditions, faites un clic droit sur votre projet de déploiement → Afficher →Conditions de lancement.

Voici à quoi doit votre gestionnaire de conditions de lancement :



Vous pouvez constater que notre gestionnaire se présente en deux parties :

- Recherche de l'ordinateur cible ;
- Conditions de lancement.

Nous allons voir comment utiliser ces deux sections.

Nous n'entrerons pas dans les détails de l'utilisation de cet outil nous tenons simplement à vous informer de son existence et vous en expliquer les principes de bases. Si vous désirez avoir davantage d'informations à ce sujet veuillez vous référer au site du MSDN.

Recherche de l'ordinateur cible :

Cette section va nous permettre de vérifier qu'un fichier, une clé registre ou encore un élément de Windows Installer sont bien présents.

En effectuant un clic droit sur le dossier *Recherche de l'ordinateur cible*, vous aurez le choix entre plusieurs possibilités :

- Ajouter une recherche de fichier ;
- Ajouter une recherche dans le registre ;
- Ajouter une recherche dans Windows Installer.

Chacun de ces éléments disposent de propriétés différentes dont les dénominations sont en général très explicites.

En cas de recherche infructueuse selon vos paramètres, l'installation renverra un message d'erreur et ne s'exécutera pas.

Conditions de lancement :

Cette section va nous permettre de tester la version d'un élément de votre choix avec des conditions prédéfinies dans ASP.NET.

Pour définir des conditions de lancement faite un clic droit sur *Conditions de lancement* → *Ajouter une condition de lancement.*

Une fois votre condition ajoutée vous avez plusieurs propriétés à gérer dont l'une d'entre elle est *Condition*. La gestion de cette propriété ne se fait pas au hasard et il existe des références prédéfinies qui vont nous permettre de gérer facilement notre condition.

Voici un tableau les listant :

Condition	Description
VersionNT	Fait référence à la version du système d'exploitation Microsoft Windows NT, comprenant les versions Windows 2000, XP, 2003 Server, Vista et 2008 Server.
Version9X	Fait référence aux anciennes versions des systèmes d'exploitation Microsoft Windows tels que Windows 95, 98 et Me.
ServicePackLevel	Numéro de la version du Service Pack.
WindowsBuild	Numéro de *Build* du système d'exploitation.
SystemLanguageID	Récupère la langue par défaut du système.
AdminUser	Permet de déterminer si l'utilisateur exécutant l'installation dispose des privilèges d'administrations.
PhysicalMemory	Vérifie la taille de la mémoire RAM installée (en megabytes).
IISVERSION	Vérifie la version de IIS si elle est installée.

Vous pouvez aussi effectuer des tests sur les variables d'environnement, pour cela il suffit de mettre le symbole **%** juste avant votre variable d'environnement.

Exemple :

`%HOMEDRIVE = "C:"` Vérifiera si le disque hôte d'installation est bien C :

`IISVERSION >= "#6"` Vérifiera si la version de IIS est supérieure ou égale à IIS 6.0

La propriété *InstallUrl* vous permettra de définir un emplacement ou l'utilisateur pourra récupérer le composant manquant ou dont la version n'est pas supportée.

La propriété *Message* vous permettra de définir un message d'erreur personnalisé.

3.4 Le déploiement en ligne de commande

Cette partie sera très courte et n'est destinée qu'à vous présenter les intérêts du déploiement en ligne de commande en tant que solution alternative.

3.4.1 Déployer un fichier d'installation MSI en ligne de commande

Il est possible de déployer notre application en ligne de commande en faisant appel à notre fichier d'installation MSI généré. L'une des raisons principales pour lesquelles cette méthode existe réside dans le fait que nous n'avons pas toujours accès au Bureau Windows et sommes obligés d'exécuter un déploiement à distance.

Voici comment se présente la syntaxe de cette commande :

```
msiexec /i cheminDeVotrePackageMsi
```

Exemple :

```
msiexec /i "c:\Users\Nicolas\Documents\Visual Studio
2008\Projects\MonPremierSetup\MonPremierSetup\Debug\MonPremierSetup.msi"
```

Pour avoir davantage d'informations sur les options de déploiement d'un package MSI, tapez dans la console Windows :

```
msiexec /?
```

3.4.2 Déploiement d'une application avec Xcopy

Le déploiement avec Xcopy est une simple copie de votre répertoire d'application WEB vers un dossier cible. Avec cette méthode aucune source ne sera précompilée, ce qui ne présente pas un avantage. Cela dit cette méthode peut parfois nous sortir de situations problématiques de déploiement qui n'ont pas pues être résolues avec les package MSI.

Xcopy s'utilise avec la console Windows, voici sa syntaxe :

```
Xcopy /S chemin_source chemin_destination
```

L'option **/S** permet de copier les répertoires et sous-répertoires non vides.

Exemple :

```
Xcopy /S "C:\Users\Nicolas\Documents\Visual Studio 2008\Projects\Test\Test"
C:\inetpub\wwwroot\Test
```

Pour avoir davantage d'informations sur les options de déploiement avec Xcopy, tapez dans la console Windows :

```
Xcopy /?
```

4 Monitorer : Généralités

Maintenant que nous savons comment déployer notre application web, il est également utile de savoir comment la monitorer. Ce processus consiste en la surveillance de tout ce qui pourrait se passer pendant l'exécution de l'application web que ça soit des erreurs d'exécution, des authentifications ratées ou réussies, etc. Tous les outils nécessaires au monitoring d'application web se trouvent dans l'espace de nom *System.Web.Management*.

Pour cela, ASP.NET vous permet « d'enregistrer » les évènements de l'application grâce à des fournisseurs de stockage. Ces fournisseurs peuvent envoyer des données dans différents endroits :

- Dans un email, grâce aux fournisseurs *SimpleMailWebEventProvider* et *TemplatedMailWebEventProvider*.
- Dans une base de données SQL Server grâce au fournisseur *SqlEventProvider*. Par défaut, ce fournisseur envoi des données dans la base de données présente dans le dossier *App_Data* du projet.
- Dans le journal d'évènements Windows grâce au fournisseur *EventLogWebEventProvider*. Ce type de fournisseur enregistre toutes les erreurs générées.
- Dans le système WMI (Windows Management Instrumentation) grâce au fournisseur *WmiWebEventProvider*.
- Dans un fournisseur de données personnalisé que vous pouvez créer en implémentant la classe *WebEventProvider* ou *BufferedWebEventProvider*.

Chacun de ces fournisseurs vont utiliser des classes décrivant les évènements à enregistrer. Le Framework ASP.NET fourni par défaut une liste d'évènements nommés « Web<type>Event » où « type » est l'un des évènements à surveiller (par exemple : *WebAuthenticationFailureAuditEvent* pour surveiller les tentatives d'authentifications ratées). Là encore, vous pouvez créer vos propres évènements en héritant d'une des classes Web<type>Event.

4.1 Explications par le code

Vous pouvez bien sur créer le système de monitoring en passant par le code-behind de votre application web ; mais ce n'est pas la méthode la plus aisée. Pour implémenter de façon simple un système de monitoring de votre application, nous allons plutôt travailler sur une balise à placer à l'intérieur de la balise <System.web> de votre fichier de configuration web.config : La balise <healthMonitoring>. Cette balise peut en contenir :

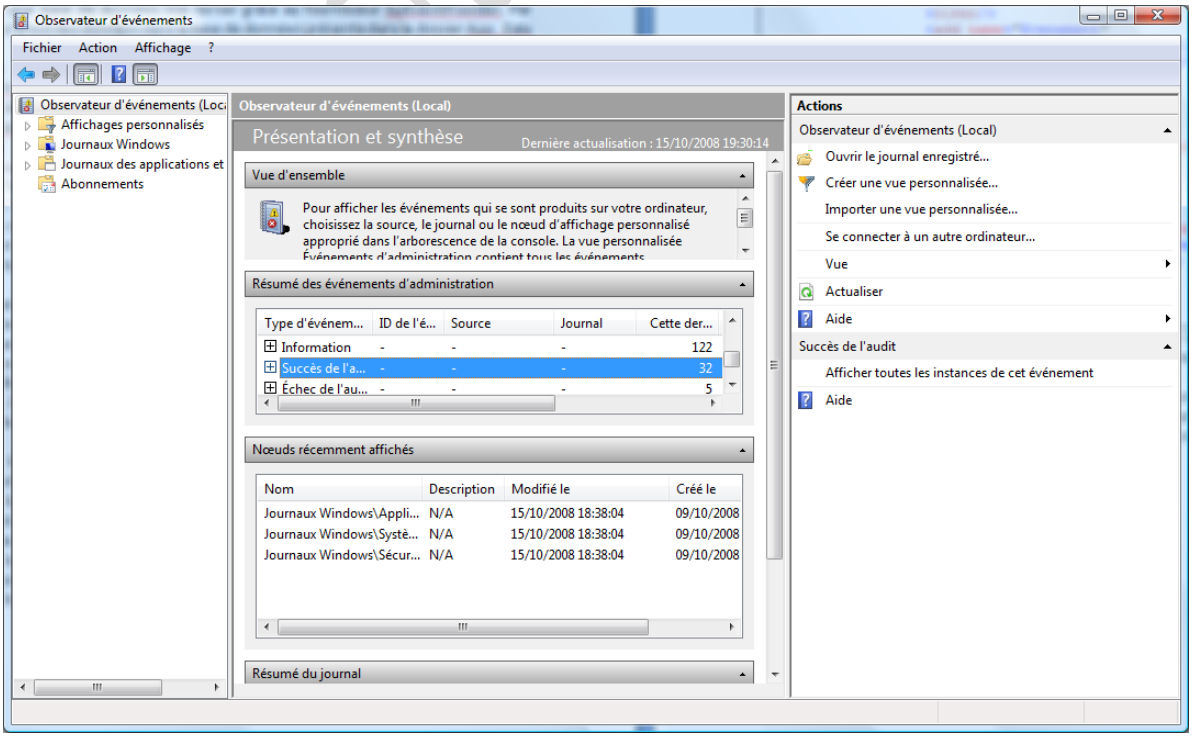
- Une balise <providers> contenant la liste des fournisseurs de données de monitoring à utiliser.
- Une balise <profiles> qui liste les intervalles de temps d'enregistrement des évènements pour éviter de surcharger la zone de stockage avec des messages redondants.
- Une balise <bufferModes> qui permet (un peu comme <profiles>) de placer les évènements enregistrés dans un buffer avant de les envoyer au fournisseur de donnée, toujours dans le but d'éviter les messages redondants.
- Une balise <rules> qui permet relier les évènements surveillés avec l'endroit vers lequel ils vont enregistrer des données.
- Une balise <eventMappings> qui permet de lier les classes d'évènements à surveiller avec des noms.

XML de web.config

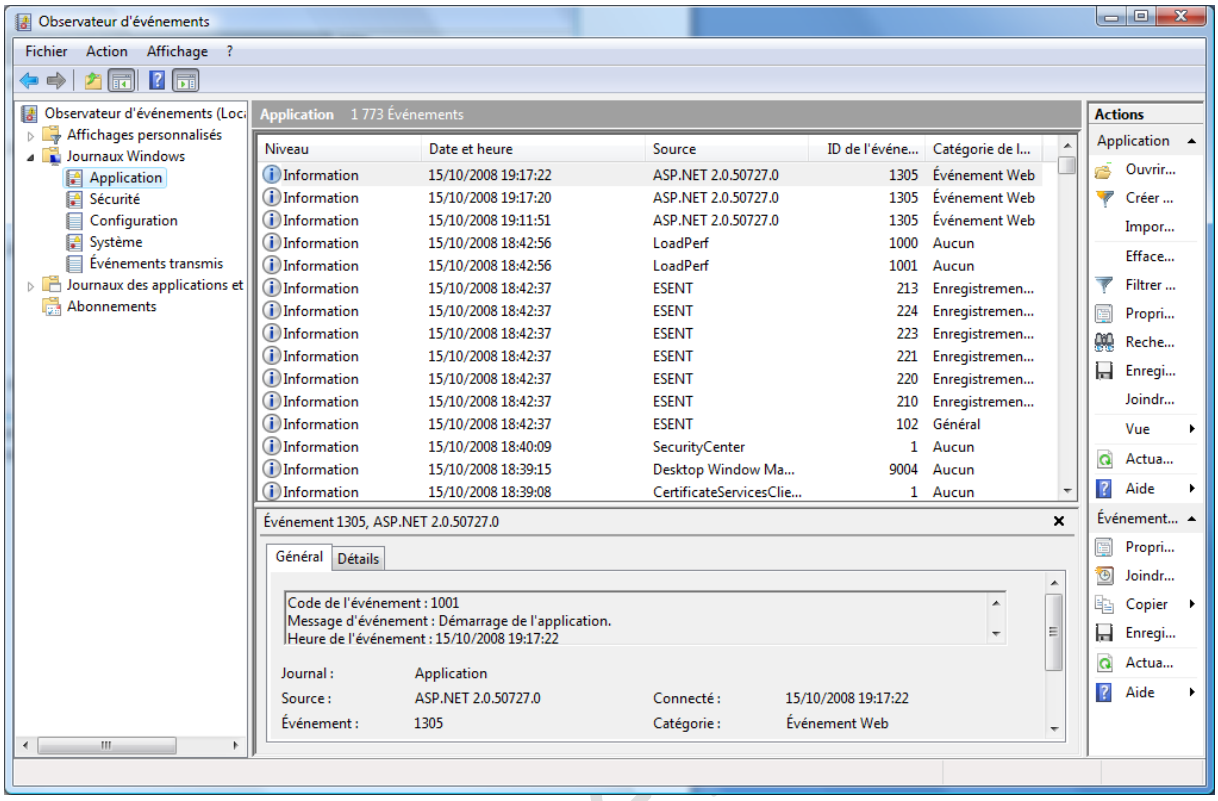
```
<system.web>
        <healthMonitoring>
        <providers>
                <clear/>
                <add name="Fournisseur" type="System.Web.Management.EventLogWebEventProvider,
                        System.Web,Version=2.0.0.0,Culture=neutral,
                        PublicKeyToken=b03f5f7f11d50a3a"/>
        </providers>
        <bufferModes>
                <clear/>
        </bufferModes>
        <eventMappings>
                <clear/>
                <add name="Evenement"
                type="System.Web.Management.WebApplicationLifetimeEvent,System.Web,Version=2.0.
                0.0,Culture=neutral,PublicKeyToken=b03f5f7f11d50a3a" startEventCode="0"/>
        </eventMappings>
        <profiles >
                <clear/>
                <add name="ProfileEvent" maxLimit="Infinite" minInstances="1"
                minInterval="00:00:30"/>
        </profiles>
        <rules>
                <clear/>
                <add eventName="Evenement" maxLimit="100" minInstances="1"
                minInterval="00:00:00" name="Regle" profile="ProfileEvent"
                provider="Fournisseur"/>
        </rules>
        </healthMonitoring>
</system.web>
```

Dans l'exemple ci-dessus, nous avons simplement créé la surveillance du démarrage et de l'arrêt de notre application Web et nous inscrivons les évènements dans le journal d'évènements Windows.

Nous pouvons ensuite simplement accéder aux informations enregistrées en ouvrant le journal d'évènements. Pour cela, faites Menu Démarrer → Panneau de configuration →Outils d'administrations et lancez le raccourci « Observateurs d'évènements ».



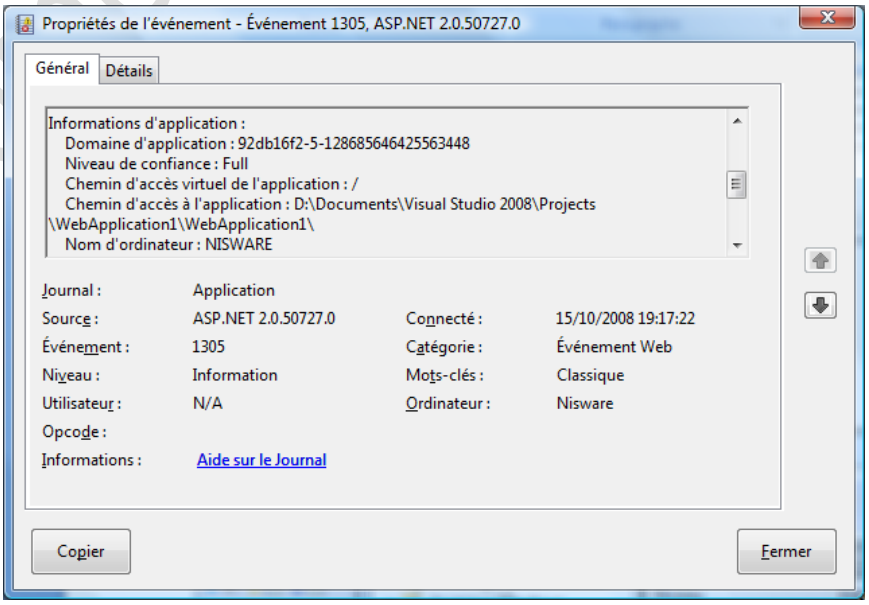
Une fois la console MMC avec le composant lancée, déroulez « Journaux Windows » et sélectionnez « Applications » :



Ici, l'évènement Web apparait trois fois car j'ai exécuté l'application web deux fois, sans la fermer la seconde fois. Dans l'ordre, j'ai donc :

- Notification de démarrage de l'application ;
- Notification d'arrêt ;
- Notification de démarrage.

Je peux également double-cliquer sur une des notifications pour avoir un peu plus de détails :



5 Conclusion

Au cours de ce chapitre, nous avons vu comment déployer une Application Web, nous avons vu aussi comment la surveiller par le monitoring.

Les notions de déploiement sont très importantes pour une entreprise et doivent être sues et assimilées. L'un des principes de base du déploiement est le gain de temps de mise en place. En effet, un bon déploiement n'est pas forcément rapide mais se doit de faire gagner du temps.

Quant au monitoring, surveiller son application pour vérifier son bon fonctionnement est nécessaire dans les premiers temps. Ceci étant, à termes, le monitoring vous permettra détecter les attaques malveillantes grâce au système de log (redémarrage intempestif de votre application par exemple). Une notion à bien connaitre donc.