Ma première application Windows Phone 7 en Silverlight

par Philippe Vialatte (Espace Perso) (Blog) Jérôme Lambert (Espace Perso) (Blog)

Date de publication : 02 juillet 2010

Dernière mise à jour :

Ce tutoriel va vous permettre de prendre en main l'environnement de développement d'applications Windows Phone 7 avec Silverlight. L'exercice vous permettra de faire vos premiers pas avec l'émulateur Windows Phone 7 grâce auquel vous pourrez créer, tester et lancer votre première visionneuse d'images.

I - Introduction

Le challenge Windows Phone 7 est l'occasion de découvrir les capacités de ce nouvel OS et de l'environnement de développement qui y est associé. On va donc, dans ce tutoriel, commencer à prendre en main les outils de développements.

À la fin de ce tutoriel, vous saurez comment :

- Créer une application Silverlight pour Windows Phone 7
- Manipuler l'émulateur Windows Phone 7
- Ajouter des boutons dans la barre d'application
- Appeler un service Web depuis une application Silverlight
- Enregistrer des données dans l'Isolated Storage
- Lire des données depuis l'Isolated Storage
- Manipuler les contrôles Button et Image

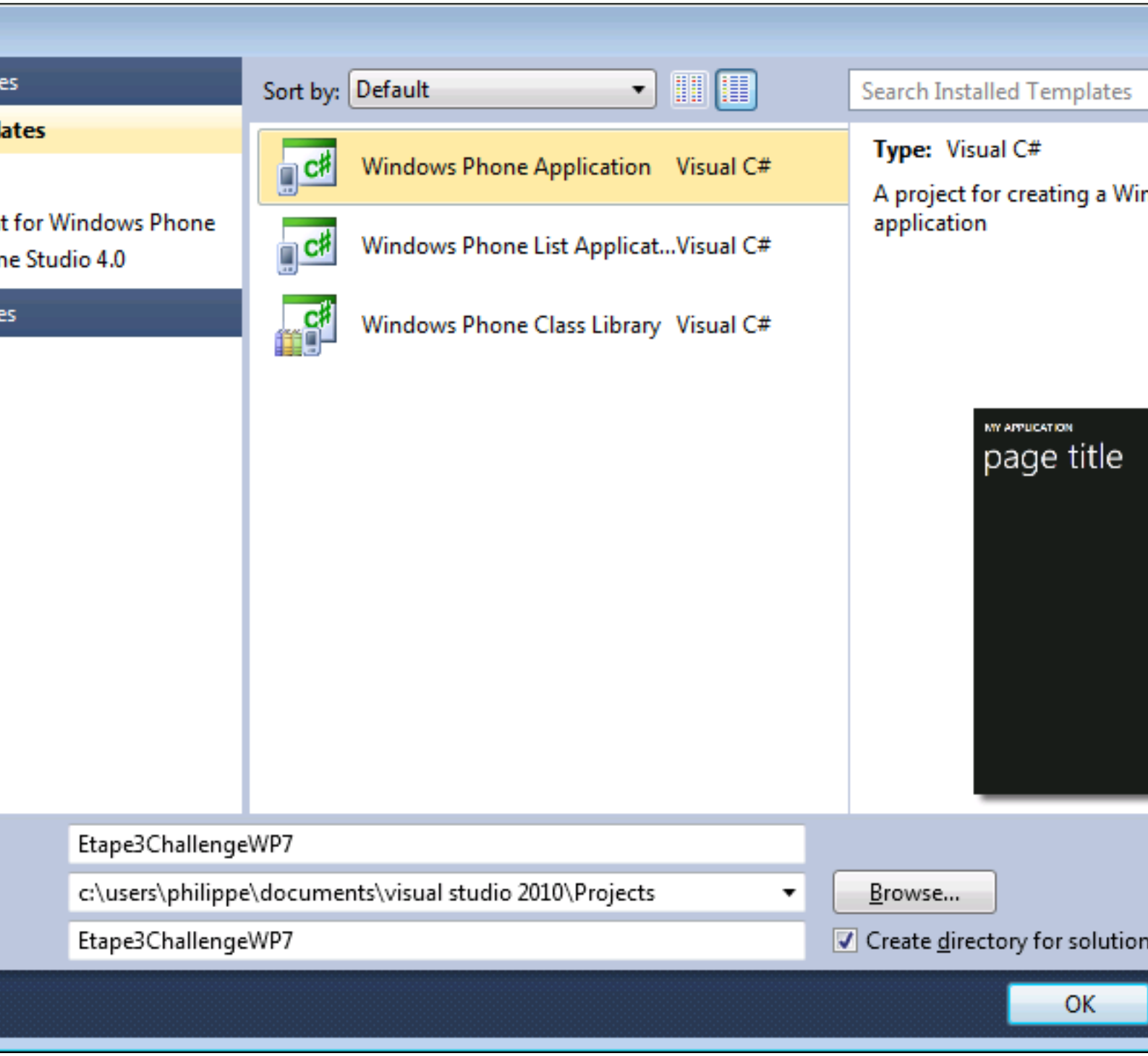
Pour cela, nous allons écrire une application qui va récupérer un ensemble d'images depuis un service web, les enregistrer dans l'Isolated Storage, puis les relire, de façon à ce que vous trouviez des informations pour valider l'épreuve.

II - Création du projet et présentation de l'environnement

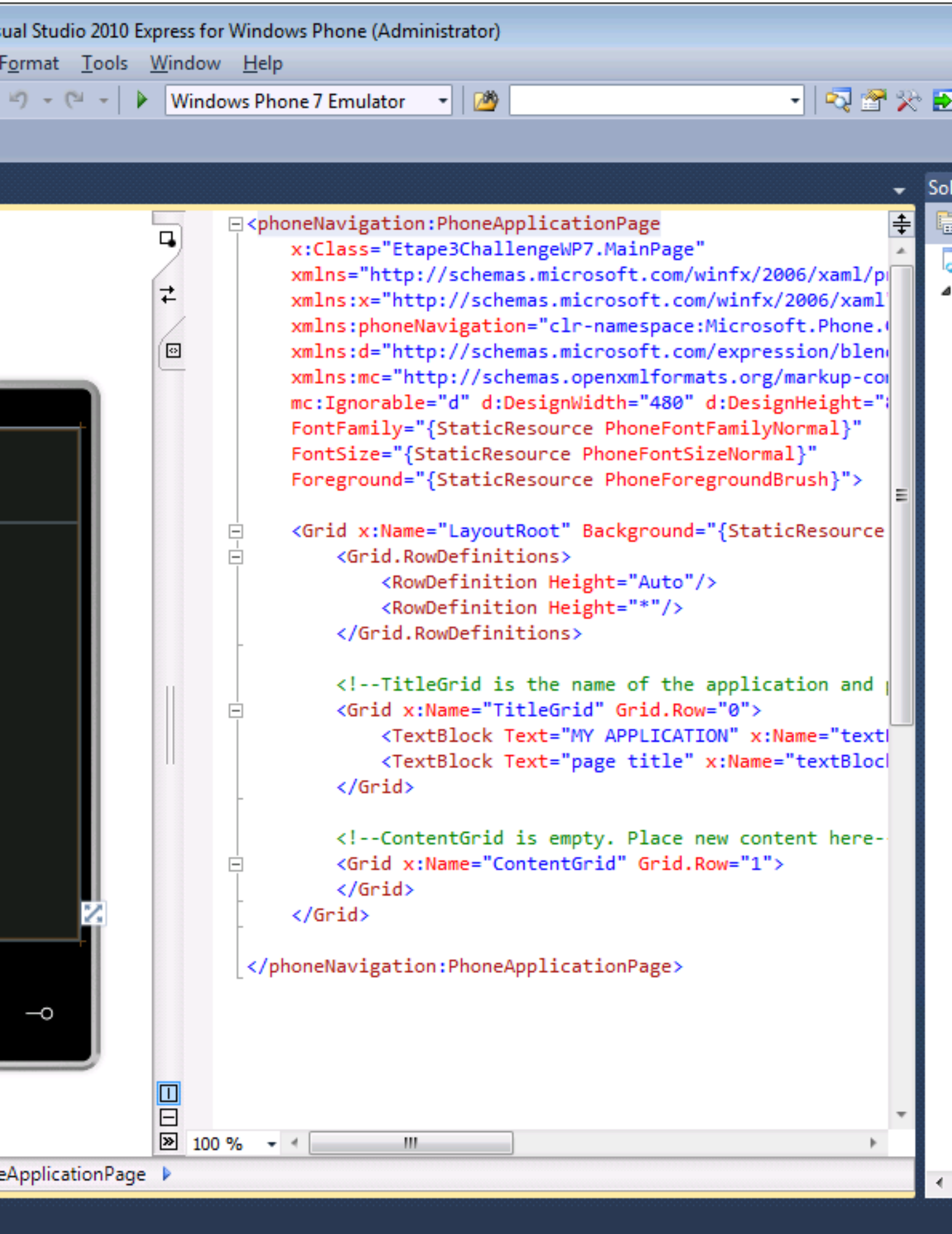
Nous allons commencer par créer le projet. Pour cela, il faut que vous ayez préalablement installé les outils de développement nécessaires (ce qui constituait **la première épreuve du challenge Windows Phone**). Au cas où cela ne serait pas encore fait, ces outils sont disponibles ici :

Une fois les outils installés, on va commencer par lancer l'application Visual Studio 2010 Express for Windows Phone CTP Refresh. Il faut ensuite créer un nouveau projet, en cliquant sur File|New Project, et en sélectionnant le template "Windows Phone Application" dans la section "C# > Silverlight for Windows Phone templates"

On va nommer le projet "Etape3ChallengeWP7".



On voit apparaitre notre espace de travail, avec sur la droite l'explorateur de solution, au milieu l'éditeur de code, et à gauche, notre prévisualisation.


ual Studio 2010 Express for Windows Phone (Administrator)
Format Tools Window Help
Windows Phone 7 Emulator
<phoneNavigation:PhoneApplicationPage
x:Class="Etape3ChallengeWP7.MainPage"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/p
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml
xmlns:phoneNavigation="clr-namespace:Microsoft.Phone.
xmlns:d="http://schemas.microsoft.com/expression/blen
xmlns:mc="http://schemas.openxmlformats.org/markup-co
mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="
FontFamily="{StaticResource PhoneFontFamilyNormal}"
FontSize="{StaticResource PhoneFontSizeNormal}"
Foreground="{StaticResource PhoneForegroundBrush}">
<Grid x:Name="LayoutRoot" Background="{StaticResource
<Grid.RowDefinitions>
<RowDefinition Height="Auto"/>
<RowDefinition Height="*"/>
</Grid.RowDefinitions>
<!--TitleGrid is the name of the application and
<Grid x:Name="TitleGrid" Grid.Row="0">
<TextBlock Text="MY APPLICATION" x:Name="textB
<TextBlock Text="page title" x:Name="textBloc
</Grid>
<!--ContentGrid is empty. Place new content here-
<Grid x:Name="ContentGrid" Grid.Row="1">
</Grid>
</Grid>
</phoneNavigation:PhoneApplicationPage>
100 %
eApplicationPage

Pour bien commencer, on va modifier le titre de l'application, en mettant à jour la grille nommée "TitleGrid" de la façon suivante :

MainPage.xaml

```
<Grid x:Name="TitleGrid" Grid.Row="0">

  <TextBlock Text="Challenge Windows Phone 7" x:Name="textBlockPageTitle" Style="{StaticResource PhoneTextPageTit

  <TextBlock Text="Epreuve 3" x:Name="textBlockListTitle" Style="{StaticResource PhoneTextPageTitle2Style}"/>
</Grid>
```

III - Ajout des boutons dans la barre d'application

L'application est, pour le moment, un peu vide. Pour la rendre plus sexy, on va lui ajouter trois boutons dans la barre d'application. Ces boutons proviennent du pack d'icônes fournies par Microsoft : **http://dotnet.developpez.com/faq/wp7s/?page=ressources-outils#icones**

Les images que l'on a choisies sont les suivantes :

appbar.back.rest.png

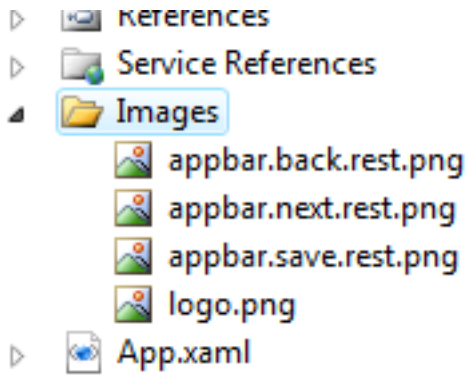
appbar.save.rest.png

appbar.next.rest.png

On va donc créer un répertoire Image, et y ajouter ces images. Le résultat devrait être le suivant :

Pour créer le répertoire Images, faites un clique droit sur le projet Etape3ChallengeWP7 dans le Solution Explorer et sélectionnez Add|New Folder. Renommez le répertoire en Images.

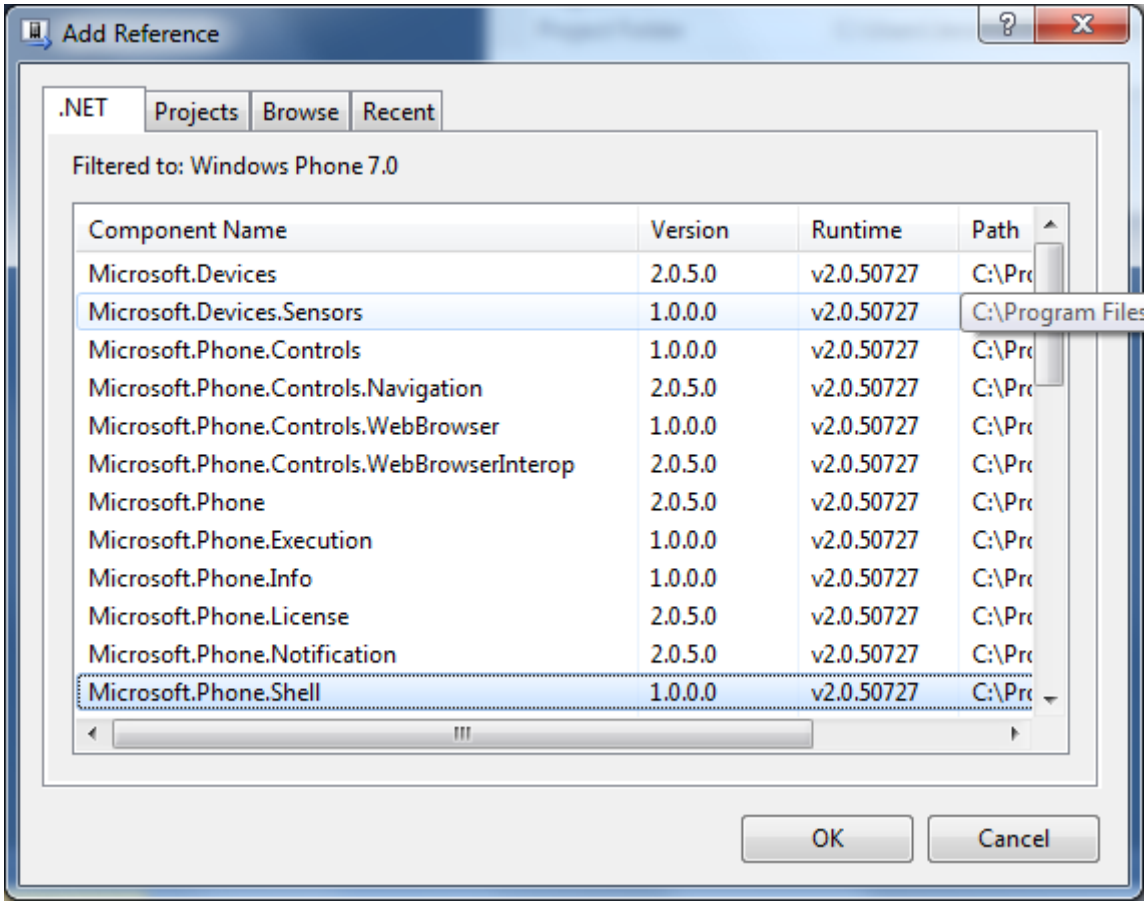
Une fois le répertoire créé, nous allons ajouter les icones cités ci-dessus. Pour cela, faites un clique droit sur le répertoire Images, sélectionnez Add|Existing Item... et ajoutez les 3 fichiers.



Il faut absolument que ces images soient configurées avec Build Action = Content et Copy to Output Directory = Always, sinon, elles n'apparaitront pas dans l'émulateur

Sélectionnez chaque fichier image et modifiez les propriétés "Build Action" en "Content" et "Copy to Output Directory" en "Copy Always" via la fenêtre des propriétés (pour afficher la fenêtre des propriétés, selectionnez "Other Windows|Properties Windows" à partir du menu "View").

Pour manipuler la barre d'application, il faut ajouter à votre projet une référence à la dll Microsoft.Phone.Shell. Pour cela, faites un clique droit sur le projet Etape3ChallengeWP7 à partir du "Solution Explorer" et sélectionnez "Add Reference...". Dans l'onglet ".NET", sélectionnez Microsoft.Phone.Shell.



Il faut ensuite ajouter une référence au namespace dans le code XAML du fichier "MainPage.xaml".

MainPage.xaml

```
xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone.Shell"
```

On va ensuite ajouter la barre d'application en elle-même. Pour cela, on va ajouter le code suivant dans MainPage.xaml

MainPage.xaml

```
<phoneNavigation:PhoneApplicationPage.ApplicationBar>
    <shell:ApplicationBar IsVisible="True">
        <shell:ApplicationBar.Buttons>
            <shell:ApplicationBarIconButton IconUri="/Images/
appbar.back.rest.png" Click="buttonPreviousPicture_Click" IsEnabled="False" />
            <shell:ApplicationBarIconButton IconUri="/Images/
appbar.save.rest.png" Click="buttonDownloadPictures_Click" IsEnabled="True" />
            <shell:ApplicationBarIconButton IconUri="/Images/
appbar.next.rest.png" Click="buttonNextPicture_Click" IsEnabled="False" />
        </shell:ApplicationBar.Buttons>
    </shell:ApplicationBar>
```

MainPage.xaml

```
</phoneNavigation:PhoneApplicationPage.ApplicationBar>
```

Et enfin, on va ajouter les événements nécessaires dans le fichier MainPage.xaml.cs

MainPage.xaml.cs

```
private void buttonDownloadPictures_Click(object sender, EventArgs e)
{
}

private void buttonPreviousPicture_Click(object sender, EventArgs e)
{
}

private void buttonNextPicture_Click(object sender, EventArgs e)
{
}
```

Notre application ne fait pas encore grand chose, mais on va vite ajouter des fonctionnalités. En attendant, si on lance l'application, l'émulateur s'affichera dorénavant ainsi :

IV - Lecture et sauvegarde de données dans l'Isolated Storage

Notre application va devoir récupérer plusieurs images depuis un service web.
On va donc devoir appeler au total six fois le service web adéquat. Ce service web est hébergé à l'adresse suivante : **http://challenge-windowsphone7.developpez.com/Epreuve3.asmx**

Avant de commencer réellement à travailler avec le service web, on va commencer par déclarer un certain nombre de variables, qui vont nous permettre de gérer l'état de notre application:

MainPage.xaml.cs

```
private string _pictureName = "epreuve3picture{0}";
private int _CountPictures = 0;
private int _CurrentPicture = 1;
```

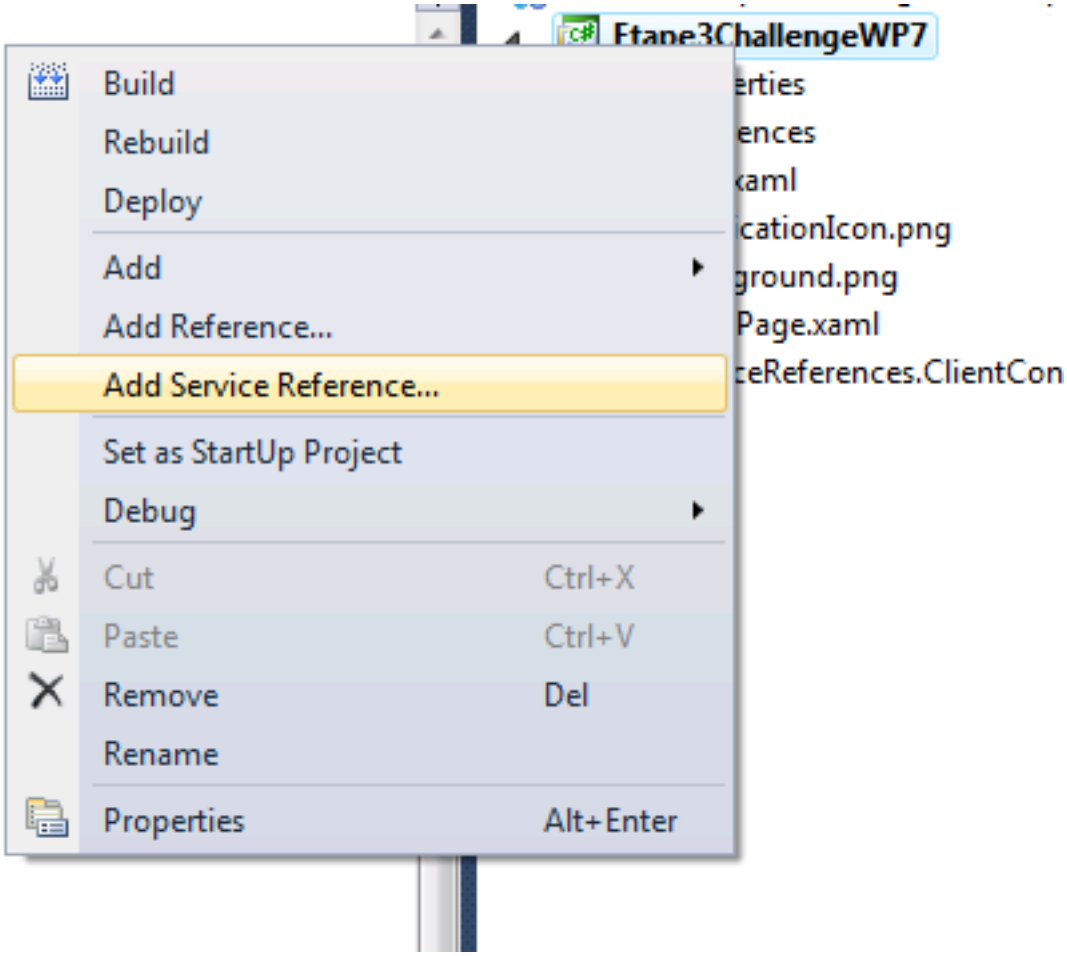
Il faudra aussi, pour utiliser le code à l'identique, avoir toutes les directives using ci-dessous déclarées.

MainPage.xaml.cs

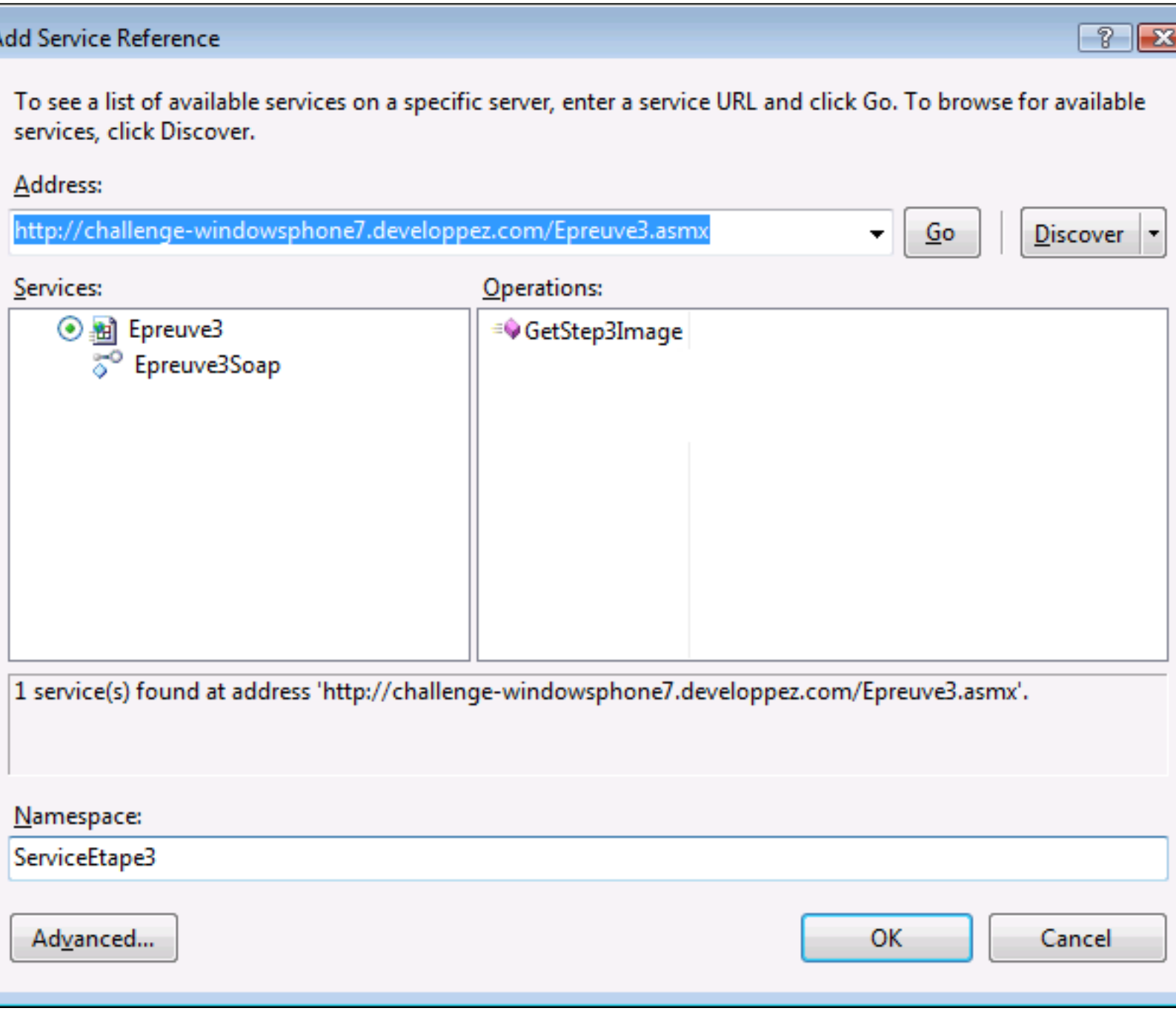
```
using Etape3ChallengeWP7.ServiceEpreuve3;
using System.ComponentModel;
```

Ajouter une référence à un service web se fait exactement de la même façon que pour un site web ou une application standard. Il faut effectuer les étapes suivantes :

D'abord, faire un clic droit sur le projet Etape3ChallengeWP7, puis sur "Add Service Reference".



Ensuite, dans le champ "Address", entrez l'adresse du service web, changez le champ Namespace en "ServiceEtape3", et cliquez "OK".



La méthode GetStep3Image permet, en lui passant un entier, de télécharger l'image correspondant à ce numéro d'index sous forme d'un tableau de bytes.

On va commencer par ajouter le code qui va être appelé grâce à un clic sur le bouton de téléchargement. Pour cela, on va ajouter le code suivant :

MainPage.xaml.cs

```
private void buttonDownloadPictures_Click(object sender, EventArgs e)
{
 // initialisation des compteurs
 _CountPictures = 0;
 _CurrentPicture = 1;

 // désactivation des boutons
 ((ApplicationBarIconButton)ApplicationBar.Buttons[0]).IsEnabled = false;
 ((ApplicationBarIconButton)ApplicationBar.Buttons[1]).IsEnabled = false;
```

MainPage.xaml.cs

```
  ((ApplicationBarIconButton)ApplicationBar.Buttons[2]).IsEnabled = false;

  // création du client du service WCF
  var client = new ServiceEpreuve3.Epreuve3SoapClient();

  // affectation des événements
  client.GetStep3ImageCompleted += new
  EventHandler<GetStep3ImageCompletedEventArgs>(client_GetStep3ImageCompleted);
  client.CloseCompleted += new EventHandler<AsyncCompletedEventArgs>(client_CloseCompleted);

  // on appelle ensuite le service web pour les six images
  for (int i = 1; i < 7; i++)
  {
      client.GetStep3ImageAsync(i, new ProxyRequestData(client, i));
  }
 }
```

Les appels à un service web étant systématiquement asynchrones avec Silverlight, il a fallu s'abonner à l'événement GetStep3ImageCompleted pour être notifié lorsque le serveur aura renvoyé à notre client le résultat de notre demande. Une classe supplémentaire a été ajoutée, ProxyRequestData, qui permettra de communiquer des informations supplémentaires à notre méthode client_GetStep3ImageComplete une fois que le serveur aura renvoyé le résultat.

MainPage.xaml.cs

```
public class ProxyRequestData
{
    public ProxyRequestData(ServiceEpreuve3.Epreuve3SoapClient client, int picturePosition)
    {
        this.Client = client;
        this.PicturePosition = picturePosition;
    }
    public ServiceEpreuve3.Epreuve3SoapClient Client { get; set; }
    public int PicturePosition { get; set; }
}
```

La méthode client_GetStep3ImageCompleted va récupérer la valeur de retour de la méthode GetStep3Image, pour la stocker dans l'Isolated Storage. Si on est à la dernière image, on va ensuite appeler la méthode Close du client WCF :

MainPage.xaml.cs

```
private void client_GetStep3ImageCompleted(object sender,
 ServiceEpreuve3.GetStep3ImageCompletedEventArgs e)
{
 if (e.Result == null) return;

    ProxyRequestData data = (ProxyRequestData)e.UserState;

    lock (data.Client)
    {
        if (e.Cancelled == true || e.Error != null)
        {
            if (data.Client.State == System.ServiceModel.CommunicationState.Opened)
                data.Client.CloseAsync();
            return;
        }

        var fileName = string.Format(_pictureName, data.PicturePosition);

        using (var appStorage = IsolatedStorageFile.GetUserStoreForApplication())
        {
   if (appStorage.FileExists(fileName))
            {
                appStorage.DeleteFile(fileName);
            }
            using (var stream = new IsolatedStorageFileStream(fileName, FileMode.CreateNew,
 appStorage))
```

MainPage.xaml.cs

```
                {
                    stream.Write(e.Result, 0, e.Result.Length);
                }
            }

            _CountPictures++;

    // on ne récupère que six images, et on appelle la méthode Close
             if (_CountPictures == 6)
            {
                data.Client.CloseAsync();
            }
        }
    }
```

Comme vous le voyez ci-dessus, l'écriture de données dans l'Isolated Storage est simplissime, le fonctionnement est quasiment le même que celui du système de fichier local.

Pourquoi utiliser l'Isolated Storage?
L'Isolated Storage est une structure de stockage logique, et est, en pratique, le seul endroit ou une application va pouvoir stocker des données. En effet, Windows Phone 7 ne permet pas la sauvegarde des données directement dans le système de fichiers.

La dernière fonction que l'on verra dans cette partie est celle qui est appelée lorsque l'on ferme le service WCF. Elle va simplement afficher un message, réactiver les boutons, et appeler une méthode permettant d'afficher l'image courante.

MainPage.xaml.cs

```
private void client_CloseCompleted(object sender, System.ComponentModel.AsyncCompletedEventArgs e)
{
    MessageBox.Show(string.Format("{0} photos ont été téléchargées !", _CountPictures),
        "Résultat", MessageBoxButton.OK);

    ((ApplicationBarIconButton)ApplicationBar.Buttons[0]).IsEnabled = true;
    ((ApplicationBarIconButton)ApplicationBar.Buttons[1]).IsEnabled = false;
    ((ApplicationBarIconButton)ApplicationBar.Buttons[2]).IsEnabled = true;

    DisplayCurrentPicture();
}
```

On verra le code nécessaire pour afficher l'image plus loin.

V - Affichage des images

Pour afficher les images, on va se baser sur le contrôle Image, fourni de base avec Silverlight.
On va donc commencer par faire glisser un contrôle de type Image depuis la barre d'outils dans notre zone de travail, et par le renommer en imageDisplay. Enfin, on va ajuster sa hauteur et sa largeur pour qu'il s'adapte à la surface de l'émulateur. Le code devrait ressembler à l'exemple suivant :

MainPage.xaml.cs

```
<Grid x:Name="ContentGrid" Grid.Row="1">
    <Image HorizontalAlignment="Stretch" VerticalAlignment="Stretch" Stretch="Uniform"
            Name="imageDisplay"
            Width="468" Margin="6,6,0,32" />
</Grid>
}
```

Notre contrôle image étant désormais présent sur la page, on va maintenant se concentrer sur la méthode DisplayCurrentPicture.

MainPage.xaml.cs

```
private void DisplayCurrentPicture()
{
    var fileName = string.Format(_pictureName, _CurrentPicture);
    byte[] data;

 // récupération des données dans l'Isolated Storage
    using (var storage = IsolatedStorageFile.GetUserStoreForApplication())
    {
        if (!storage.FileExists(fileName)) return;

        using (var file = storage.OpenFile(fileName, FileMode.Open))
        {
            data = new byte[file.Length];
            file.Read(data, 0, data.Length);
        }
    }

 // transformation du flux de données en image
    var image = new BitmapImage();
    using (var stream = new MemoryStream(data))
    {
        image.SetSource(stream);
    }

 // affectation de l'image au contrôle
    imageDisplay.Source = image;
}
```

On voit que la récupération de données depuis l'Isolated Storage est aussi simple que l'écriture. Le contrôle imageDisplay attendant une source de type ImageSource, on convertit le flux de données en BitmapImage, puis on affecte cette nouvelle image au contrôle.

Les derniers contrôles à gérer sont les contrôles "Précédent" et "Suivant". Pour les gérer, on va simplement s'assurer de pouvoir respectivement passer à l'image précédente ou suivante, puis changer l'index de l'image courante, et rappeler DisplayCurrentPicture. Tout cela se fait ainsi :

MainPage.xaml.cs

```
 private void buttonPreviousPicture_Click(object sender, EventArgs e)
{
    // notre première image à l'index 1
    if (_CurrentPicture < 2) return;

    _CurrentPicture--;
    DisplayCurrentPicture();
}

private void buttonNextPicture_Click(object sender, EventArgs e)
{
    if (_CurrentPicture >= _CountPictures) return;

    _CurrentPicture++;
    DisplayCurrentPicture();
}
```

L'application est terminée ! Vous pouvez désormais visionner les images fournies par le service web.

VI - Validation de l'épreuve

Une fois les images visualisées, vous devez désormais avoir toutes les informations nécessaires pour valider l'épreuve.

Vous n'avez plus qu'à valider l'épreuve pour continuer l'aventure !
Pour tout blocage, incompréhension ou autre question, n'hésitez pas à ***vous rendre sur le forum***

VI - Remerciements

Merci à tous les participants au challenge, j'espère que cette épreuve vous a plu, et que vous ne saurez jamais à quel point la préparation en a été compliquée.