Hands-On Lab

Hello Windows Phone

Version : 1.1.0

Dernières mise à jour : 1/10/2012

Table des matières

Hello Windows Phone..... 1

OBJECTIFS 4

PREREQUIS 4

EXERCICES 5

Exercice 1 – Créer un projet d'application Windows Phone dans Visual Studio 7

Exercice 2 – Compilez et tester l'application dans l'émulateur Windows Phone 15

Exercice 3 – Définir l'interface utilisateur 20

Exercice 4 – Gérer les évènements depuis l'interface utilisateur 26

Exercice 5 – Gérer les erreurs dans l'application 29

Exercice 6 – Vérification 33

Exercice 1 – Création d'un bouton personnalisé avec Expression Blend 38

Exercice 2 – Ajouter des effets visuels à un contrôle 48

Exercice 3 – Créer une animation pour le texte de la bannière 51

Exercice 4 – Vérification 60

Vue d'ensemble

Cet atelier s'appuie sur le développement du traditionnel "*Hello World*" pour présenter les outils et les procédures nécessaires pour construire et tester des applications Silverlight pour Windows Phone. La plateforme applicative Windows Phone ouvre la porte à :

Des applications et des jeux fantastiques. Utiliser les Frameworks Silverlight et XNA pour concevoir des applications et jeux interactifs de haute-qualité pour les téléphones Windows Phone 7.

Des applications rapides et incroyablement riches. La combinaison de Visual Studio 2010 et Expression Blend avec les spécifications matérielles standards Windows Phone 7 permet de gagner du temps pour faire en sorte que la vision de chacun devienne une réalité sur téléphone.

Plus d'opportunités et de partenariats. Concevez des applications et des jeux innovants avec vos compétences actuelles et les outils disponibles pour Windows Phone 7, et touchez de nombreux clients, indépendamment du matériel qu'ils utilisent. Vous n'avez pas à vous soucier des ventes et de la distribution, le Marketplace Windows Phone s'en charge. Développez et recueillez-en les fruits !

Au cours de l'atelier, vous verrez comment utiliser *Microsoft Visual Studio 2010 Express pour Windows Phone* et *Expression Blend* pour concevoir vos applications Windows Phone. *Visual Studio* fournit un environnement pour les développeurs d'applications tandis qu'*Expression Blend* est plus destiné aux designers, en charge de l'expérience utilisateur (UX). Les outils se complètent donc mutuellement et permettent une étroite collaboration designer / développeur.

Cet atelier présente un scénario de développement typique, dans lequel vous concevez tout d'abord dans Visual Studio la logique de l'application Windows Phone, les mises en pages et ses contrôles, avant de définir l'expérience utilisateur avec Expression Blend en appliquant les styles et les animations des pages et contrôles. Dans cet atelier, vous découvrirez la structure d'une application Windows Phone et apprendrez à utiliser l'*Extensible Application Markup Language* (XAML) pour créer l'interface utilisateur (UI) de votre application. XAML est un langage déclaratif, et le format principal pour définir une interface utilisateur Silverlight et ses composants. Enfin, vous apprendrez à déployer et déboguer votre application Windows Phone au travers de l'émulateur Windows Phone.

Objectifs

Cet atelier est principalement destiné aux développeurs peu familiers avec les outils de développement Microsoft tels quel Visual Studio, et qui découvrent Silverlight.

Ainsi, si vous êtes à l'aise avec le modèle de programmation Silverlight et travaillez déjà avec Visual Studio et la suite Expression de Microsoft, vous pouvez passer cet atelier.

Par contre, si vous débutez totalement sur Silverlight, nous vous recommandons fortement d'enrichir vos compétences en commençant par quelques ateliers spécifiques disponibles sur : http://silverlight.net/learn/ .

Dans cet atelier pratique, vous allez:

- Vous familiariser avec les outils de développement Windows Phone : *Microsoft Visual Studio 2010 Express pour Windows Phone* et *Windows Phone Emulator.* Rien d'autre n'est nécessaire pour créer et tester n'importe quelle application Windows Phone.
- Utiliser *Microsoft Expression Blend* pour améliorer l'expérience utilisateur de votre application en appliquant des styles et animations.
- Apprendre la structure sous-jacente à une application Silverlight pour Windows Phone, et les différences entre *Silverlight* et *Silverlight pour Windows Phone*.
- Écrire, tester, déployer et déboguer votre application Silverlight pour Windows Phone en utilisant Microsoft Visual Studio 2010 Express pour Windows Phone et Windows Phone Emulator.

Prérequis

Sont présentés ci-dessous les prérequis logiciels pour réaliser cet atelier :

- Microsoft Visual Studio 2010 Express for Windows Phone ou Microsoft Visual Studio 2010
- Microsoft Expression Blend
- Les outils de développement *Windows Phone Developer Tools*, que vous pouvez télécharger sur : http://create.msdn.com

---

Exercices

Cet atelier comprend les exercices suivants :

1. Création d'une application Windows Phone avec Microsoft Visual Studio 2010 Express pour Windows Phone
2. Utilisation d'Expression Blend pour concevoir l'interface utilisateur (UX) de l'application Windows Phone

---

Durée estimée de réalisation de l'atelier : **45 minutes**.

Exercice 1: Créer des applications Windows Phone avec Microsoft Visual Studio 2010 Express pour Windows Phone

Dans cet exercice, vous allez créer, tester, déployer et exécuter votre 1ère application « Hello World » avec Silverlight pour Windows Phone. Vous apprendrez comment utiliser les outils de développement Windows Phone, notamment l'environnement de développement intégré (IDE) gratuit Microsoft Visual Studio 2010 Express pour Windows Phone et l'émulateur Windows Phone.

Durant l'exercice, vous créerez un nouveau projet Silverlight pour Windows Phone. Vous utiliserez les éléments de base de toute application Windows Phone, ajoutant des boutons et zones de texte au modèle de projet Windows Phone par défaut. S'agissant d'une application Silverlight pour Windows Phone, vous devrez suivre le modèle de programmation Silverlight et définir votre interface utilisation grâce au XAML. Bien qu'il soit possible de développer une interface utilisateur basée sur XAML avec n'importe quel code, il sera bien plus aisé de tirer parti de l'outillage performant et simple disponible lorsque vous utiliserez XAML.

Pour concevoir les interfaces utilisateurs pour Windows Phone avec XAML, vous éditerez le code XAML de la page principale pour ajouter 3 éléments : une zone de texte, un bouton et un bloc de texte. Ces contrôles Silverlight ont été portés pour fonctionner dans l'environnement d'exécution Silverlight pour Windows Phone. Une fois finalisée, l'application vous permettra d'afficher, via un bouton, le texte que vous aurez saisi préalablement dans la zone de texte appropriée.

Enfin, une fois le code écrit et finalisé, vous le déploierez sur l'émulateur Windows Phone afin de le tester. Pour déboguer les applications exécutées dans l'émulateur, vous pourrez utiliser des points d'arrêts vous permettant de faire du pas-à-pas au sein de votre code source pour analyser les variables de votre programme.

Remarque : les étapes de cet atelier sont présentées au travers de Microsoft Visual Studio 2010 Express pour Windows Phone, mais peuvent tout aussi bien s'appliquer de la même manière à Microsoft Visual Studio 2010 avec les outils de développement Windows Phone (*Developer Tools*). Les instructions se référant à Visual Studio s'appliquent aux 2 produits.

Exercice 1 – Créer un projet d'application Windows Phone dans Visual Studio

Vous utiliserez ici un modèle prédéfini dans Microsoft Visual Studio 2010 Express pour Windows Phone pour créer un projet d'application Silverlight pour Windows Phone comme base de démarrage de votre première application

1. Ouvrez Microsoft Visual Studio 2010 Express pour Windows Phone depuis le menu **Start | All Programs | Microsoft Visual Studio 2010 Express**.

   Visual Studio 2010 : Ouvrez Visual Studio 2010 depuis le menu **Start | All Programs | Microsoft Visual Studio 2010**.

2. Dans le menu **File**, sélectionnez **New Project**.

   Visual Studio 2010 : Dans le menu **File**, allez sur **New** puis sélectionner **Project**.

3. Sélectionnez la catégorie **Silverlight for Windows Phone** dans la liste des modèles installés visibles dans la boîte de dialogue **New Project** , puis saisissez le nom « **HelloPhone** » et cliquez sur **OK**.

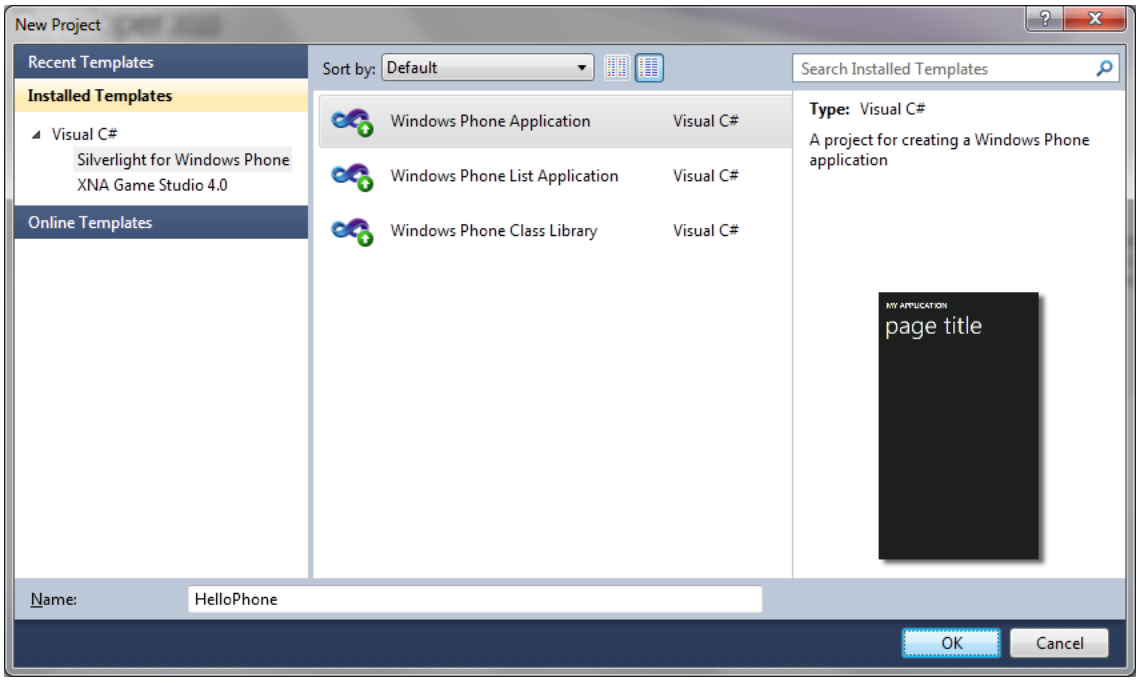

Figure 1

Créer un nouveau projet Windows Phone dans Microsoft Visual Studio 2010 Express pour Windows Phone

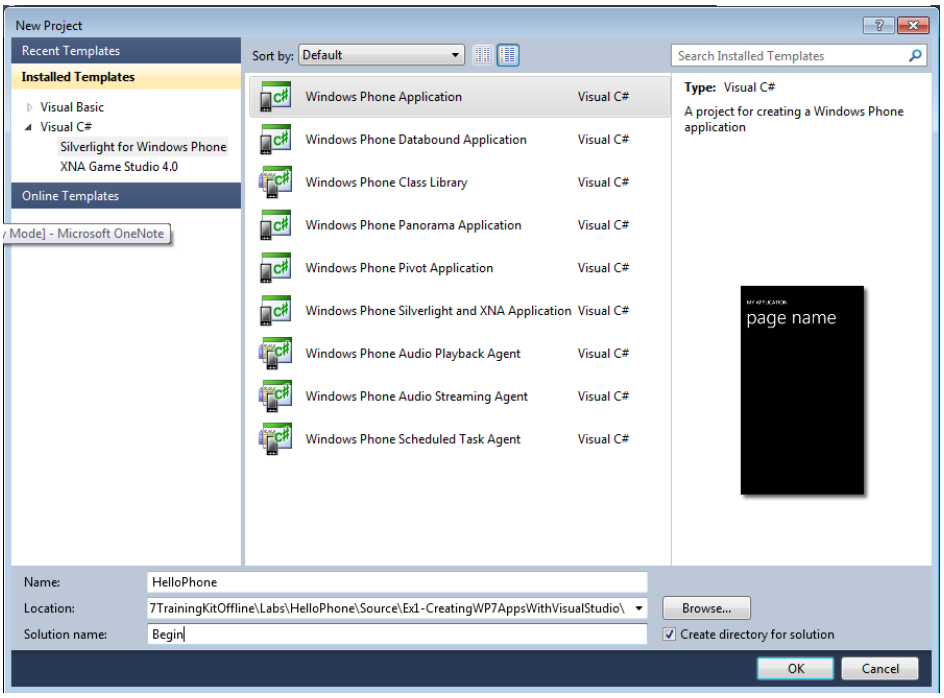

Figure 2

Créer et sauvegarder un nouveau projet d'application Windows Phone dans Microsoft Visual Studio 2010

Visual Studio va afficher ensuite une fenêtre pour vous demander pour quelle version de Windows Phone sera créé votre projet.

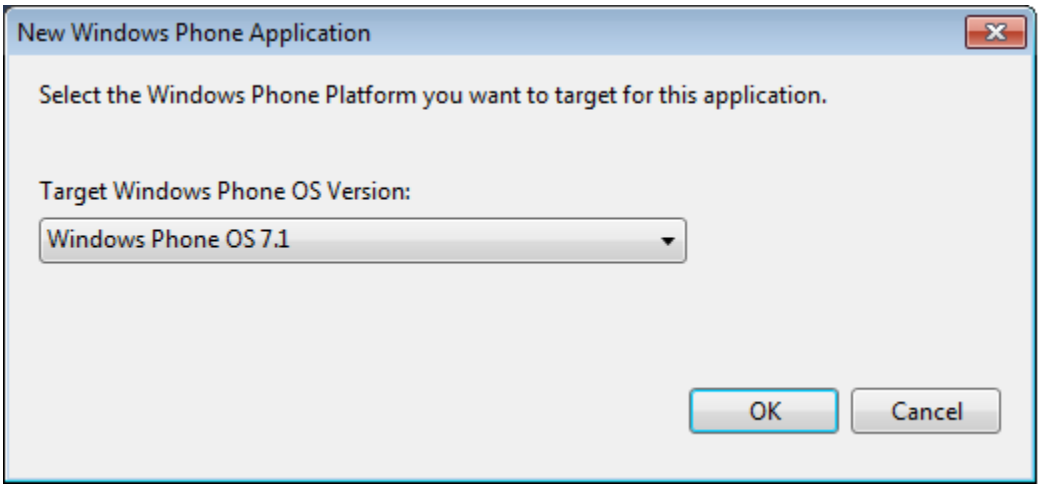

Figure 3

La fenêtre de choix de la version du système d'exploitation du téléphone cible.

4. Dans l'explorateur de solutions, regardez la structure de la solution générée par le modèle d'application Windows Phone. Toute solution Visual Studio regroupe un ensemble de projets liés ; dans le cas présent, la solution ne comprend qu'un projet Silverlight pour Windows Phone nommé « **HelloPhone** ».

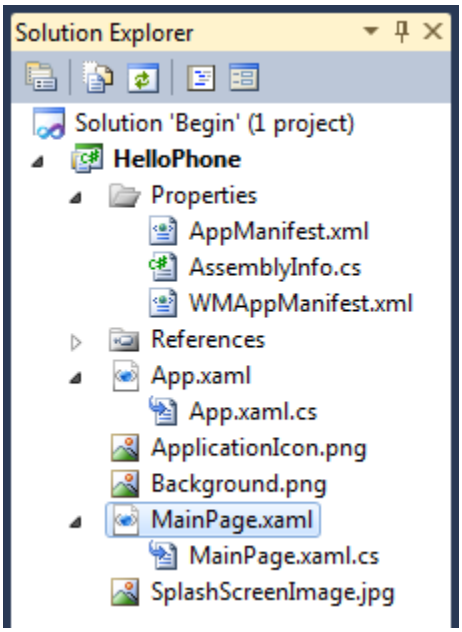

Figure 4

L'explorateur de solution présentant la structure de l'application HelloPhone

Remarque : l'explorateur de solution permet de visualiser ses éléments et réaliser des tâches d'administration de ceux-ci au sein d'une solution ou d'un projet. Pour afficher l'explorateur de solution, appuyez sur **CTRL + W, S** ou sélectionner **Other Windows | Solution Explorer** dans le menu **View.**

Le projet **HelloPhone** contient les éléments suivants :

Elément	Description
App.xaml / App.xaml.cs	Définit le point d'entrée de l'application, initialise les ressources de l'application, et affiche l'interface utilisateur
MainPage.xaml / MainPage.xaml.cs	Définit une page avec l'interface utilisateur de l'application
ApplicationIcon.png	Un fichier image représentant l'icône de l'application en cours dans la liste des applications du téléphone
Background.png	Un fichier image représentant l'icône de l'application dans le menu de démarrage
SplashScreenImage.jpg	Un fichier image représentant l'image affichée en plein écran au lancement de l'application
Properties\AppManifest.xml	Un fichier de manifeste de l'application requis pour générer le package de celle-ci
Properties\AssemblyInfo.cs	Contient le nom et la version des métadonnées embarquées dans l'*assembly* générée
Properties\WMAppManifest.xml	Un fichier de manifeste comprenant les métadonnées spécifiques à l'application Silverlight, incluant des fonctionnalités disponibles uniquement avec Silverlight pour Windows Phone
Dossier References	Une liste de librairies (*assemblies*) fournissant des services et fonctionnalités nécessaires à l'application pour fonctionner

5. Tout d'abord, faites un clic-droit sur **App.xaml** dans l'explorateur de solution, et sélectionnez **View Designer**. Notez que le fichier contient du code XAML avec un élément racine **Application** et en son sein une section **Application.Resources** contenant la définition des couleurs, motifs (*brushes*) et styles. Le code XAML initialise la propriété **RootVisual** de l'**Application** pour définir la première page de celle-ci.

App.xaml

```
<Application
    x:Class="HelloPhone.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone">

    <!--Application Resources-->
    <Application.Resources>
    </Application.Resources>

    <Application.ApplicationLifetimeObjects>
        <!--Required object that handles lifetime events for the application-->
        <shell:PhoneApplicationService
            Launching="Application_Launching" Closing="Application_Closing"
            Activated="Application_Activated" Deactivated="Application_Deactivated"/>
    </Application.ApplicationLifetimeObjects>

</Application>
```

Figure 5

Fichier App.xaml par défaut généré par le modèle d'application Windows Phone

Remarque : le fichier **App.xaml**, combiné avec le code sous-jacent du fichier **App.xaml.cs**, définit une instance de la classe **Application**. Cette classe encapsule une application Silverlight pour Windows Phone et lui fournit son point d'entrée. Elle contient également des ressources de l'application comme les couleurs, motifs (*brush*) et objets de style utilisés au travers de l'application. Vous pouvez totalement modifier l'apparence de l'application en changeant les définitions contenues dans ce fichier, comme les feuilles de style contrôlent l'apparence de pages HTML.

La propriété **RootVisual** dans la classe **Application** identifie la page de démarrage de l'application. Toute application Windows Phone dispose d'un unique élément maître dont le type est **PhoneApplicationFrame**. Cet élément intègre un ou plusieurs éléments de type **PhoneApplicationPage** présentant le contenu de l'application, et gérant la navigation entre les pages.

6. Désormais, faites un clic-droit sur **App.xaml** dans l'explorateur de solution et sélectionnez **View Code** afin d'ouvrir le fichier de code sous-jacent. Notez que cette classe dérivée d'**Application** souscrit déjà un gestionnaire pour l'évènement **UnhandledException**. Plus loin dans cet atelier, vous serez amené à modifier le gestionnaire généré par le modèle pour aller à une page d'erreur et afficher des informations à propos de l'exception.

> **Remarque :** même si la classe générée par le modèle d'application Silverlight pour Windows Phone ne comprend pas de gestionnaires pour les évènements **Startup** et **Exit**, il est toutefois possible de modifier la classe **App** et souscrire à ces évènements pour exécuter du code au cours du lancement et de la fermeture de l'application.

7. Le projet généré contient un document par défaut comprenant un markup XAML définissant l'interface utilisateur (UI) de l'application. Pour visualiser ce fichier dans le designer, double-cliquez sur MainPage.xaml dans l'explorateur de solutions.

   Par défaut, le designer affiche le document de manière fractionnée. D'une part le markup XAML et d'autre part une vue comprenant une représentation WYSIWYG des éléments de l'interface utilisateur. Hormis certains éléments intégrés au modèle pour afficher un nom et titre d'application, que vous pouvez supprimer si vous le souhaitez, le document XAML fournit une page blanche à laquelle vous êtes libre d'ajouter des contrôles pour créer votre propre interface utilisateur.

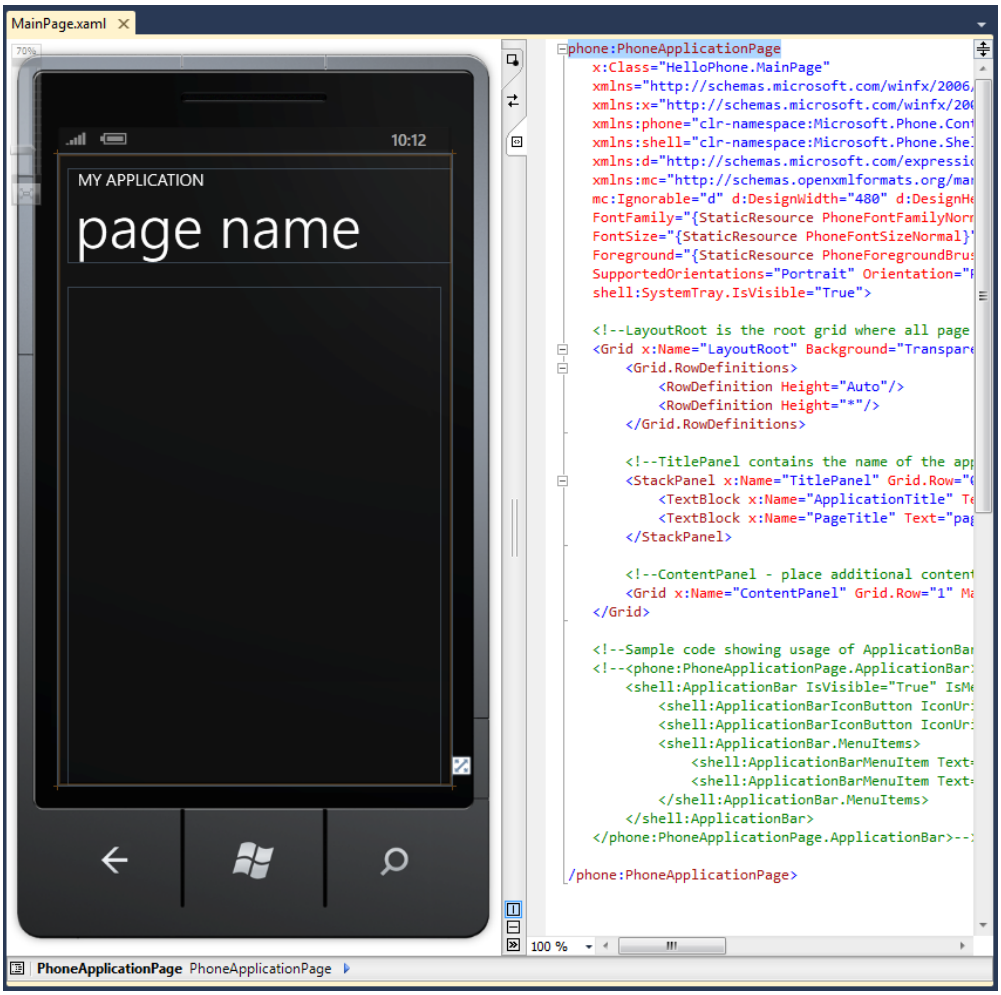

Figure 6

Le designer XAML présentant l'interface utilisateur principale de l'application

> **Note:** Extensible Application Markup Language (XAML) est un langage déclaratif. Vous pouvez créer des éléments de l'interface utilisateur dans le markup XAML, puis utiliser un fichier de code sous-jacent pour gérer des événements et manipuler les objets déclarés dans le XAML. Un langage déclaratif basé sur XAML est très intuitif pour créer des interfaces, que ce soit dans le cadre de prototypes ou d'applications de production, notamment pour les personnes ayant des compétences en design et technologies Web.

8. Le fichier **ApplicationIcon.png** contient l'icône identifiant l'application dans l'écran de démarrage rapide du téléphone. Vous pouvez double-cliquer sur l'élément dans l'explorateur de solution pour ouvrir le fichier avec une application d'édition d'image telle que **Paint.exe**.

Remarque : Dans Visual Studio 2010, double-cliquer sur le fichier icône dans l'explorateur de solution ouvre l'éditeur d'image.

9. Une application Windows Phone peut tirer parti des services fournis par la plateforme sous-jacente ou d'autres librairies. Pour cela, l'application doit référencer les librairies correspondantes implémentant ces services. Pour afficher les librairies référencées par le projet, dépliez les **References** dans l'explorateur de solution, et analysez la liste. Elle contient aussi bien des librairies Silverlight classiques que des librairies spécifiques à la plateforme Windows Phone.

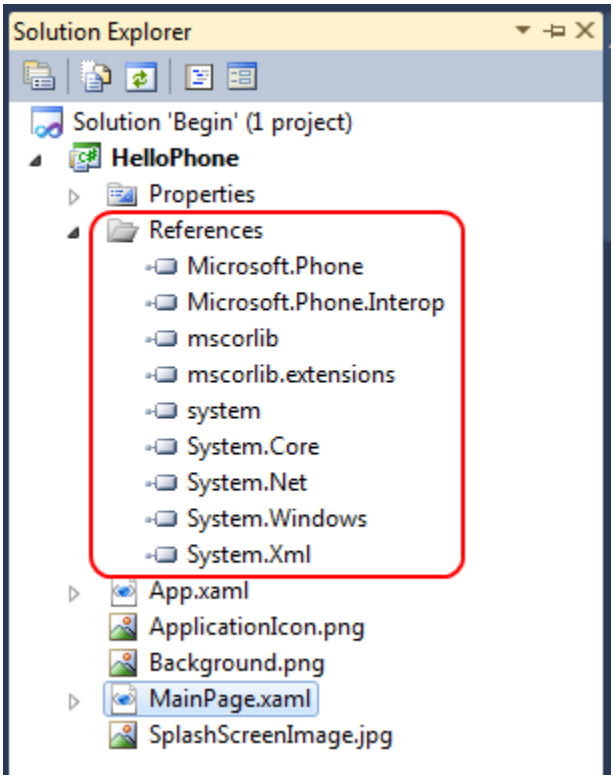

Figure 7

Les librairies référencées dans le projet au sein de l'explorateur de solution

10. Comme présenté dans l'image ci-dessous, la fenêtre **Properties** du projet est le seul moyen d'accès au fichier manifeste Windows Phone. Pour ouvrir cette fenêtre, faites un clic-droit sur le projet **HelloPhone** dans l'explorateur de solution, et sélectionnez **Properties**.

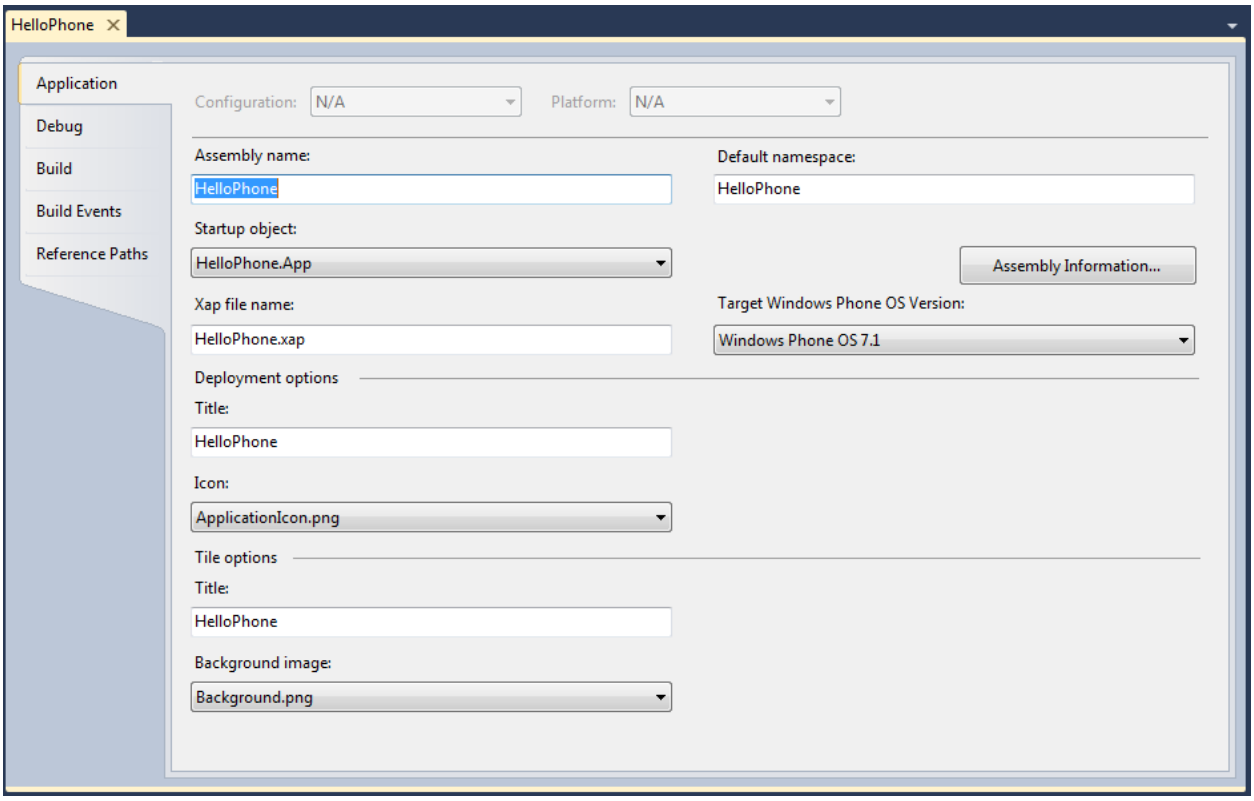

Figure 8

Propriétés du projet

Remarque : La fenêtre de propriétés du projet permet de modifier certaines propriétés spécifiques au téléphone. Ces propriétés concernent le déploiement et l'apparence de l'application sur l'appareil. Les paramètres sont stockés dans le fichier **WMAppManifest.xml**. Même si vous modifiez manuellement le fichier XML, ces modifications seront écrasées à chaque fois que vous sauvegarderez des modifications des propriétés du projet au travers de cette boite de dialogue.

Exercice 2 – Compilez et tester l'application dans l'émulateur Windows Phone

A ce stade de l'atelier, l'application ne dispose pas de beaucoup de fonctionnalités, mais suffisamment toutefois pour un premier test d'exécution ! Ainsi dans cette tâche, vous compilez l'application, la déployez dans l'émulateur Windows Phone puis l'exécutez pour comprendre le cycle de développement.

1. Dans le menu **View**, sélectionnez **Output** pour ouvrir la fenêtre en question.
2. Sélectionnez **Build Solution** dans le menu **Debug** ou appuyez sur **SHIFT + F6** pour compiler les projets dans la solution.

> **Visual Studio 2010 :** Sélectionnez **Build Solution** dans le menu **Build** ou appuyer sur **CTRL + SHIFT + B** pour compiler les projets dans la solution.

3. Observez la fenêtre **Output** et les messages générés durant le process de compilation :

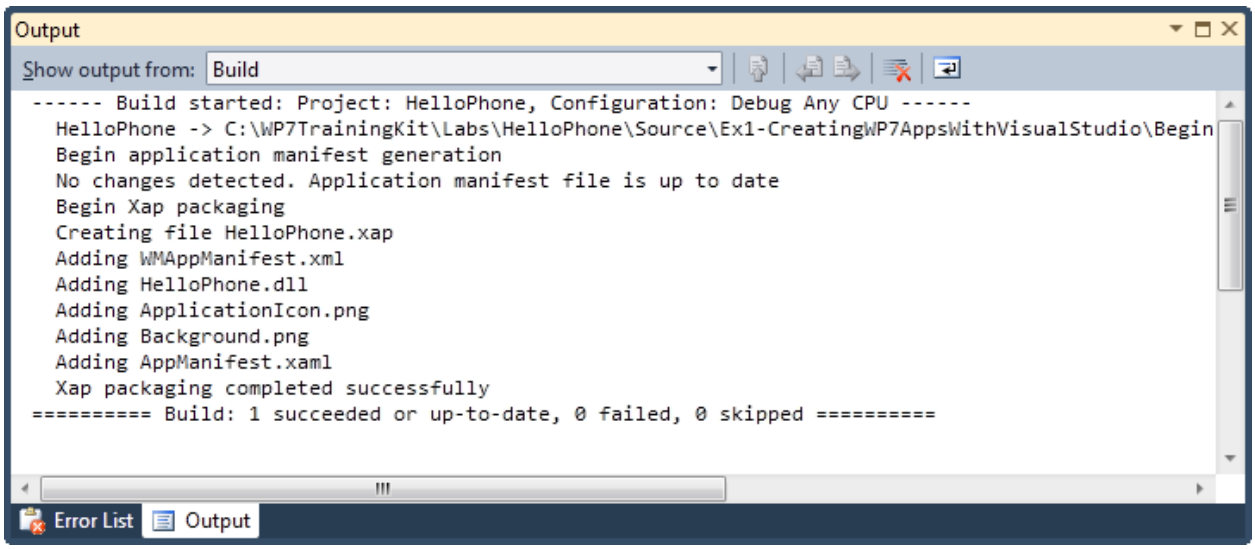

Figure 9

Compiler l'application dans Visual Studio

4. Vous ne devriez pas constater d'erreur à ce stade mais si toutefois des erreurs de compilation survenaient, elles apparaitraient dans cette fenêtre **Output**. Pour gérer ces types d'erreurs, vous pouvez utiliser la fenêtre **Error List**. Cette fenêtre affiche les erreurs, avertissements et messages générés par le compilateur dans une liste triable et filtrable selon la criticité du message. De plus, il est possible de double-cliquer sur un élément de la liste afin d'ouvrir automatiquement le code source correspondant pour identifier la source de l'erreur. Pour ouvrir la fenêtre de liste d'erreurs, allez sur **Other Windows** dans le menu **View** puis sélectionner **Error List**.

> **Visual Studio 2010 :** Pour ouvrir la liste d'erreurs, sélectionner **Error List** dans le menu **View**.

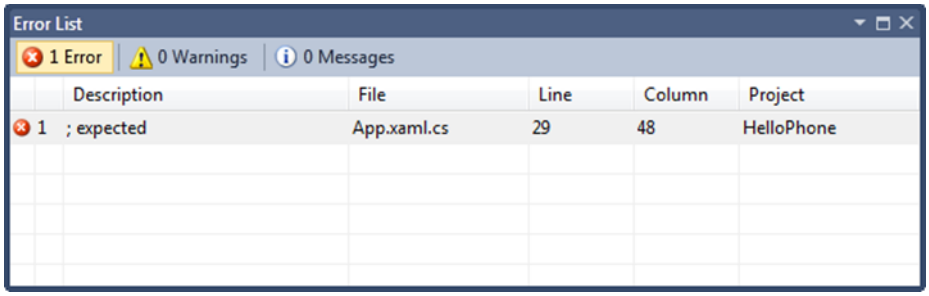

Figure 10

Fenêtre de liste des erreurs indiquant les erreurs survenues lors de la compilation

5. Vérifiez que le déploiement va se faire vers l'émulateur Windows Phone. Pour cela, assurez-vous que Windows Phone 7 Emulator est sélectionné dans la liste déroulante **Select Device** à côté du bouton de lancement du débogage de la barre d'outils.

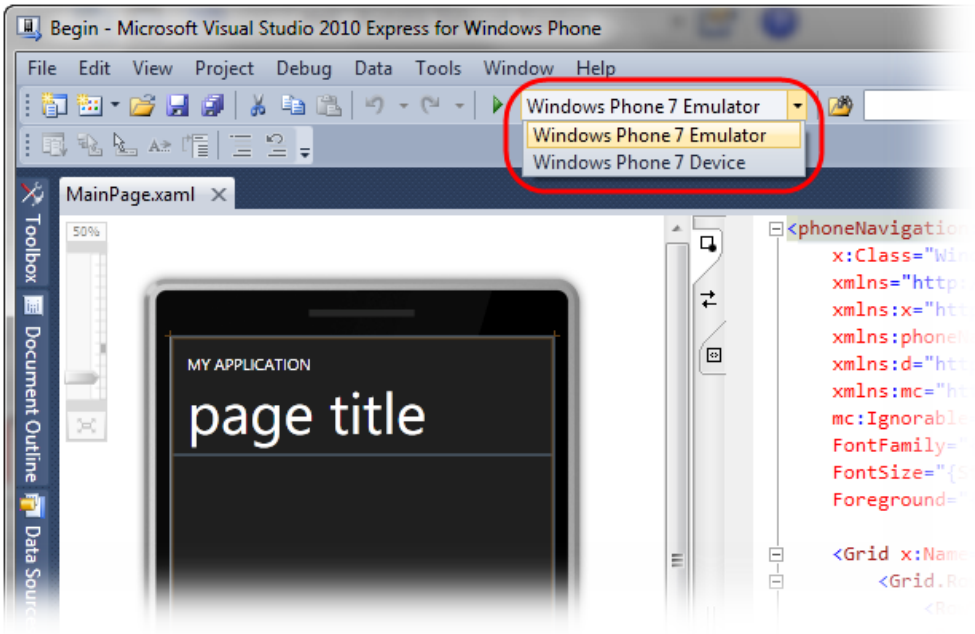

Figure 11

Choix de la cible de déploiement de l'application

Remarque : lorsque vous déployez votre application depuis Visual Studio, vous avez le choix entre le faire vers l'émulateur Windows Phone ou un appareil 'physique'.

6. Appuyez sur **F5** pour lancer l'application dans l'émulateur Windows Phone. Notez l'apparition d'une fenêtre d'émulateur de téléphone apparait, patientez alors pendant que Visual Studio installe l'environnement de l'émulateur et déploie l'image de l'application.

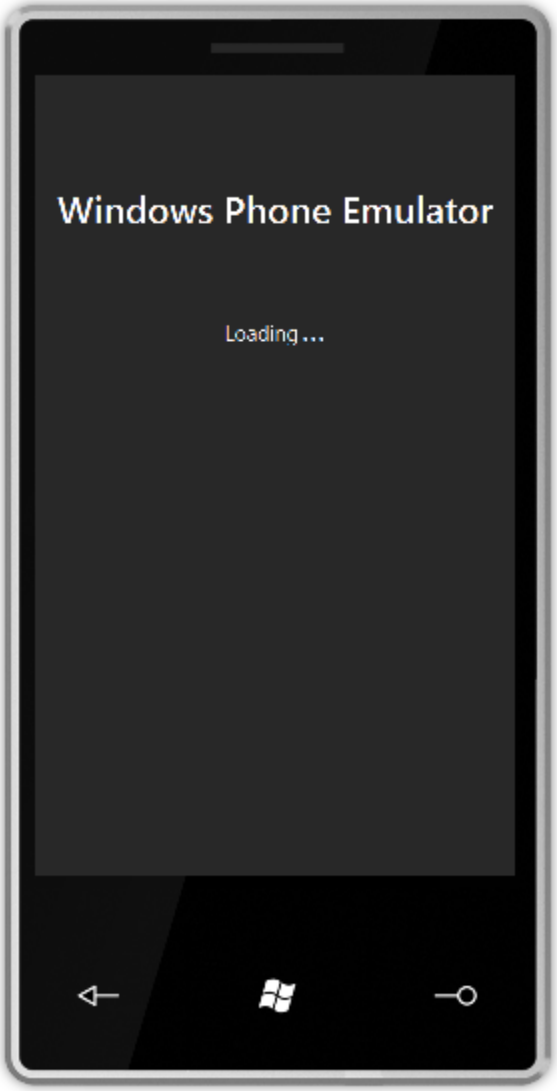

Figure 12

Déploiement d'une image de l'application vers l'émulateur Windows Phone

7. Une fois prêt, l'émulateur affiche la page de démarrage puis dans la foulée l'application

Figure 13

Exécuter l'application dans l'émulateur Windows Phone

8. Peu de choses peuvent être réalisées avec l'application à ce stade. Appuyez sur **SHIFT + F5** ou cliquez sur le bouton **Stop** de la barre d'outils pour sortir du mode debug. Ne fermez toutefois pas la fenêtre de l'émulateur.

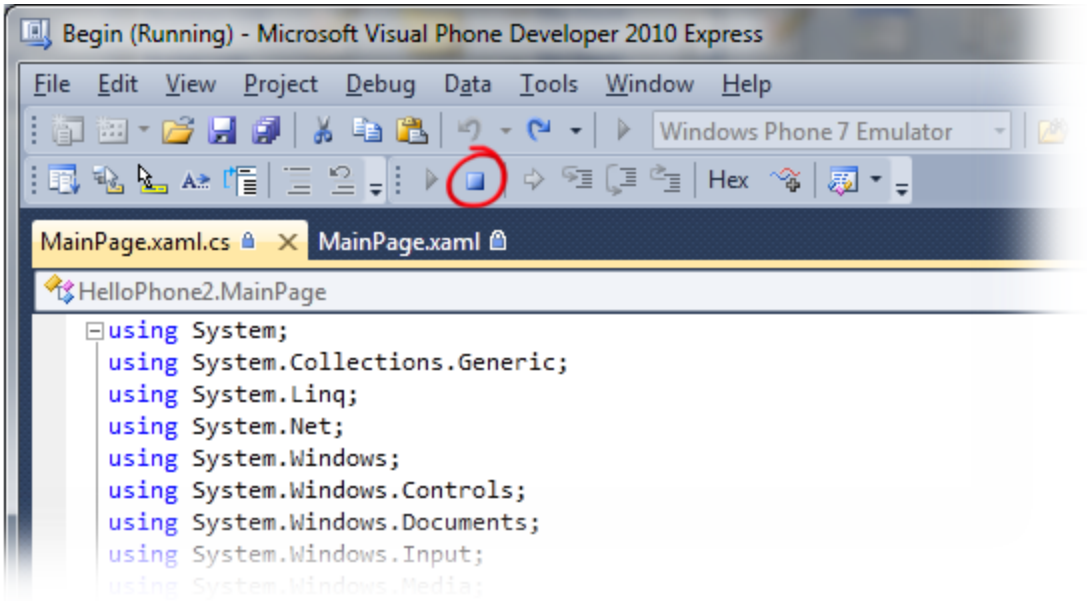

Figure 14

Terminer la session de débogage

> **Astuce :** Lorsque vous passez en mode debug, un délai est nécessaire pour monter l'environnement de l'émulateur et lancer l'exécution. Afin d'optimiser vos phases de débogage, **évitez de fermer l'émulateur** pendant que vous travaillez sur le code source dans Visual Studio. Une fois l'émulateur lancé, il est très rapide d'arrêter la session en cours, éditer le code source, puis compiler et déployer une nouvelle image de l'application pour refaire une passe de débogage.

Exercice 3 – Définir l'interface utilisateur

Vous allez créer ici les éléments d'UI de l'application, très simple, **HelloPhone**.

Une fois terminée, l'interface utilisation contient un label, une zone de texte et un bouton. Pour utiliser l'application, vous saisissez du texte dans la zone appropriée puis, lorsque vous cliquez sur le bouton, l'application affiche le texte saisi préalablement. Cela donnera la représentation suivante :

Figure 15
L'interface utilisateur de l'application

1. Dans l'explorateur de solution, double-cliquez sur **MainPage.xaml** pour ouvrir le fichier dans le designer.

> **Remarque :** Le designer fournit 2 vues séparées d'affichage des fichiers XAML, les vues **Design** et **XAML**. En mode Design, vous disposez d'un espace où vous pouvez glisser-déposer des contrôles de la boîte à outils, tout autant que sélectionner, redimensionner, modifier et fixer les propriétés des contrôles existants sur la surface de design. En mode XAML, un éditeur permet simplement d'éditer le XAML dans la page. Les onglets de la fenêtre de design permettent de choisir le mode désiré. Un mode mixte est également disponible, affichant les 2 vues simultanément. En mode mixte, un bouton pour Déplier/Replier le panneau est disponible pour maximiser la visibilité dans le mode d'affichage courant.

2. Alors que l'IDE (Environnement de développement intégré) permet de manipuler graphiquement les objets, dans cette tâche, vous vous efforcerez à éditer manuellement le XAML. Une fois terminé, vous pourrez voir le résultat de votre travail au travers du mode **Design**. Afin de passer en vue XAML dans l'éditeur et de maximiser la visibilité, double-cliquez

sur l'onglet XAML sur le bord droit de la fenêtre du designer. En cas de difficulté, déplacez le curseur de la souris sur chaque onglet afin de faire s'afficher une info-bulle l'identifiant.

Remarque: si votre designer est affiché en mode horizontal, les onglets sont situés au niveau du bord inférieur de la fenêtre.

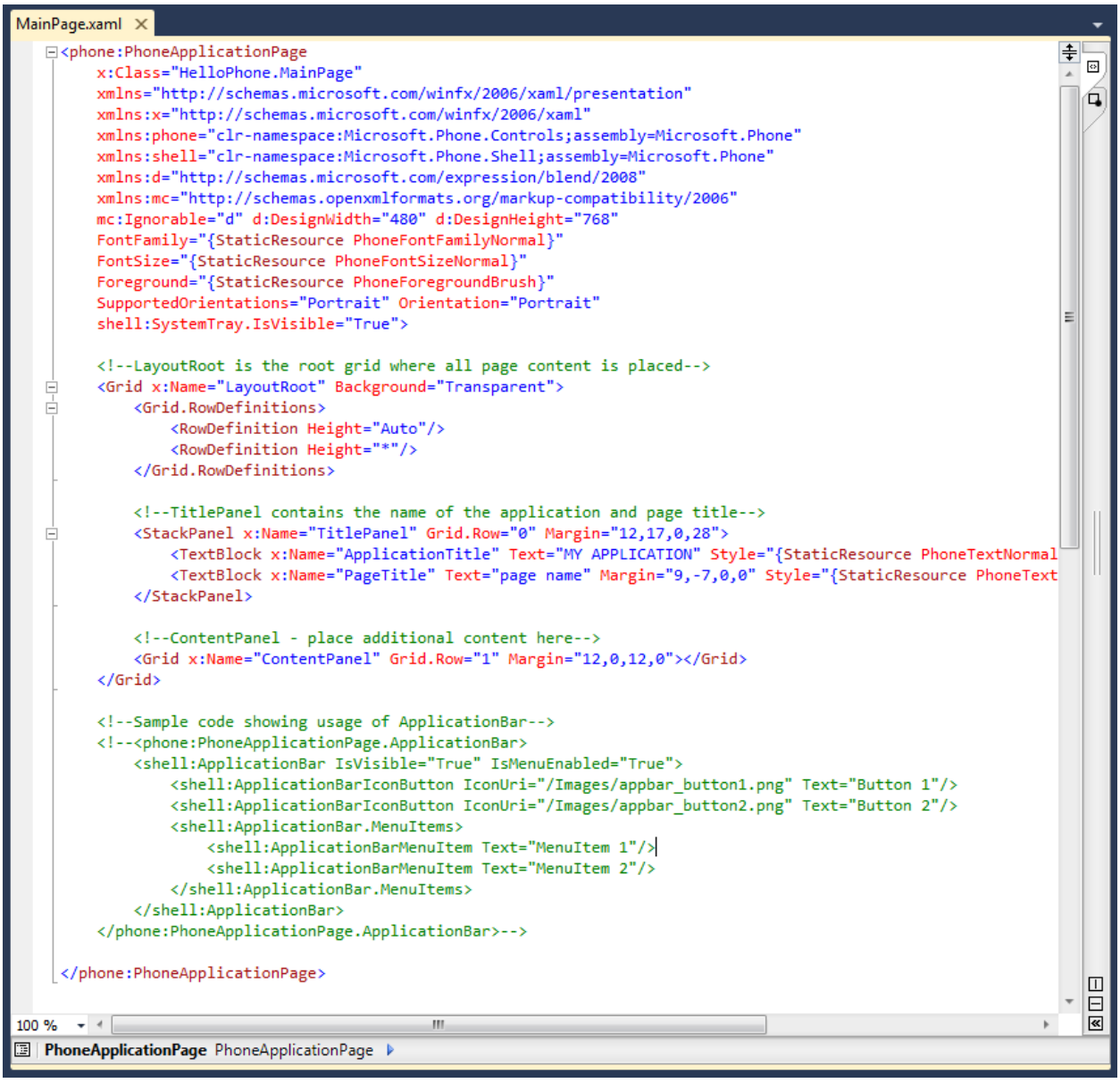

Figure 16

Le designer XAML montrant les vues XAML

3. Au sein du markup XAML généré par le modèle par défaut d'application Windows Phone, localisez l'élément **Grid** nommé **LayoutRoot**. Son but est d'organiser les éléments sur la page. A l'intérieur de sa propriété **RowDefinition**, insérez une ligne entre les 2 existantes afin d'y fixer à ***Auto*** sa propriété **Height**. Cette ligne incluera sous peu une zone de texte et un bouton.

XAML

```
  ...
  <Grid x:Name="LayoutRoot" Background="Transparent">
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="*"/>
    </Grid.RowDefinitions>
     ...

  </Grid>
</navigation:PhoneApplicationPage>
```

Remarque : un élément **Grid** est un élément de mise en page opérant comme un containeur pour d'autres contrôles. Son but principal est de positionner et organiser les contrôles '*fils*'. Plusieurs contrôles de mise en page existent afin de gérer différents comportements :

Canvas: Définit une zone au sein de laquelle l'on peut explicitement placer des éléments au travers de coordonnées relatives à l'espace en question.

Grid: Définit une zone de grille flexible compose de lignes et colonnes.

StackPanel: Organise les éléments au sein d'une seule ligne pouvant être orientée horizontalement ou verticalement.

Pour plus d'informations, se rendre sur Silverlight Layout System.

4. Notez que l'élément racine **Grid** contient d'autres éléments imbriqués dont la position dans la grille est définie par une propriété **Grid.Row**. Localisez l'élément **StackPanel** nommé *TitlePanel* et placez le dans la ligne *0* ; cet élément porte un libellé pour cette page. Indiquez "w*indows phone 7 series*" comme propriété **Text** du premier élément **TextBlock**. De la même manière, indiquez "*hello phone*" comme propriété **Text** du second élément **TextBlock.**

```
<phone:PhoneApplicationPage
    x:Class="HelloPhone.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="windows phone 7 series" Style="
            <TextBlock x:Name="PageTitle" Text="hello phone" Margin="9,-7,0,0" Style="
        </StackPanel>
```

Figure 17

Définir un libellé pour l'application

> **Remarque:** pour assigner des éléments fils à une ligne donnée dans la grille, chacun de ces éléments doit spécifier une valeur pour la propriété **Grid.Row**, considérée comme une propriété attachée. Les propriétés attachées permettent à différents éléments fils de spécifier des valeurs pour une propriété définie dans un élément parent ; dans ce cas, la propriété **Row** appartient à l'élément **Grid**.

5. Localisez maintenant l'élément **Grid** nommé *ContentPanel*, assignez-le à la ligne *1* qui doit être vide, et collez le markup XAML suivant (surligné en bleu) à l'intérieur de l'élément.

XAML

```
...
<Grid x:Name="LayoutRoot" Background="{StaticResource PhoneBackgroundBrush}">
  ...
  <!--ContentPanel - place additional content here-->
  <Grid x:Name="ContentPanel" Grid.Row="1">
<Grid.RowDefinitions>
            <RowDefinition Height="*"/>
```

```
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <TextBox Grid.Row="0" Name="MessageTextBox" FontSize="{StaticResource
PhoneFontSizeExtraLarge}" Margin="20,20,10,20" />
        <Button Grid.Row="1" Name="ClickMeButton" Content="click me"
                HorizontalAlignment="Center" Padding="4" Margin="10,20,20,20"
                Click="ClickMeButton_Click" />
  </Grid>

</Grid>
...
```

> **Remarque:** Une **Grid** organise ses contrôles fils sur la page en se basant sur la largeur de chaque colonne telle que définie dans sa collection **ColumnDefinitions**. Notez que la largeur de la première colonne est spécifiée par *, garantissant ainsi qu'elle utilise tout l'espace inutilisé dans la ligne suivante après allocation de toutes les autres colonnes, et que la largeur de la seconde colonne soit fixée à **Auto** afin d'ajuster la taille de colonne à son contenu.

6. Pour finaliser le design de la page, ajoutez une troisième ligne intégrant la zone de texte affichant le message saisi par l'utilisateur. Pour créer cette ligne, insérez le markup XAML suivant (surligné en bleu) juste après la balise de fin de la grille externe.

XAML

```
...
<Grid x:Name="LayoutRoot" Background="{StaticResource PhoneBackgroundBrush}">
  ...

  <Grid Grid.Row="2">
    <TextBlock Name="BannerTextBlock" Style="{StaticResource
PhoneTextBlockBase}"
               Foreground="#FFFF9A00" HorizontalAlignment="Stretch"
               TextWrapping="Wrap" TextAlignment="Center" FontWeight="Bold" />
  </Grid>
</Grid>
...
```

7. Cliquez sur l'onglet **Design** situé sur le bord droit de la fenêtre pour passer en mode design. Notez l'affichage des contrôles sur la page, qui doit correspondre à l'image ci-dessous :

> **Remarque :** Si vous avez configuré le designer en mode vue horizontale, les onglets sont situés au niveau du bord inférieur de la fenêtre.

Figure 18

Affichage de la page en mode Design

Exercice 4 – Gérer les évènements depuis l'interface utilisateur

Dans cette tâche, vous définissez la gestion des évènements liés aux actions effectuées depuis l'interface utilisateur, c'est-à-dire en l'occurrence le clic sur le bouton. Cette gestion d'événement est définie en utilisant du code dans le fichier de code sous-jacent de la page.

1. Assurez-vous que vous être en mode **Design**. Pour cela, double-cliquez sur l'onglet Design sur le bord droit de la fenêtre. En cas de difficulté, déplacez le curseur de la souris sur chaque onglet afin de faire s'afficher une info-bulle l'identifiant.
2. Sélectionnez en cliquant dessus le bouton "*click me*" puis appuyer sur **F4** pour ouvrir la fenêtre de propriétés.
3. Dans les propriétés, cliquez sur l'onglet **Events** pour afficher une fenêtre listant les événements disponibles. Localisez l'événement **Click** dans la liste puis saisissez **ClickMeButton_Click** dans la zone de texte située à côté de l'événement. Appuyer sur **ENTER** pour générer un gestionnaire d'événement avec ce nom et ouvrir le fichier de code sous-jacent afin d'afficher la méthode générée par Visual Studio.

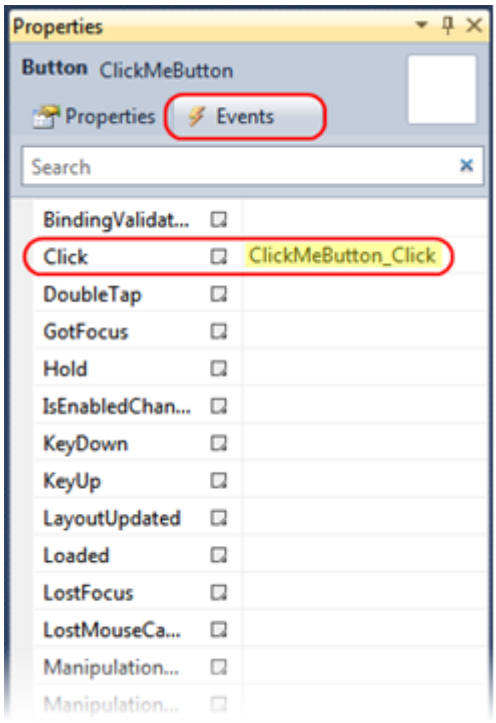

Figure 19

Création d'un gestionnaire d'événement pour le bouton

```
<TextBox Grid.Column="0" Name="MessageTextBox" FontSize="{StaticResource Phc
<Button Grid.Column="1" Name="ClickMeButton" Content="click me"
        HorizontalAlignment="Right" Padding="4" Margin="10,20,20,20"
        Click="ClickMeButton_Click" />
```

Figure 20

Vue XAML montrant le nouveau gestionnaire associé à l'événement Click.

Remarque : Une autre méthode est possible pour créer le gestionnaire d'événement. Dans Visual Studio, vous pouvez double-cliquer sur un contrôle dans le designer pour créer un gestionnaire pour son événement par défaut, en l'occurrence l'événement **Click** pour les contrôles de type bouton.

4. L'implémentation de la méthode (à date une méthode vide) et dans le fichier **MainPage.xaml.cs**. Insérez le code suivant dans le corps de la méthode **ClickMeButton_Click**.

C#

```
private void ClickMeButton_Click(object sender, RoutedEventArgs e)
{
    BannerTextBlock.Text = MessageTextBox.Text;
    MessageTextBox.Text = String.Empty;
}
```

Remarque: Le code lit le texte de la zone de texte saisi par l'utilisateur puis affiche la bannière avec ce texte.

Exercice 5 – Gérer les erreurs dans l'application

Dans cet exercice, vous mettez à jour l'application pour afficher une page d'erreur à chaque fois qu'une exception non gérée survient dans l'application. Elle permet de voir comment améliorer la gestion d'erreurs, mais aussi comment ajouter des pages à son application et naviguer entre elles.

> **Remarque:** Indépendamment de la présence d'un gestionnaire d'erreurs, votre application doit intégrer une manipulation propre des exceptions pouvant survenir.

1. Tout d'abord, ajoutez une nouvelle page au projet. Dans l'explorateur de solution, faites un clic-droit sur le nœud du projet **HelloPhone**, et sélectionnez **Add \ New Item**. Dans la boite de dialogue **Add New Item**, sélectionnez **Windows Phone Portrait Page** dans la liste des modèles, saisissez le nom **ErrorPage.xaml** puis cliquez sur **Add**.

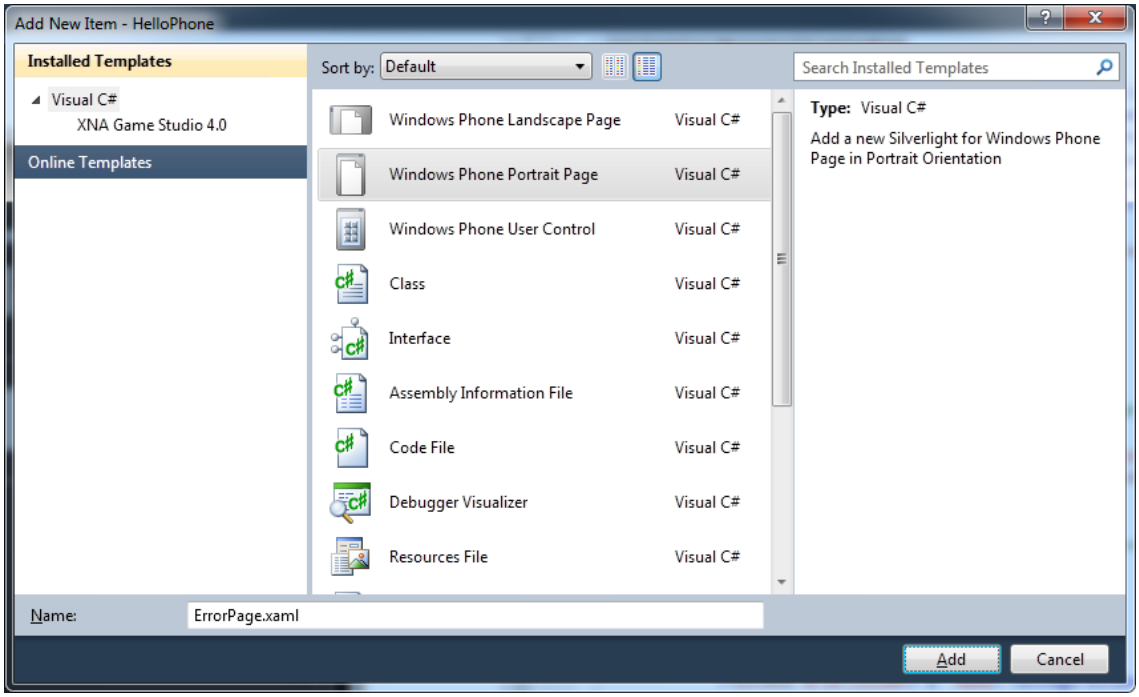

Figure 21

Ajouter une nouvelle page à un projet

2. Dans **ErrorPage.xaml**, localisez l'élément *LayoutRoot* **Grid** et remplacez ses contrôles fils avec le markup XAML surligné en bleu. Ce code XAML définit un titre d'application et de page, tous

deux nommés erreur. Il définit également un objet **TextBlock** désigné par **x:Name="ErrorText"** qui récupérera le texte de l'erreur de toute future exception survenant.

XAML

```
...
<Grid x:Name="LayoutRoot" Background="Transparent">
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="*"/>
  </Grid.RowDefinitions>

  <!--TitlePanel contains the name of the application and page title-->
  <StackPanel x:Name="TitlePanel" Grid.Row="0">
    <TextBlock x:Name="ApplicationTitle" Text="windows phone 7 series"
Style="{StaticResource PhoneTextNormaltyle}"/>
    <TextBlock x:Name="PageTitle" Text="error" Style="{StaticResource
PhoneTextTitle1Style}"/>
  </StackPanel>

  <Grid x:Name="ContentPanel" Grid.Row="1">
    <Border BorderBrush="White">
      <TextBlock x:Name="ErrorText" Style="{StaticResource
PhoneTextSmallStyle}" TextWrapping="Wrap" />
    </Border>
  </Grid>
</Grid>
...
```

3. Appuyez maintenant sur F7 pour ouvrir le fichier de code sous-jacent de la nouvelle page ou bien faites un clic-droit sur **ErrorPage.xaml** dans l'explorateur de solution et sélectionnez **View Code**. Puis insérez l'extrait de code suivant dans la classe **ErrorPage** à l'endroit surligné. Cela créé un objet **Exception** relié à ErrorText.Text lors de la navigation sur la page.

C#

```
public partial class ErrorPage : PhoneApplicationPage
{
  public ErrorPage()
  {
    InitializeComponent();
  }

  public static Exception Exception;

  // Executes when the user navigates to this page.
```

```
  protected override void OnNavigatedTo(System.Windows.Navigation.
NavigationEventArgs e)
  {
    ErrorText.Text = Exception.ToString();
  }
}
```

4. Une fois la page prête, créez un gestionnaire d'événement pour renvoyer vers la page d'erreur et afficher un message à chaque exception non gérée. Dans l'explorateur de solution, faites un clic-droit que **App.xaml**, et sélectionnez **View Code** pour ouvrir le fichier de code associé à la classe **Application**.

5. Localisez le gestionnaire d'événement **Application_UnhandledException** et insérez l'extrait de code suivant à l'endroit surligné juste avant l'accolade fermante.
Application_UnhandledException est un 'réceptacle' de sécurité où toutes les exceptions non gérées de l'application sont dirigées. Comme vous pouvez le voir, vous rattachez l'objet exception à l'objet **ErrorPage.Exception** et lorsque vous arrivez vers la page d'erreur, la chaine de caractère de l'objet exception est récupérée (**Exception.ToString();**) et affichée sur la page. Cela s'avérera très utile lorsque vous commencerez à déboguer vos applications sur un véritable téléphone.

C#

```
// Code to execute on Unhandled Exceptions
private void Application_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
{
  if (System.Diagnostics.Debugger.IsAttached)
  {
    // An unhandled exception has occurred, break in the debugger
    System.Diagnostics.Debugger.Break();
  }
  e.Handled = true;
  ErrorPage.Exception = e.ExceptionObject;
  (RootVisual as Microsoft.Phone.Controls.PhoneApplicationFrame).Source =
      new Uri("/ErrorPage.xaml", UriKind.Relative);
}
```

> **Remarque :** Le gestionnaire généré par le modèle contient du code vérifiant si le code est en cours de débogage, et dans ce cas, stoppe l'exécution dans le débogueur à chaque exception non gérée. Cela permet d'analyser l'état de l'application and déterminer la cause de l'exception.

Le code inséré passe la propriété **Handled** à *true* afin d'éviter tout traitement supplémentaire de l'exception après que le gestionnaire **UnhandledException** ait terminé. Il sauvegarde ensuite l'information à propos de l'exception non gérée dans un membre statique de la classe **ErrorPage** et fixe la propriété **Source** du cadre pour afficher la page d'erreur. A chaque fois que vous fixez la propriété **Source** à une valeur différente du contenu affiché, le cadre renvoie vers le nouveau contenu.

Exercice 6 – Vérification

Dans cette tâche, vous compilez, déployez et exécutez l'application dans l'émulateur Windows Phone afin de le tester son bon fonctionnement. En complément, vous utilisez un point d'arrêt et le débogueur pour naviguer dans le code source, et examinez la valeur des variables. Ceci vous donne un bref aperçu de la façon dont vous pouvez utiliser Visual Studio pour déboguer les applications exécutées dans l'émulateur.

1. Si nécessaire, ouvrez à nouveau le fichier **MainPage.xaml.cs**. Pour cela, faites un clic-droit sur ce fichier dans l'explorateur de solution et sélectionnez **View Code**.
2. Positionnez maintenant un point d'arrêt pour stopper l'exécution au niveau de l'appel du gestionnaire d'événement pour le bouton "*Click Me*". Pour positionner le point d'arrêt, localisez la première ligne de la méthode **ClickMeButton_Click** dans le fichier source et cliquez dans la marge grise située à gauche de la fenêtre d'édition, dans l'axe de cette ligne. Un cercle rouge ( ) indique la position du point d'arrêt inséré. Vous pouvez également procéder en sélectionnant la ligne dans la fenêtre d'édition puis en appuyant sur **F9**.

```
using Microsoft.Phone.Controls;

namespace HelloPhone
{
    public partial class MainPage : PhoneApplicationPage
    {
        public MainPage()
        {
            InitializeComponent();

            SupportedOrientations = SupportedPageOrientation.Portrait | Sup
        }

        private void ClickMeButton_Click(object sender, RoutedEventArgs e)
        {
            BannerTextBlock.Text = MessageTextBox.Text;
            MessageTextBox.Text = String.Empty;
        }
    }
}
```

Figure 22

Positionnement de points d'arrêt dans Visual Studio

> **Remarque :** Pour activer/désactiver un point d'arrêt, cliquez sur le symbole dans la marge ou sélectionnez la ligne visée et appuyer sur **F9**.

3. Appuyez sur **F5** pour compiler et déployer l'application vers l'émulateur Windows Phone, et commencez à déboguer. Attendez que l'application se lance et affiche sa page principale.

> **Remarque :** Si vous laissez fonctionner l'émulateur après l'avoir utilisé plus tôt au cours de l'atelier, le lancement de l'application ne doit prendre que quelques secondes ; si vous aviez fermé l'émulateur, cela pourra prendre plus de temps.

4. Dans la fenêtre de l'émulateur, saisissez du texte dans la zone de texte puis cliquez sur le bouton situé à côté. De retour dans Visual Studio, notez que l'exécution s'est arrêtée au point d'arrêt précédemment positionné, et que la prochaine ligne de code en exécution apparait sur lignée en jaune.

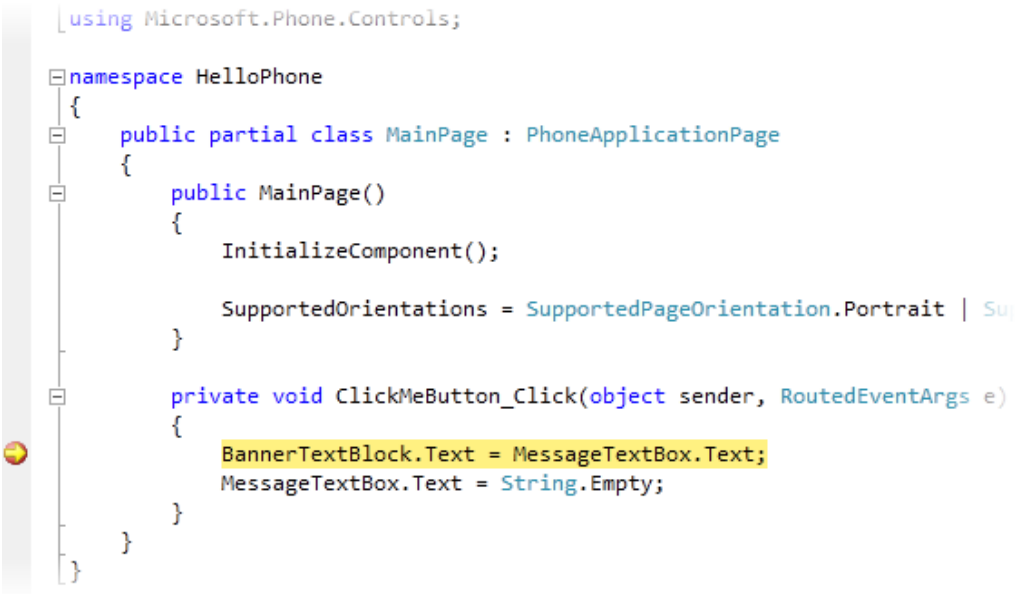

Figure 23

Pas-à-pas au travers du débogueur

5. Examinez le contenu actuel de la zone de texte dans le débogueur. Pour cela, dans la fenêtre source, positionnez le curseur de la souris sur la propriété **MessageTextBox.Text.** Une fenêtre info-bulle (généralement appelée *datatip*) apparait pour afficher la valeur actuelle de la propriété, qui doit correspondre au texte saisi dans la fenêtre de l'émulateur. Assurez-vous que le curseur est bien positionné au-dessus du champ **Text** ; dans le cas contraire, la datatip affiche l'information relative à l'objet **MessageTextBox**.

Figure 24

Analyse de la valeur des variables dans le débogueur

> **Remarque :** Vous pouvez analyser l'état de n'importe quel objet en plaçant le curseur de la souris sur la variable correspondante.

6. Appuyez sur **F10** pour exécuter l'instruction actuelle et passer à la suivante. Afficher le datatip pour la propriété **BannerTextBlock.Text** afin de confirmer que sa valeur correspond maintenant bien à celle de la zone de texte.

> **Remarque:** Quand vous utilisez **F10**, vous indiquez au débogueur de passer par-dessus en cours. Vous pouvez aussi utiliser **F11** pour passer sur l'instruction courante. Dans ce cas, si cette instruction engendre un appel de méthode, le débogueur passe à la méthode cible pour vous permettre de la déboguer.

7. Appuyez à nouveau sur **F10** pour exécuter l'instruction suivante et vider le contenu de la zone de texte. Une fois encore, affichez le datatip pour la propriété **MessageTextBox.Text**, et vérifiez qu'il est bien vide.
8. Enfin, appuyez sur **F5** pour reprendre l'exécution de l'application. Retournez à l'émulateur Windows Phone. Si nécessaire, fermez l'écran de verrouillage dans la fenêtre de l'émulateur.

Figure 25
Ecran de verrouillage de l'émulateur

Astuce : En fonction des paramètres de configuration, l'émulateur Windows Phone pourrait afficher un écran de verrouillage lorsque l'application tourne au ralenti pendant une courte période. Pour restaurer l'application, faites un geste sec (cliquez sur la surface de la fenêtre de l'émulateur et déplacez rapidement le curseur vers le haut de l'écran, le bouton de la souris étant toujours pressé) pour déverrouiller l'écran. Notez que la bannière dans la fenêtre de l'émulateur affiche désormais le texte saisi précédemment.

Figure 26

L'application HelloPhone exécutée dans l'émulateur Windows Phone

9. Cliquez sur le bouton **Back** dans l'émulateur pour revenir à la page précédente. Notez qu'en faisant cela, la session de débogage stoppe et l'émulateur affiche alors son menu de démarrage car vous sortez de la première et unique page de l'application et qu'il n'y a pas d'autre application en cours d'exécution.

> **Astuce** : Pour reprendre le débogage après la fin de la session en cours, appuyez sur **F5** pour lancer à nouveau l'application et relancer le débogueur. Notez cependant que cette action redémarre l'application et donc que son état précédent ne sera plus disponible.
>
> Fermer l'émulateur stoppe l'application et désactive le débogueur. En faisant cela, le débogueur Visual Studio de désactive et affiche un message indiquant qu'il a perdu la connexion avec le terminal.

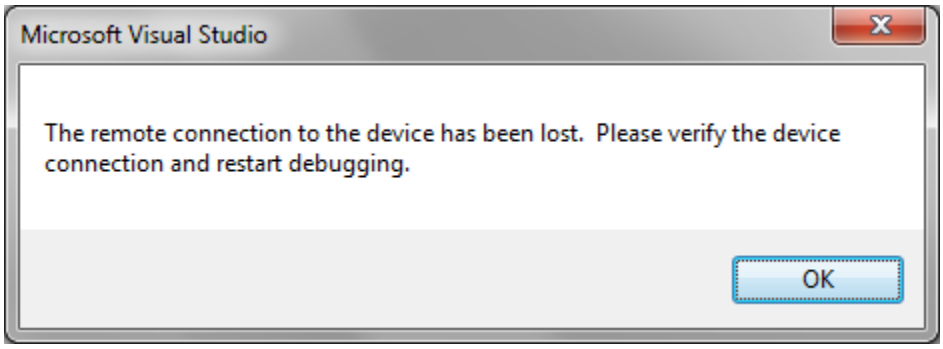

Figure 27

Une session de débogage s'arrête lorsque vous sortez de l'application

Exercice 2: Utiliser Expression Blend pour définir l'expérience utilisateur (UX) de votre application Windows Phone

Exercice 1 – Création d'un bouton personnalisé avec Expression Blend

Nativement, les contrôles Silverlight séparent leur logique de leur apparence en utilisant des modèles. Un **ControlTemplate** spécifie les structures et comportements visuels d'un contrôle. Vous pouvez personnaliser l'apparence de la plupart des contrôles en modifiant leurs **ControlTemplate** par défaut**.** Cela permet de modifier l'apparence d'un contrôle sans en changer les fonctionnalités ; par exemple, il est possible de définir des boutons ronds plutôt que carrés tout en conservant leurs fonctionnalités, telles que l'événement **Click**.

Dans cette tâche, vous ouvrez dans Expression Blend le projet Visual Studio créé dans l'exercice précédent et remplacez le **ControlTemplate** du bouton pour modifier son apparence. Via la création d'un **ControlTemplate** en XAML, vous pouvez modifier l'apparence d'un contrôle sans écrire de code.

Figure 28
Modification du 'look and feel' des contrôles

1. S'il n'est pas déjà ouvert, lancez Microsoft Visual Studio 2010 Express pour Windows Phone depuis le menu **Start | All Programs | Microsoft Visual Studio 2010 Express**

   Visual Studio 2010 : Ouvrez Visual Studio 2010 depuis le menu **Start | All Programs | Microsoft Visual Studio 2010**.

2. Si vous avez finalisé toutes les étapes de l'exercice précédent, vous pouvez continuer avec la solution créée à cette occasion.
3. Ouvrez maintenant la solution dans **Expression Blend**. Vous pouvez le faire depuis Visual Studio en faisant un clic-droit sur **MainPage.xaml** dans l'explorateur de solution, puis en sélectionnant **Open in Expression Blend**.
4. Assurez-vous que **MainPage.xaml** est ouvert dans la fenêtre de design (si ce n'est pas le cas, double-cliquez sur le fichier pour l'ouvrir) et que l'espace de travail courant est en mode **Design**. Pour cela, vérifier que l'option **Design** est cochée dans le sous-menu **Workspaces** du menu **Window**.

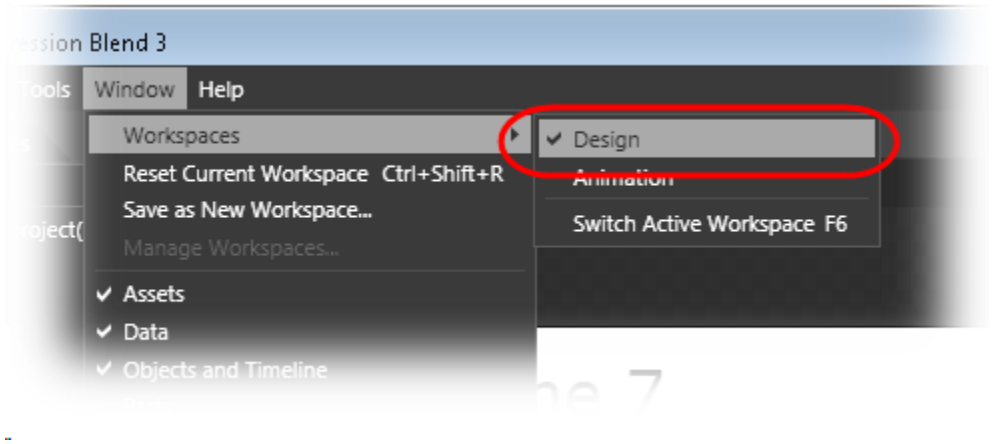

Figure 29

Sélection du l'espace de travail courant dans Expression Blend

> **Remarque :** L'espace de travail (Workspace) dans Expression Blend correspond à l'ensemble des éléments d'interface visuels, incluant les outils, les éléments de configuration, de création des vues et menus.

5. Dans la fenêtre de design, faites un clic-droit sur le bouton "*click me*", allez sur **Edit Template** et sélectionnez **Create Empty**.
6. Dans la boite de dialogue **Create ControlTemplate Resource**, indiquez le nom *FancyButton*, laissez la valeur actuelle de "**This document**" dans l'option **Define in**, puis cliquez sur **OK**.

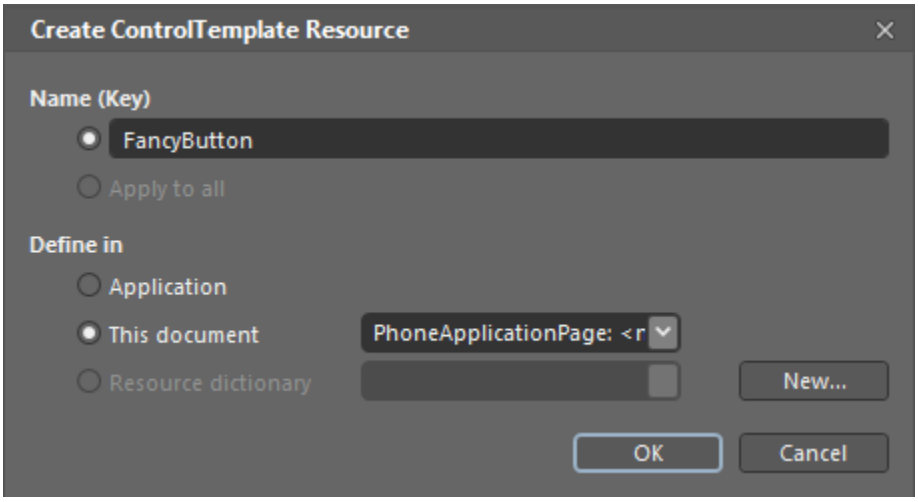

Figure 30

Création d'un modèle de contrôle vide pour le bouton

> **Remarque :** Les ressources permettent de réutiliser simplement les objets et valeurs fréquemment utilisées. Vous pouvez créer des définitions pour les éléments fréquemment utilisés comme les modèles de contrôles, styles, motifs (*brushes*), couleurs et animations, et les stocker dans des dictionnaires de ressources. Un dictionnaire de ressource est un dictionnaire d'objets utilisables dans du code et du XAML. Vous pouvez créer des dictionnaires de ressources à plusieurs niveaux : application ou pages.
>
> La boîte de dialogue ci-dessus permet de définit le modèle de contrôle au niveau **Application**, auquel cas cela est stocké dans **App.xaml,** et peut être réutilisé dans d'autres pages de l'application, à moins que vous le stockiez dans **This document** afin qu'il ne puisse être utilisé que dans la même page.

7. Si le panneau **Objects and Timeline** n'est pas visible, sélectionnez-le depuis le menu **Window** pour l'afficher. Si nécessaire, vous pouvez remettre le Workspace à son état par défaut en sélectionnant **Window | Reset Current Workspace**.

> **Remarque :** Quand vous créez un nouveau modèle, Expression Blend entre dans un mode permettant de le visualiser et l'éditer. Dans le panneau **Objects and Timeline**, le terme **Template** à la racine de la nouvelle arborescence indique le niveau actuel édité.

8. Modifiez le mode d'affichage du modèle. Pour cela, dans le panneau **Objects and Timeline**, faîtes un clic-droit sur l'élément **Grid** du **Template**, et sélectionnez **Border** au sein du sous-menu **Change Layout Type**.

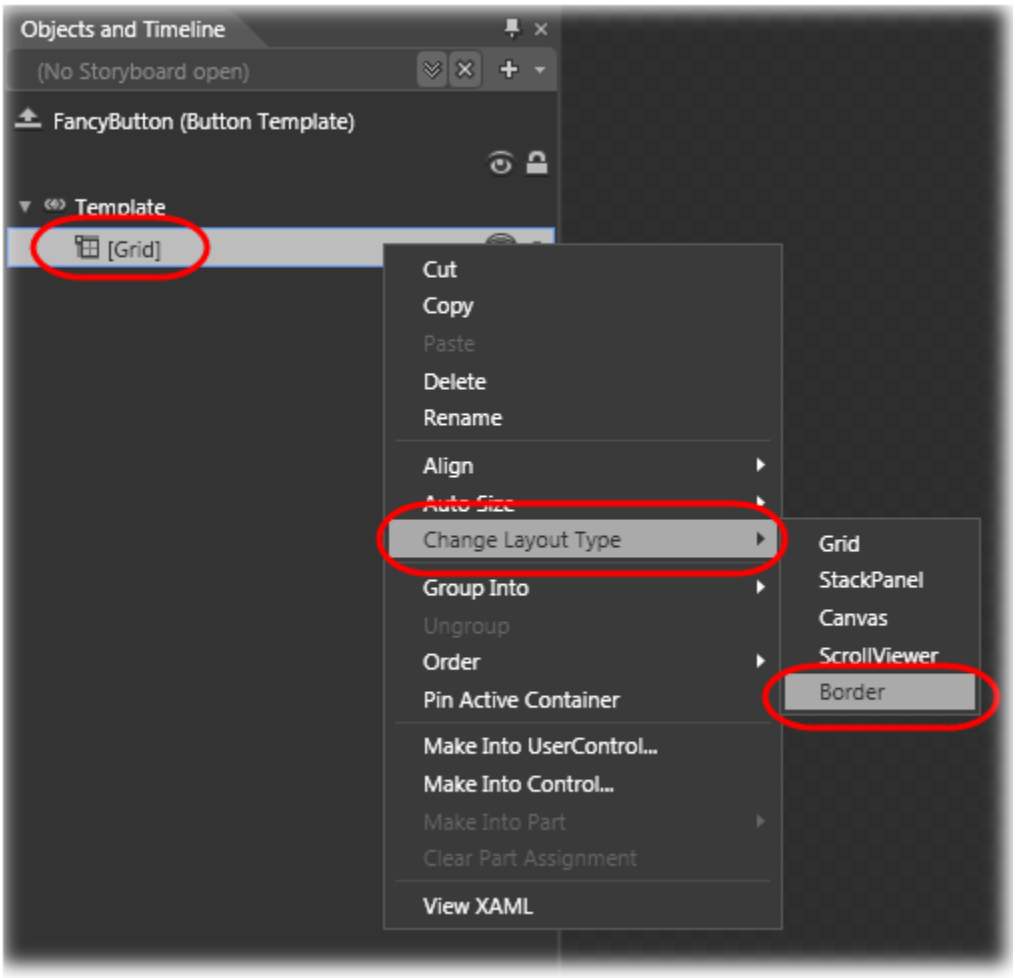

Figure 31

Modification du mode d'affichage du modèle

> **Remarque :** Lorsque vous créez un **ControlTemplate**, vous combinez des objets **FrameworkElement** pour bâtir un contrôle. Un **ControlTemplate** ne doit avoir qu'un seul **FrameworkElement** en tant qu'élément racine**.** L'élément racine lui contient généralement d'autres objets **FrameworkElement**. La combinaison d'objets gère la structure visuelle du contrôle.

9. Désormais, dans la section **Brushes**, sélectionnez la propriété **Background** et choisissez l'option **Solid color brush**. Affectez-lui une couleur orangée, par exemple *#FFFF9A00* et donnez-lui une hauteur de 72 pixels.

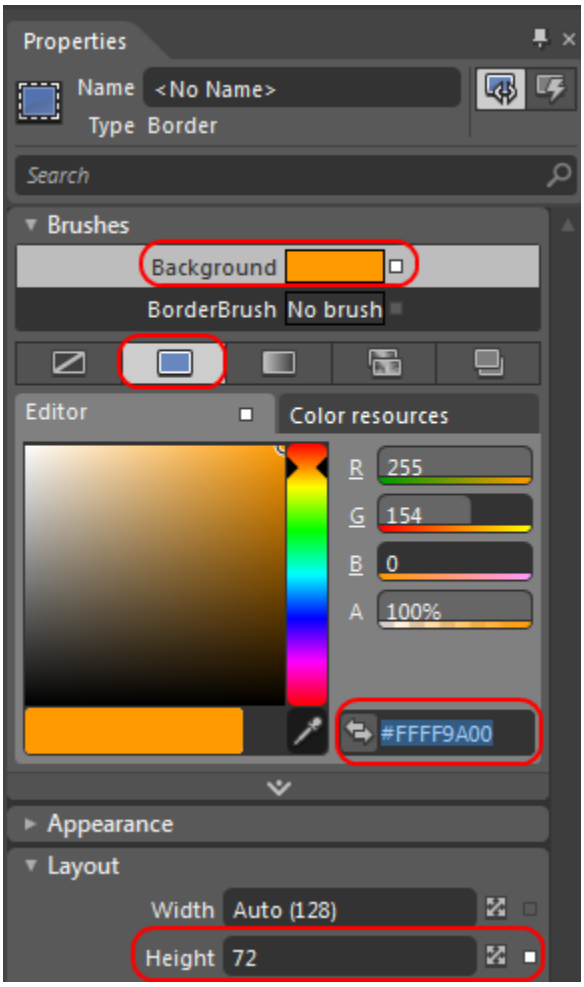

Figure 32

Configuration du motif de l'arrière-plan du bouton

10. Une fois l'apparence du bouton créée, l'étape suivant consiste à lui rajouter un libellé. Tout d'abord, assurez-vous que l'élément **Border** est toujours sélectionné dans le panneau **Objects and Timeline**.

11. Ensuite, passez sur le panneau **Assets**, sélectionnez la catégorie **Controls** puis faites défiler la liste à droite pour localiser le contrôle **TextBlock**. Double-cliquez ensuite sur l'élément de la liste pour insérer une instance de ce contrôle imbriqué au sein de l'élément **Border** du modèle.

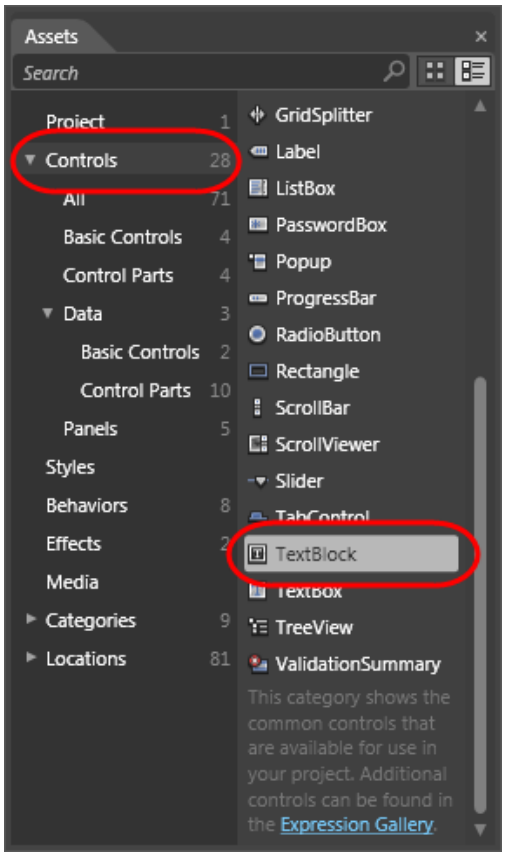

Figure 33

Insertion d'un contrôle TextBlock depuis la boîte à outils

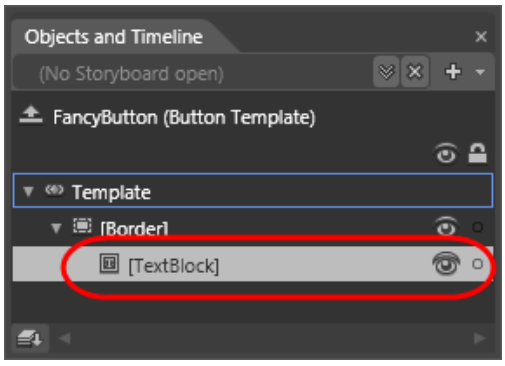

Figure 34

Le panneau Objects and Timeline montrant le nouveau TextBlock imbriqué dans l'élément Border

12. Sélectionnez maintenant  dans le panneau **Tools**, ou bien appuyez sur **B** pour rétablir le mode de sélection.

> **Remarque :** Après insertion du TextBlock, le designer reste en mode insertion, vous permettant donc d'ajouter d'autres éléments TextBlock jusqu'à ce que vous quittiez ce mode.

13. Dans le panneau **Objects and Timeline**, sélectionnez le nouvel élément **TextBlock**. Puis dépliez la catégorie **Brushes** du panneau **Properties** afin d'y fixer le **Foreground** à une couleur claire comme par exemple *#FFFFFFFF.*

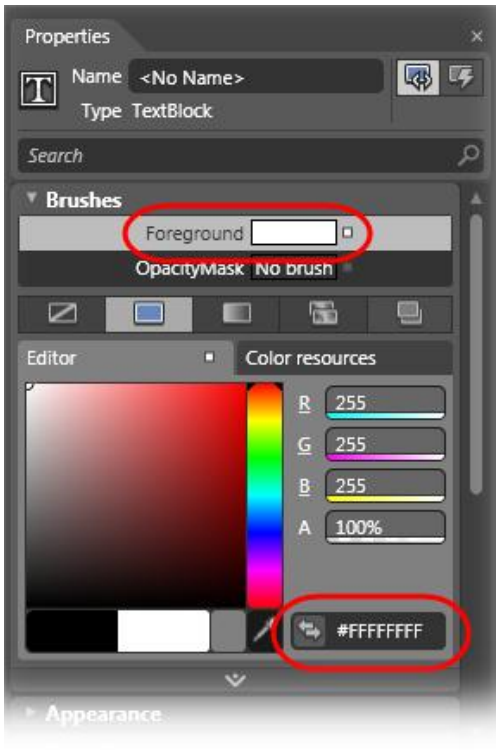

Figure 35
Détermination de la couleur de premier plan du bouton

14. Dépliez ensuite la catégorie **Layout** et fixer à *Center* les propriétés **HorizontalAlignment** et **VerticalAlignment**. Fixez ensuite à *10* la propriété **Margin** pour les bords gauche et droit, et à *4* pour les bords supérieur et inférieur.

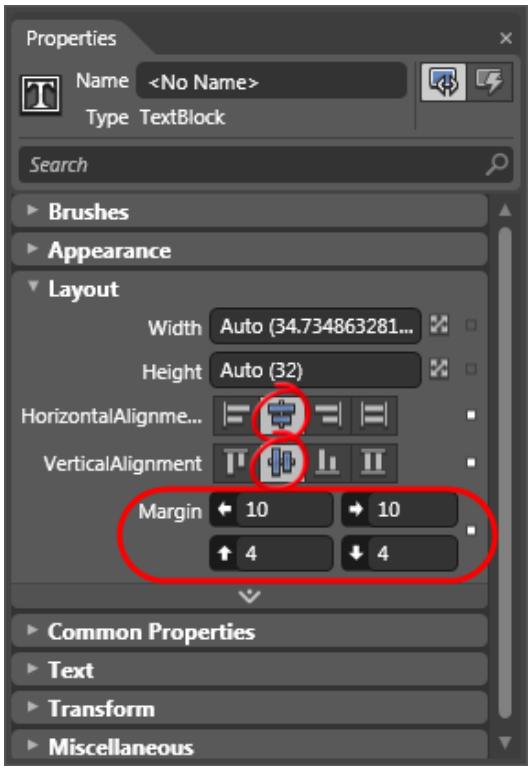

Figure 36

Configuration des propriétés de marge et alignement du libellé

15. Liez la propriété **Text** du contrôle **TextBox** dans le modèle à la propriété Content du contrôle bouton. Pour cela :
    - Dépliez la catégorie **Common Properties**
    - Cliquez sur **Advanced property options** représenté par une icône carrée à côté de la valeur de la propriété
    - Sélectionnez **Template Binding** pour afficher une liste de propriétés dans le contrôle basé sur le modèle qui puisse être liée à cette propriété
    - Choisissez la propriété **Content** de la liste

      Notez que la valeur de la propriété **Text** passé à "*Click Me*", soit la valeur actuelle de la propriété **Content** du bouton, et que la propriété est maintenant mise en évidence avec un trait jaune indiquant qu'elle est liée au modèle

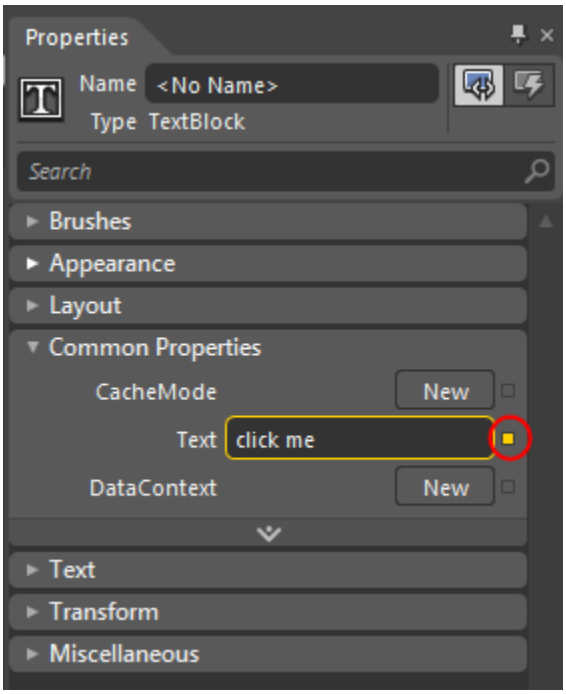

Figure 37

Binding de la propriété Text du modèle

> **Remarque :** Le binding du modèle relie la valeur d'une propriété dans un modèle de contrôle à la valeur d'autres propriétés exposées par ce contrôle, permettant ainsi de désigner un contrôle personnalisé, tout en permettant des modifications lorsque vous incluez le contrôle dans l'application.

16. Appuyez sur **CTRL + S** pour sauvegarder le fichier modifié.
17. Vous êtes désormais prêt à tester le nouveau bouton personnalisé. Appuyez sur**F5** pour compiler et lancer l'application. Saisissez du texte et cliquez sur le bouton.

    Notez que même si le clic engendre le comportement attendu, il n'y a aucune indication visuelle permettent de voir si le bouton a le focus ou s'il est pressé. Dans la tâche suivant, vous modifierez le modèle pour modifier l'apparence du bouton sur cette base.

> **Remarque :** Lorsque vous appuyez sur F5 dans Expression Blend, vous compilez et exécutez l'application Silverlight et l'affichez dans un navigateur pour jauger rapidement des modifications effectuées sur l'interface utilisateur. Ultérieurement, vous testerez l'application dans l'émulateur Windows Phone pour déterminer comment cela se comportera une fois exécutée sur un téléphone.

Exercice 2 – Ajouter des effets visuels à un contrôle

Dans cette tâche, vous modifierez le modèle de contrôle pour y ajouter des effets visuels afin d'avoir un aperçu lorsque le bouton a le focus, et de modifier sa position sur la page lorsqu'il est pressé.

1. Sélectionnez le contrôle **Border** dans le panneau **Objects and Timeline**.
2. Passez maintenant au panneau **States** et regardez les effets visuels disponibles pour ce contrôle.

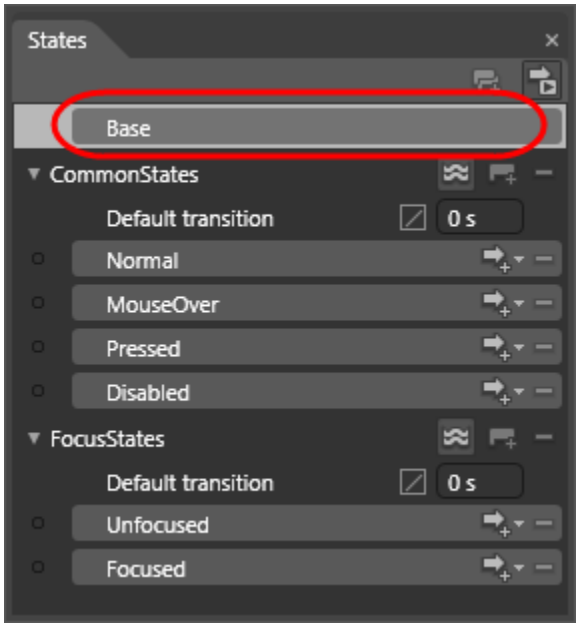

Figure 38

Panneau States indiquant l'état sélectionné

> **Remarque :** Le panneau **States** présente tous les effets visuels du contrôle. **Base** définit les éléments communs à chaque état. Dans cet état, vous personnalisez le modèle pour déterminer l'apparence du contrôle. Puis pour chacun des autres états, vous modifiez le modèle de telle sorte qu'il fournisse un indicateur visuel correspondant à l'état, par exemple une bordure autour du contrôle lorsqu'il a le focus.

3. Dans la catégorie **FocusStates**, sélectionnez l'état **Pressed** dans la catégorie **CommonStates** pour activer l'enregistrement de l'état engendré lorsqu'un clic de souris (pression sur l'écran sur un terminal Windows Phone 7) est effectué.

4. Dans la catégorie **FocusStates**, sélectionnez l'état **Unfocused** pour activer l'enregistrement pour l'état engendré lorsque le bouton n'a pas le focus.
5. Tout changement opéré sur le modèle selon ce mode ne s'applique qu'à cet état particulier.

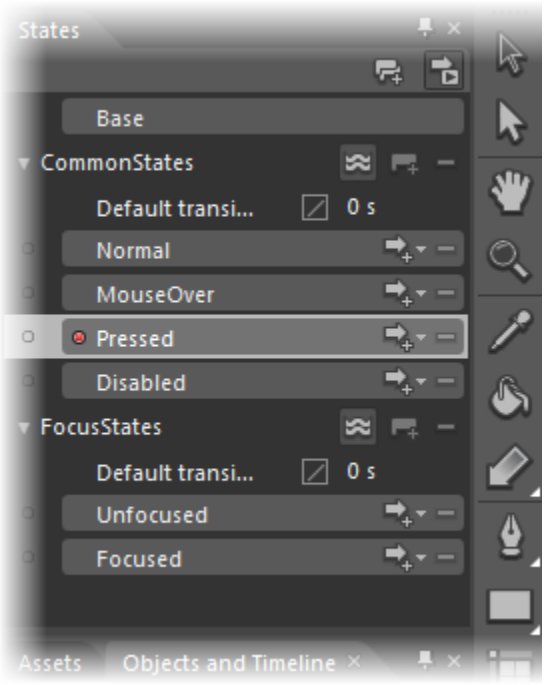

Figure 39
Enregistrement d'état dans Expression Blend

> **Remarque :** Plusieurs changements s'opèrent lorsque vous choisissez un état autre que normal. Tout d'abord, un rond rouge à côté de l'état sélectionné indique qu'il est actif. A côté, un aperçu rouge apparaît autour de la zone de design. Finalement, un libellé dans le coin supérieur gauche de la zone de design indique que l'enregistrement de l'état est activé. Dans ce mode, tout changements effectués sur les éléments du modèle sont enregistrés et stockés dans le document XAML. A l'exécution, ces changements sont appliqués au modèle de base lorsque l'état devient actif.

6. Retournez au panneau **States** et sélectionnez l'état **Pressed** dans la catégorie **CommonStates** pour activer l'enregistrement de l'état engendré lorsqu'un clic de souris (pression sur l'écran sur un terminal Windows Phone 7) est effectué.
7. Dans le panneau **Properties**, dépliez la catégorie **Transform** et sélectionnez la transformation **Translate.** Mettez à 2 les valeurs **X** et **Y**.

Ce changement engendre un léger déplacement de la position du bouton lorsqu'il est cliqué pour indiquer qu'il est pressé.

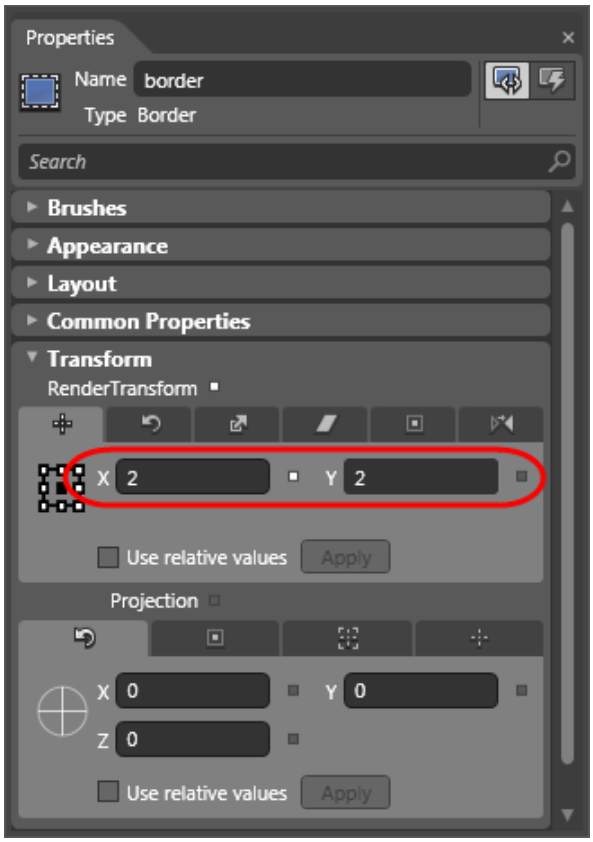

Figure 40

Déplacement de la position du bouton lorsqu'il est pressé

8. Enfin, dans le panneau **States**, sélectionnez le modèle Base pour désactiver le mode d'enregistrement.
9. Appuyez sur CTRL + S pour sauvegarder le fichier mis à jour.
10. Maintenant que vous avez ajouté des effets visuels au contrôle, vous pouvez tester le bouton :
    - Appuyez **F5** pour compiler et lancer l'application
    - Saisissez du texte et appuyez sur **Tab** pour modifier le focus d'entrée du bouton
    - Notez que lorsque le bouton a le focus, une bordure apparaît autour ce celui-ci, indiquant qu'il est actif
    - Pressez maintenant la barre d'espace ou cliquez sur le bouton et notez comment sa position évolue pour indiquer qu'il est enfoncé

Exercice 3 – Créer une animation pour le texte de la bannière

Les animations sont basées sur des images principales définissant les points de début et fin d'une transition visuelle. Pour créer une animation dans Expression Blend, vous créez un Storyboard, et y définissez des images principales sur une *timeline* pour indiquer les changements de propriétés. Par exemple, vous pourriez fixer une image à la marque correspondant à la seconde 0 pour enregistrer la position d'un rectangle sur le bord gauche de l'espace de travail, puis fixer une image à la marque de la seconde 1 pour enregistrer la position de ce même rectangle sur le bord droit de cet espace de travail. L'animation générée serait basée sur la transformation des propriétés X et Y du rectangle en 1 seconde. Lorsque vous exécutez un jeu d'animations, Expression Blend gère les changements de propriétés sur la période de temps spécifiée et affiche les résultats dans l'application. Vous pouvez animer toute propriété d'un objet de cette manière, y compris les propriétés non-visuelles.

Dans cette tâche, vous créerez un Storyboard dans Expression Blend pour animer le texte dans la bannière quand le bouton est pressé.

1. Passez du workspace actuel au workspace Animation. Pour cela, dans le menu **Window**, sélectionnez **Animation** dans le sous-menu **Workspaces**.

   Notez que cela réorganise les fenêtres pour maximiser l'espace d'affichage de la *timeline*.

   > **Remarque :** Pour remettre un workspace à son état par défaut, sélectionnez le menu **Window | Reset Current Workspace**.

2. Si nécessaire, quittez le champ d'édition du modèle de bouton. Pour cela, cliquez sur le bouton **Scope Up**  à côté de l'élément **FancyButton (Button Template)** dans le panneau **Objects and Timeline** pour afficher l'arborescence d'éléments de la page.

   > **Remarque :** Lorsque vous éditez un modèle de bouton, le panneau Objects and Timeline ne montre que l'arborescence d'éléments du modèle. Pour éditer d'autres éléments de la page, vous devez quitter le champ courant.

3. Cliquez désormais sur **New** dans le panneau **Objects and Timeline** pour créer un Storyboard. Il s'agit du bouton marqué avec un '+' situé en haut à droite du panneau.

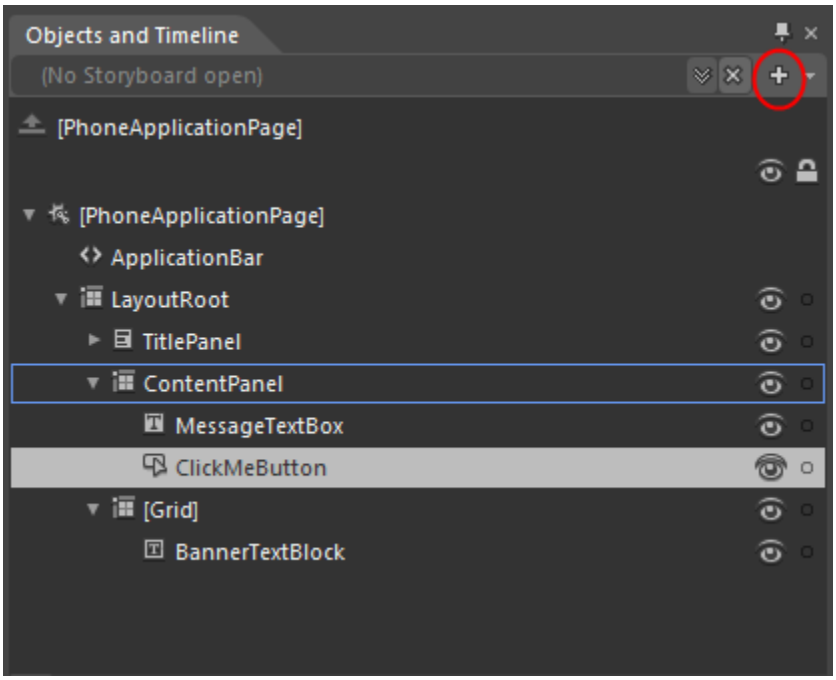

Figure 41

Panneau Objects and timeline en mode Animation

4. Dans la boîte de dialogue **Create Storyboard Resource**, indiquez le nom *AnimateBanner* et cliquez sur **OK**.

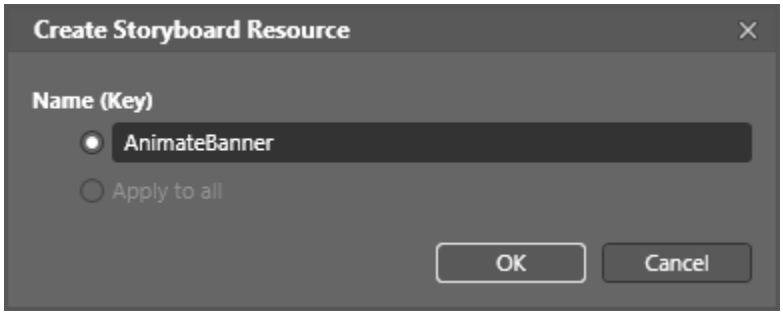

Figure 42

Création d'un nouveau Storyboard pour animer le texte de la bannière

5. Pour créer les images principales de l'animation, cliquez tout d'abord sur l'élément **BannerTextBlock** dans l'arborescence du panneau **Objects and Timeline** pour le sélectionner. Si nécessaire, dépliez son membre parent pour l'afficher.

6. Ensuite, cliquez et glissez la tête de lecture de la position courante sur la timeline à un décalage d'1 seconde.

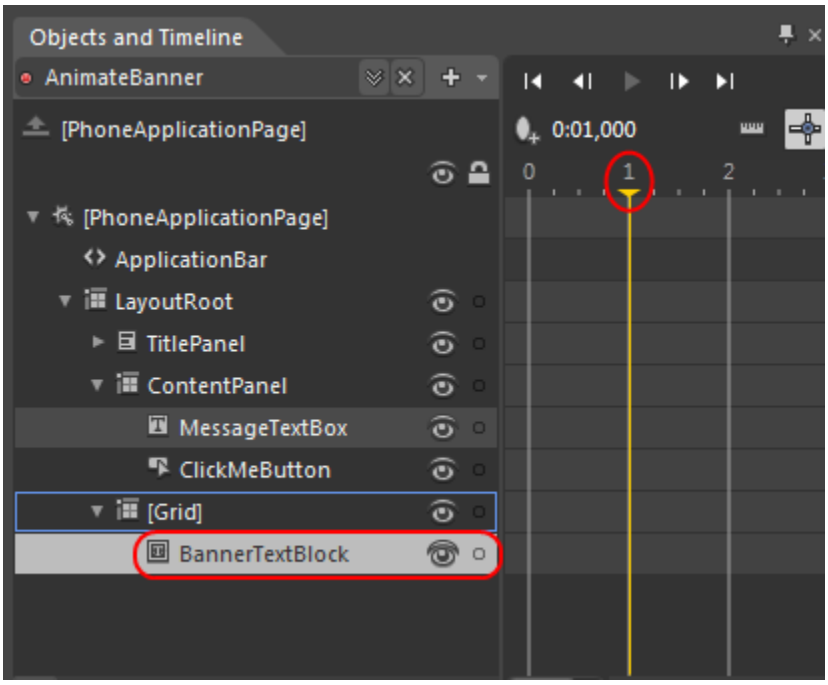

Figure 43

Changement du décalage de temps sur la tête de lecture

> **Remarque :** La tête de lecture correspond au marqueur jaune situé en haut de la *timeline*. Tous changements sur les éléments de la page sont enregistrés comme des images principales à la position de la tête de lecture de la *timeline*.

7. Passez ensuite au panneau **Properties** pour y déplier la catégorie **Transform** et sélectionnez la transformation **Scale**. Fixez à *-1* la valeur de la propriété **X**. Cette transformation

8. Ensuite, passez au panneau **Properties**, dépliez la catégorie **Transform** et sélectionnez la transformation de type **Scale** (échelle). Définissez la valeur de la propriété **X** à *-1*. Cette transformation reflète l'élément le long de son axe horizontal.

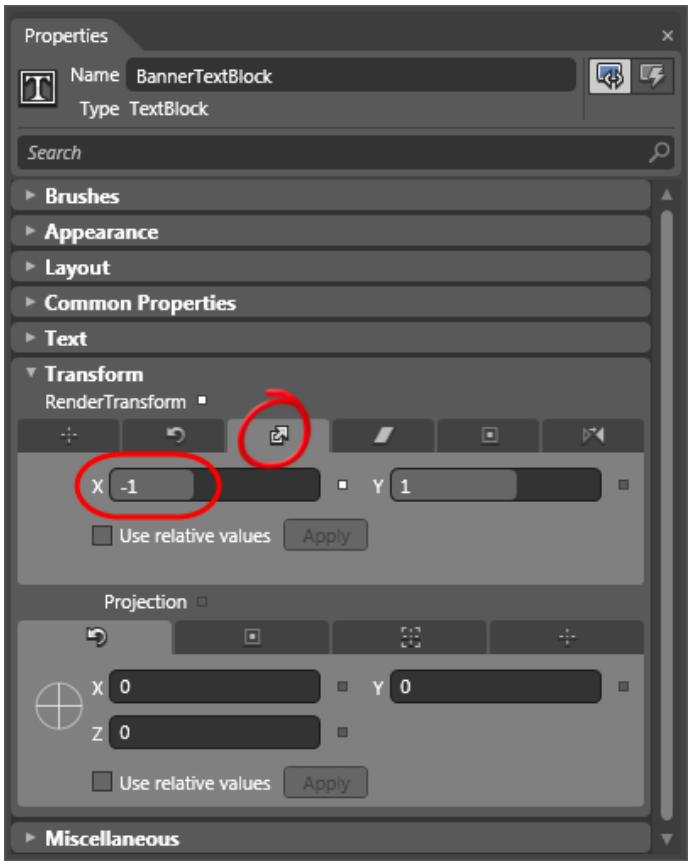

Figure 44

Application d'une transformation d'échelle

9. Retournez au panneau de la *timeline*.

   Notez qu'il contient une nouvelle image, identifiée par un cercle gris, au décalage de temps défini qui est le résultat des changements d'éléments dans l'arborescence alors que la *timeline* est active.

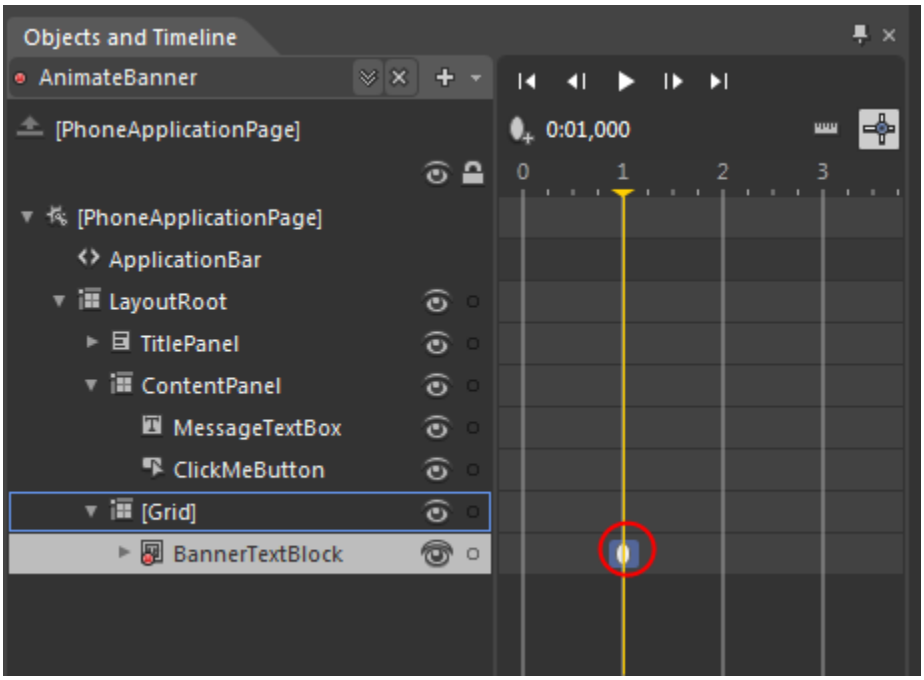

Figure 45

Création d'une nouvelle image dans le Storyboard d'animation

10. Décalez désormais la tête de lecture 2 secondes
11. Retournez une fois encore au panneau **Properties** pour y déplier la catégorie **Transform** et y sélectionner la trasnformation **Scale**. Remettez la propriété **X** à *1*pour remettre l'élément dans ton état initial.

    Notez qu'une seconde image est apparue dans la *timeline* du storyboard suite à ce changement.

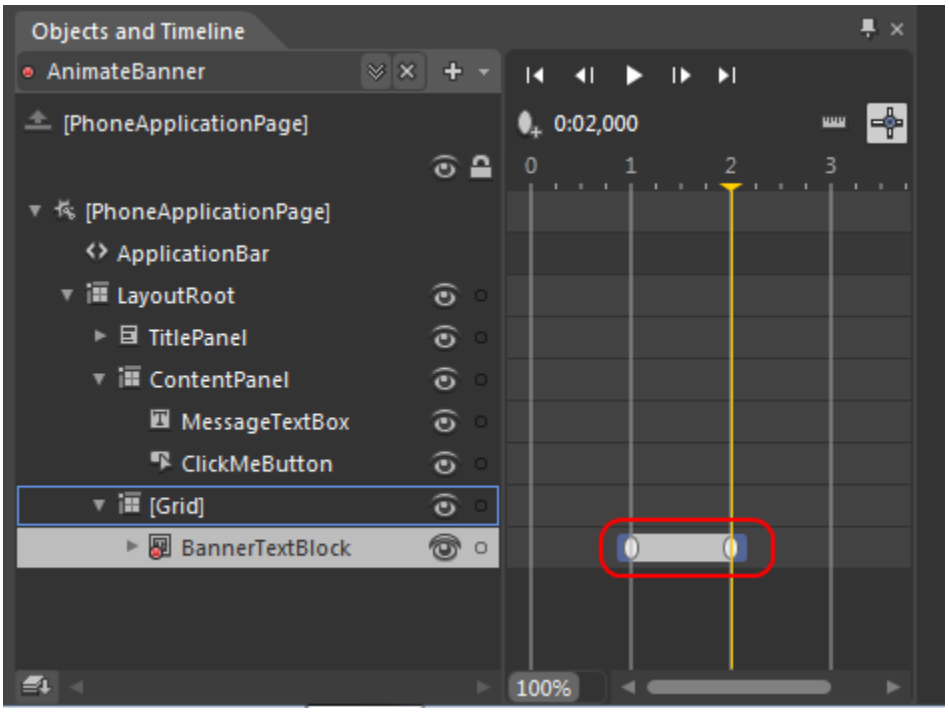

Figure 46

Panneau Objects and Timeline montrant les images insérées

12. Pour tester l'animation dans le designer, vous devez ajouter du texte dans la bannière. Cliquez tout d'abord sur le cercle rouge à gauche du nom de l'animation *AnimateBanner* en haut à gauche du panneau pour désactiver temporairement l'enregistrement, ne voulant pas intégrer le texte ajouté à l'animation.

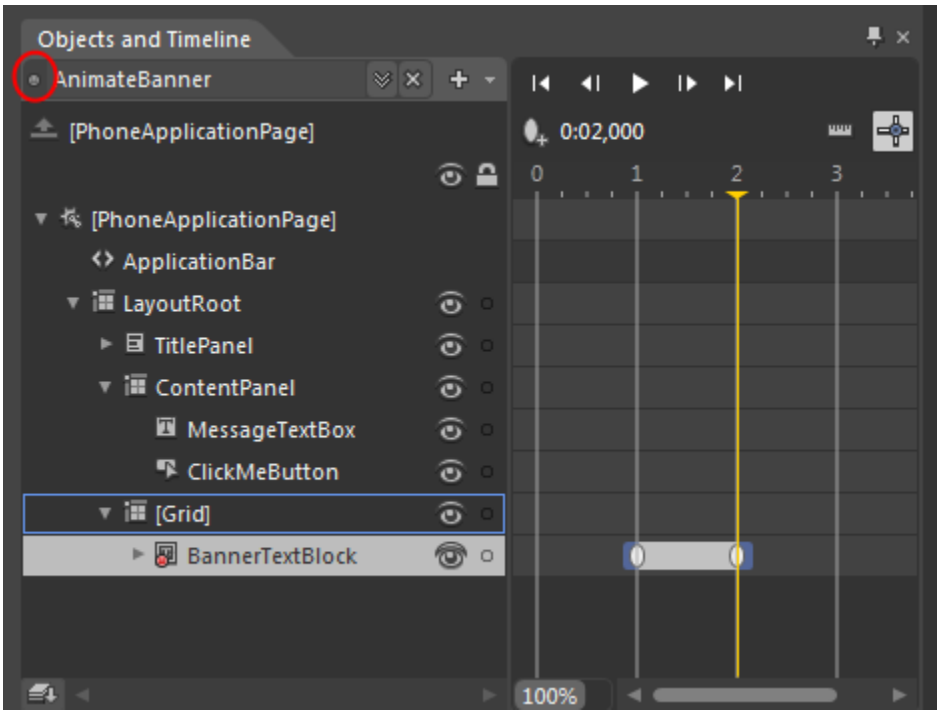

Figure 47

Désactivation du mode recording

> **Remarque :** Lorsque vous désactivez l'enregistrement, toutes les modifications faites sur les éléments de la page ne sont pas intégrées à l'animation.

13. Effectuez maintenant un clic-droit sur l'élément **BannerTextBlock** sur la surface de design et sélectionnez **Edit Text**. Saisissez un texte pertinent dans la bannière et appuyez sur **ENTER**.
14. Pour tester l'animation, appuyer sur le bouton **Play** au-dessus de la *timeline*.

    Notez comment le texte de la bannière pivote horizontalement autour de son axe médian et l'uniformité du mouvement lors de l'animation. Dans les étapes suivantes, vous appliquerez une fonction simple pour modifier l'interpolation entre les images principales et produire une animation plus naturelle.

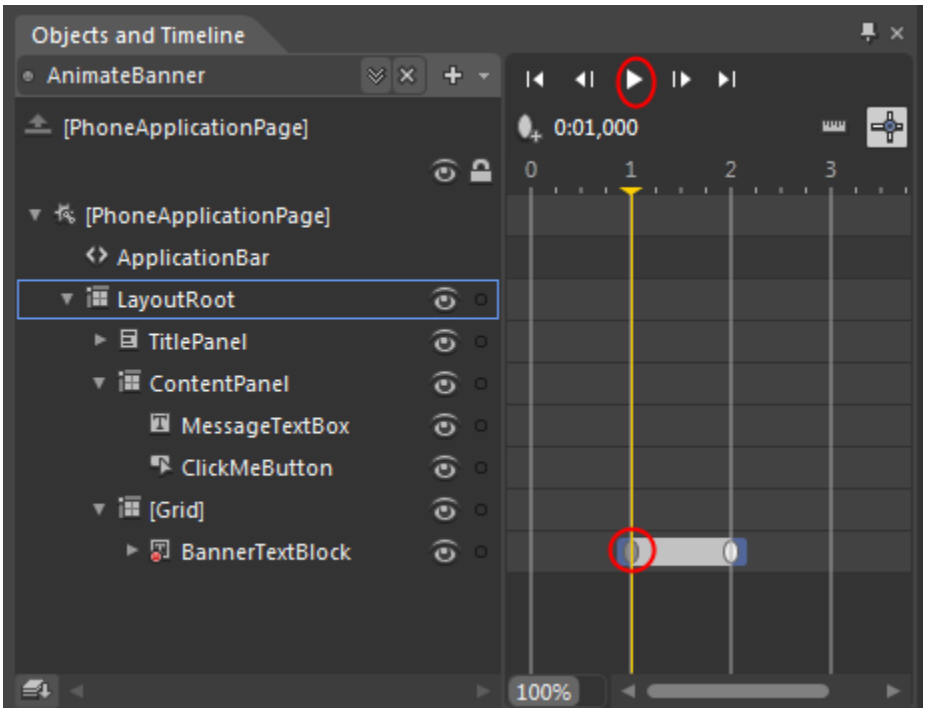

Figure 48

Test de l'animation dans le designer

15. Sélectionnez la première image principale en cliquant à l'intérieur de l'icône représentant un cercle gris dans la *timeline*.

    Notez que lorsque vous sélectionnez une image, la vue designer se met à jour pour afficher l'état des éléments de l'interface utilisateur tels qu'ils apparaissent dans cette image ; dans ce cas, le texte de la bannière apparait réfléchi.

    Assurez-vous maintenant que dans la catégorie **Easing** du panneau **Properties**, **EasingFunction** est sélectionné ; dépliez la liste déroulante pour afficher une liste de fonctions disponibles puis choisissez la fonction **Cubic Out**.

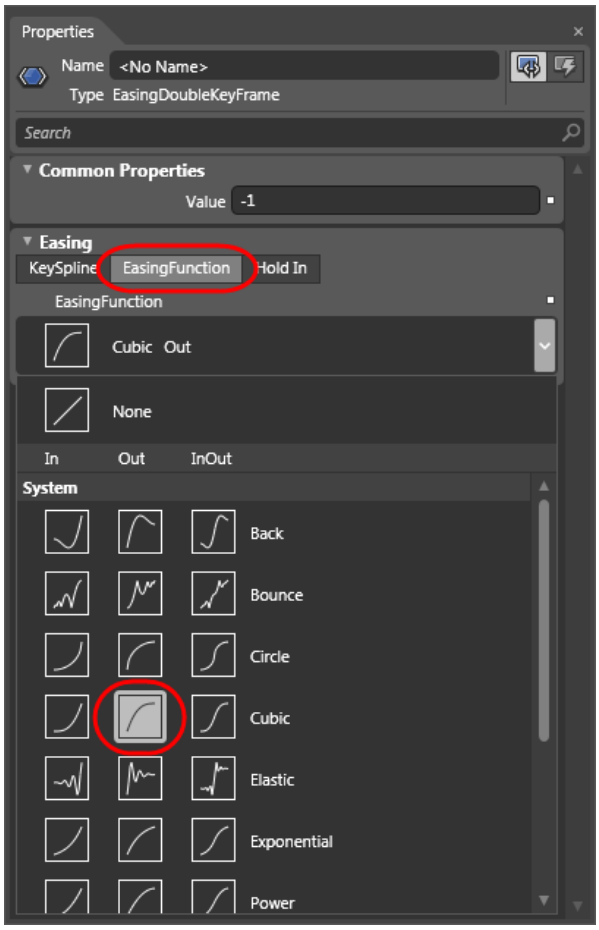

Figure 49

Application d'une fonction d'interpolation entre des images

> **Remarque :** L'interpolation d'image décrit comment les changements de propriétés sont animés durant le laps de temps qui s'écoule entre 2 images principales. Vous pouvez modifier les valeurs d'interpolation en appliquant une fonction à la transition pour obtenir une animation plus réaliste.

16. Répétez la procédure de l'étape précédente pour configurer la fonction pour la seconde image. Choisissez cette fois la fonction **Cubic InOut** pour faire démarrer l'animation à un rythme lent, accélérant progressivement, et finalement ralentissant à l'approche de l'image de fin.
17. Pour tester l'effet de la fonction de l'animation, appuyez sur **Play** au-dessus de la *timeline*.

Notez que la bannière démarre sa rotation rapidement, puis ralentit à l'apparition du texte, puis ré-accélère avant de finalement ralentir à nouveau pour retourner à sa position initiale. Sauvegardez tous les fichiers qui ont été modifiés sous Blend.

18. Vous avez ici créé un ensemble d'animations pour le texte de la bannière, mais n'est à date qu'une ressource inutilisée du projet. Pour jouer l'animation, vous devez associer le storyboard à un événement, dans notre cas, à chaque fois que le bouton *Click Me* est pressé. Retournez alors sous Visual Studio pour ouvrir le gestionnaire d'événement du **Click** pour ce bouton :
    - Affichez la vue **Design** du fichier **MainPage.xaml**
    - Double-cliquez sur le bouton sur la surface de design pour ouvrir le fichier du code sous-jacent
    - Placez le curseur sur le gestionnaire d'événement
19. Pour modifier le gestionnaire d'événement pour jouer l'animation, insérer l'extrait de code suivant dans l'espace surligné immédiation avant l'accolade de fin.

C#

```
private void ClickMeButton_Click(object sender, RoutedEventArgs e)
{
  BannerTextBlock.Text = MessageTextBox.Text;
  MessageTextBox.Text = String.Empty;
  AnimateBanner.Begin();
}
```

Exercice 4 – Vérification

Lors du design dans Expression Blend, vous avez pré visualisé l'interface utilisateur de votre application Silverlight et exécutée dans un navigateur. Dans cette tâche, vous exécutez l'application dans l'émulateur Windows Phone pour voir le résultat de vos modifications dans l'interface utilisateur.

1. Dans Visual Studio, appuyez sur **F5** pour compiler et déployer l'application dans l'émulateur Windows Phone. Attendez que l'application se lance et affiche sa page principale.
2. Saisissez du texte dans la zone de texte
3. Appuyez sur **Tab** une fois et notez la bordure affichée autour du bouton lorsqu'il reçoit le focus
4. Cliquez sur le bouton pour fixer le texte de la bannière et notez comment sa position évolue pour indiquer qu'il est pressé
5. Regardez l'animation du texte de la bannière se faire lorsque vous cliquez sur le bouton.

Figure 50

Exécuter l'application dans l'émulateur Windows Phone

6. Une fois le test terminé, cliquez sur l'icône **Close** dans le coin supérieur droit de l'émulateur pour l'éteindre et mettre ainsi fin à la session de débogage.

> **Remarque :** Il est recommandé d'éteindre l'émulateur uniquement si vous ne comptez pas ré-exécuter l'application ; sinon, laisser le ouvert afin de redéployer et déboguer l'application à chaque nouvelle mise à jour à tester.

Résumé

Cet atelier vous a présenté comment développer des applications Windows Phone avec Silverlight. En le réalisant, vous vous êtes familiarisés avec les outils nécessaires à la création et aux tests d'application Windows Phone.

Vous avez créé une nouvelle application Silverlight pour Windows Phone avec Microsoft Visual Studio 2010 Express pour Windows Phone, puis utilisé cet outil gratuit pour définir la logique applicative et l'interface utilisateur. Vous avez appris à déployer une application dans l'émulateur Windows Phone et comment utiliser le débogueur pour positionner des points d'arrêt, faire du pas-à-pas à travers le programme et analyser les valeurs de variables. Enfin, vous avez vu comment ouvrir un projet dans Expression Blend pour modifier le style des contrôles et créer des animations.