

# Workshop Silverlight Réaliser un lecteur vidéo en 20 étapes

Edgar Maucourant (<http://blog.nftinside.com>)



# Sommaire

<b>Introduction</b> -----	<b>p3</b>
<b>Première partie : Création du corps du lecteur vidéo</b> -----	<b>p4</b>
Etape 1 : Ouverture de Microsoft Expression Blend et création du projet-----	p4
Etape 2 : Redimensionner la zone de travail-----	p6
Etape 3 : Elargir la zone de travail-----	p7
Etape 4 : Créer le fond du lecteur vidéo-----	p7
Etape 5 : Créer le fond de la vidéo-----	p10
Etape 6 : Ajouter la vidéo-----	p11
Etape 7 : Test du lecteur-----	p13
<b>Deuxième partie : Ajoutons un peu d'interactivité !</b> -----	<b>p15</b>
Etape 8 : Ajout d'un Canvas bouton-----	p15
Etape 9 : Ajout d'un fond pour notre bouton-----	p16
Etape 10 : Ajout d'un canvas 'Play'-----	p17
Etape 11 : Blocage de la lecture et ajout du gestionnaire d'évènement-----	p19
Etape 12 : Ajouter la fonction JavaScript « onPlay »-----	p20
Etape 13 : Test du lecteur-----	p22
<b>Troisième partie : Ajoutons un peu d'animation !</b> -----	<b>p23</b>
Etape 14 : Création d'un bouton Pause-----	p23
Etape 15 : Ajouter un comportement au bouton 'Pause'-----	p25
Etape 16 : Ajouter l'animation de 'Play' à 'Pause'-----	p25
Etape 17 : Animation de Pause à Play-----	p29
Etape 18 : Déclenchement des animations-----	p29
Etape 19 (optionnelle) : Ajouter un effet d'illumination-----	p31
Etape 20 : Tester le lecteur vidéo -----	p34
<b>Quatrième partie : Pour aller plus loin...</b> -----	<b>p34</b>
<b>Conclusion</b> -----	<b>p34</b>

## Introduction

Bienvenu(e) dans ce Workshop d'une heure consacré à Silverlight, et plus spécifiquement à l'utilisation de Microsoft Expression Blend et Visual Studio 2008 pour créer un Lecteur Vidéo en Silverlight 1.0.

Nombre d'entre vous ont sûrement déjà entendu parler de Silverlight mais il est bon de rappeler quelques points essentiels de cette merveilleuse technologie. Silverlight est la nouvelle pierre angulaire de Microsoft pour la création d'applications Web riches et interactives coté client. Même si Silverlight pourrait être comparé à son grand-frère WPF, les deux technologies sont très différentes dans leurs implémentations et dans leurs stratégies.

Silverlight et WPF partagent des comportements communs (Silverlight peut être vu comme un sous-ensemble de WPF) mais reposent sur deux moteurs bien différents. Alors que WPF tire parti du Framework .Net 3.0 (*et maintenant 3.5*), Silverlight dispose de son propre moteur embarqué dans un plug-in très léger disponible gratuitement. Bien que WPF soit bien plus puissant que Silverlight cette nécessité de présence du .Net Framework le limite à quelques plateformes seulement. Au contraire Silverlight puisqu'il est destiné essentiellement au Web a dès le départ été pensé pour être disponible sur plusieurs plateformes.

Silverlight, repose sur l'utilisation d'un plug-in permettant d'interpréter un nouveau langage : XAML. Ce langage reposant sur XML est apparu au départ avec WPF. Silverlight (*anciennement connu sous le nom WPF/E pour Everywhere*) comprend un sous-ensemble de ce langage. Une des forces de ce langage est d'être entièrement compréhensible par les moteurs de recherche et donc de permettre aux applications Silverlight de pouvoir être très bien indexées.

Pour manipuler ce code coté client Microsoft a choisi pour cette mouture de reposer sur un langage largement connu des développeurs Web : JavaScript. En ce basant sur un langage éprouvé et facile d'accès, Silverlight est promu à un bel avenir concernant sa percée auprès des développeurs Web.

**Note :** Silverlight 2.0 (la prochaine version) sera capable de travailler avec C# et VB.Net

Annoncé comme la contre-attaque de Microsoft envers Adobe Flash, Silverlight pourrait bien rapidement se tailler la part du lion sur le Web !

Après ce préambule, passons maintenant aux choses sérieuses et commençons notre Projet !

## Première partie : Création du corps du lecteur vidéo

Lors cette première partie vous verrez comment créer le corps du lecteur vidéo ainsi que la manière

**Note :** La version utilisée est une pré-version de Blend 2 intitulée : Microsoft Expression Blend 2 December Preview. Comme son nom l'indique cette version n'est pas finalisée et est en anglais, seule cette version supporte la création de projet Silverlight, c'est pourquoi nous avons du l'utiliser au détriment d'une version finalisée en français.

A sa sortie Blend 2 supportera les projets Silverlight 1.0 (comme celui que nous allons créer) et Silverlight 2.0 (sortie prévue pour Mars 2008). A l'heure de l'écriture de ce Workshop la date de sortie de la version finale de Microsoft Expression Blend 2 n'est pas encore connue.

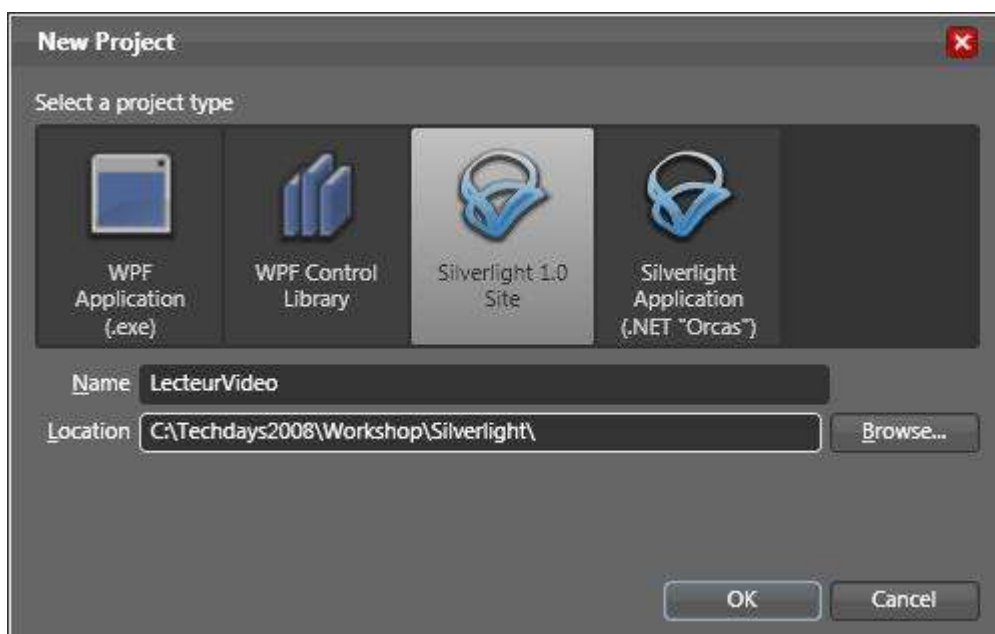
d'inclure une vidéo à l'intérieur de votre projet Silverlight. Vous vous familiariserez aussi avec Microsoft Expression Blend 2, l'outil de prédilection lorsqu'il s'agit de créer des projets Silverlight.

### Etape 1 : Ouverture de Microsoft Expression Blend et création du projet

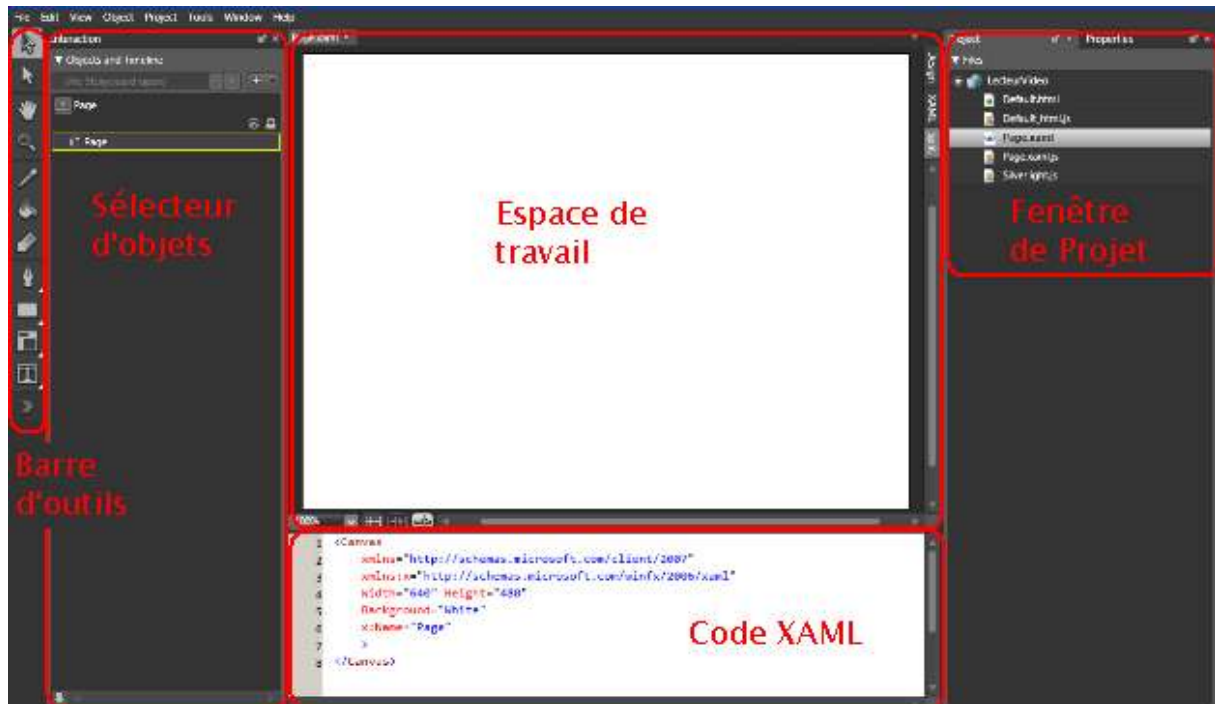
Ouvrez Microsoft Expression Blend 2 en double cliquant sur l'icône présent sur le bureau de la machine. Puis créez un nouveau projet en utilisant l'une des deux méthodes ci-dessous :

- Le menu « File » puis « New Project »
- Le raccourci clavier : Ctrl + Shift + N

Une fenêtre semblable à celle présentée ci-dessous va s'ouvrir. Choisissez un projet de type '**Silverlight 1.0 Site**' et saisissez les mêmes informations que celle inscrites dans l'image :



Blend va alors créer un nouveau projet pour vous et ouvrir l'interface de travail du projet présentée ci-dessous :



Prenons le temps de détailler rapidement cette interface. Tout d'abord au centre se trouve l'espace de travail et la fenêtre de code XAML associée. Tout objet ainsi que ces attributs (par exemple la couleur, la taille, certains comportements, etc...) sont définis par du code XAML (prononcez « Zammel »). L'espace de travail est synchronisé avec la fenêtre de code XAML, si bien que le moindre changement dans l'une ou l'autre des fenêtres provoquera la mise à jour de l'autre partie. Ainsi si je modifie la taille d'un objet en écrivant sa valeur dans le code XAML l'objet sera automatiquement redimensionné sur l'espace de travail. De la même façon si je déplace un objet sur l'espace de travail en utilisant l'outil de sélection (la flèche noire en haut de la barre d'outils), le code XAML définissant sa position sera mis à jour en conséquence.

**Note :** Par défaut vous êtes en visualisation partagée (Split en anglais) c'est-à-dire que vous voyez en même temps l'espace de travail et la fenêtre de code XAML. Vous pouvez si vous le souhaitez passer en « Espace de travail seul » (option « Design » en anglais), Code XAML seul (option « XAML ») ou espace partagé en utilisant le menu en haut à droite de l'espace de travail.

A gauche se trouve la barre d'outils permettant de sélectionner les outils que nous utiliserons pour sélectionner, dessiner, modifier etc... les objets sur l'espace de travail.

Entre cette barre d'outils et l'espace de travail se situe le sélecteur d'objets et de scénarios qui nous permet de voir la hiérarchie des objets sur la scène et de facilement les sélectionner.

A droite se trouve la fenêtre de projet et la fenêtre de propriétés (vous pouvez passer de l'un à l'autre en cliquant sur '**Project**' ou '**Properties**' en haut de la fenêtre). La fenêtre projet permet de

visualiser tout les fichiers du projet, tandis que la fenêtre de propriétés permet de modifier les propriétés de l'élément sélectionné.

## Etape 2 : Redimensionner la zone de travail

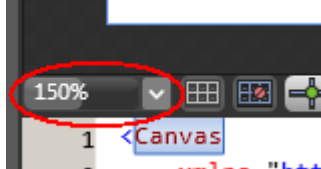
Tout projet Silverlight comporte un Canvas (zone de travail) de base. Il est l'objet depuis lequel tous les autres objets seront positionnés. Il représente en quelque sorte la surface d'affichage de votre animation. Par défaut Blend crée un Canvas racine appelé « Page » et possédant une taille de 640pixels de large sur 480pixels de haut. Dans notre cas cette taille est trop importante (le lecteur vidéo serait trop gros sur la page) et nous allons la réduire à 400pixels de large pour 300pixels de haut.

<p>Pour redimensionner le Canvas, sélectionnez le en cliquant sur la ligne « Page » dans le sélecteur d'objets à gauche de l'espace de travail.</p>	
<p>Puis dans la fenêtre de projet à droite de l'espace de travail cliquez sur « Properties » ce qui affichera les propriétés du Canvas.</p> <p>Assurez-vous que le champ Name contienne bien la valeur « Page » et que le type soit « Canvas » comme sur l'image ci-dessus.</p>	
<p>Ensuite descendez dans la fenêtre « Properties » jusqu'à trouver, dans la catégorie '<b>Layout</b>' (Disposition en anglais), les propriétés '<b>Width</b>' (Largeur en anglais) et '<b>Height</b>' (Hauteur en anglais). Saisissez les valeurs suivantes et validez à chaque fois en tapant sur la touche 'entrée'</p>	

Votre Canvas devrait se redimensionner à l'écran et devenir plus petit.

### Etape 3 : Elargir la zone de travail


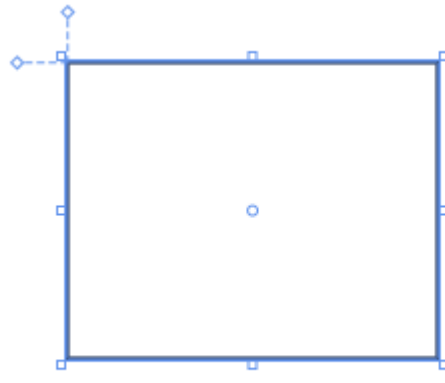
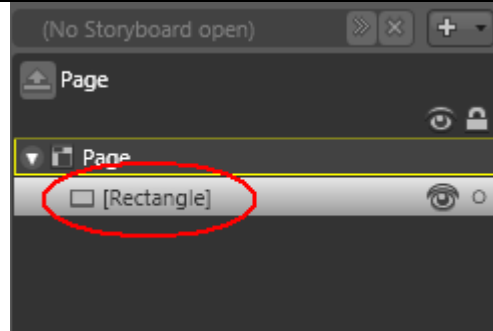
Lorsque l'on travaille sur un projet le mieux est de toujours disposer de la plus grande vue possible dans la zone de travail. Dans le cas présent notre Canvas possède une taille intéressante pour Internet mais apparaît un peu petit dans Blend pour travailler facilement. Nous allons donc zoomer la zone de travail de 150%.

Vous pouvez utiliser la fenêtre de Zoom située en bas à gauche de l'espace de travail.	
--	--

Vous pouvez à présent utiliser les ascenseurs en bas et à droite de l'espace de travail pour centrer votre Canvas à l'écran.

### Etape 4 : Créer le fond du lecteur vidéo

Afin d'avoir un design plus attrayant nous allons ajouter un fond à notre lecteur vidéo.

Sélectionnez l'outil rectangle dans la barre d'outils	
Dessinez un rectangle quelconque sur le Canvas (nous changerons ses propriétés de taille et de placement juste après)	
Notez qu'un nouvel objet sans nom de type rectangle ( <b>[Rectangle]</b> ) a été ajouté à la scène et est disponible dans le sélecteur d'objet. Cet objet est un enfant du Canvas racine comme le symbolise le retrait dans l'affichage.	

**Note :** Tous les objets dans Silverlight (à part le Canvas racine) ont un père, c'est-à-dire un objet qui les contient. Ils peuvent eux-mêmes avoir des enfants, c'est-à-dire contenir d'autres objets. Cette hiérarchie permet de traiter les objets par groupe. Ainsi si un objet rectangle (enfant) est contenu dans un Canvas (Père) et que ce Canvas devient invisible l'objet enfant Rectangle sera lui aussi invisible. Le contraire n'est pas vrai : rendre le rectangle invisible ne modifie pas le parent. Les objets peuvent ainsi composer des hiérarchies complexes (un enfant pouvant lui-même avoir des enfants...). Cette hiérarchie est très importante aussi pour les événements comme nous le verrons plus tard.

A présent que nous avons dessiné notre rectangle nous allons modifier ses propriétés :

En utilisant la fenêtre de propriété, cliquez sur **'Fill'** (« Remplissage » en anglais) ce qui va nous permettre de définir la couleur d'arrière plan de notre rectangle.

Nous souhaitons un gris ni trop clair ni trop foncé dont la composante ARGB (Alpha, Red, Green, Blue) sera : « #FF707070 ».

(Cf : Note ci-dessous pour une explication de cette notation.)

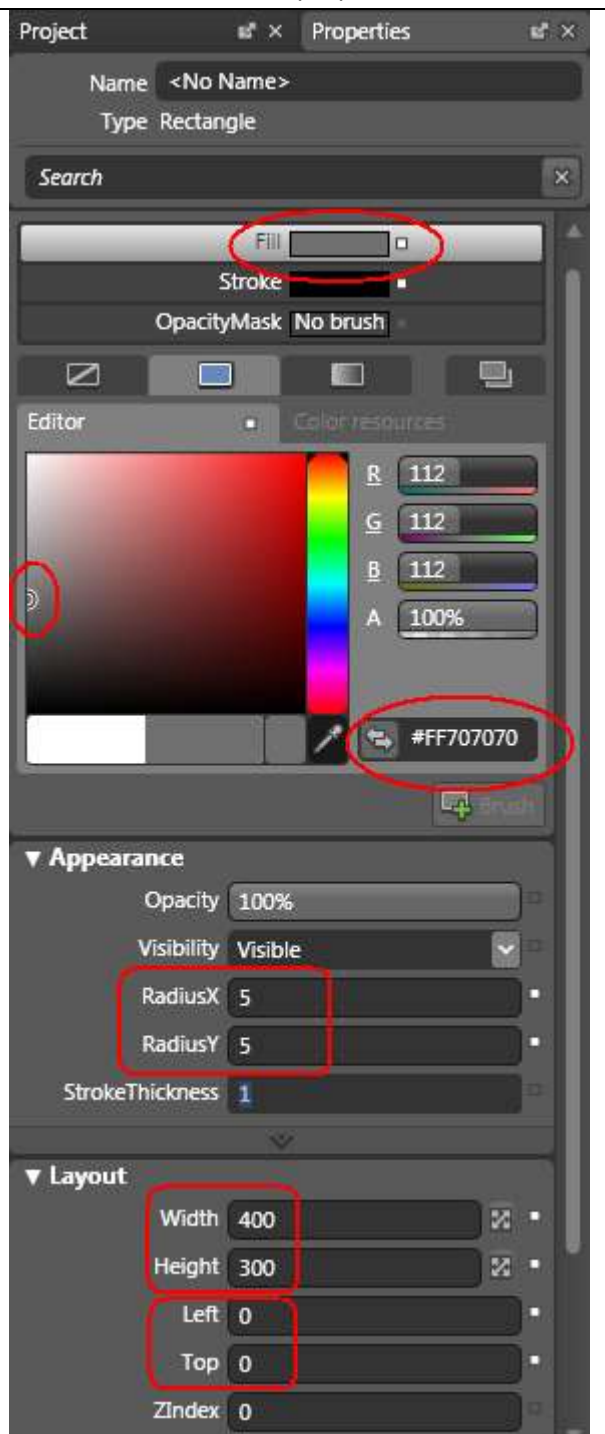
Nous pouvons donc utiliser le sélecteur de couleur (le carré avec des dégradés de couleurs), en glissant le curseur jusqu'à obtenir la bonne teinte ou saisir directement cette valeur dans le champ hexadécimal (en bas à droite du sélecteur de couleurs)

Nous souhaitons aussi avoir des coins arrondis sur ce rectangle, nous allons pour cela utiliser les propriétés : RadiusX et RadiusY. Une valeur de 5 pour chacune d'elle sera parfait.

Enfin nous souhaitons que ce rectangle ait la même taille que le Canvas de base, nous utiliserons donc les propriétés **'Width'** et **'Height'** pour lui donner les bonnes dimensions.

Notez que nous plaçons aussi les propriétés **'Left'** (position gauche) et **'Top'** (position haut) toutes les deux à 0.

(Cf : Deuxième note ci-dessous pour une explication de ces propriétés)



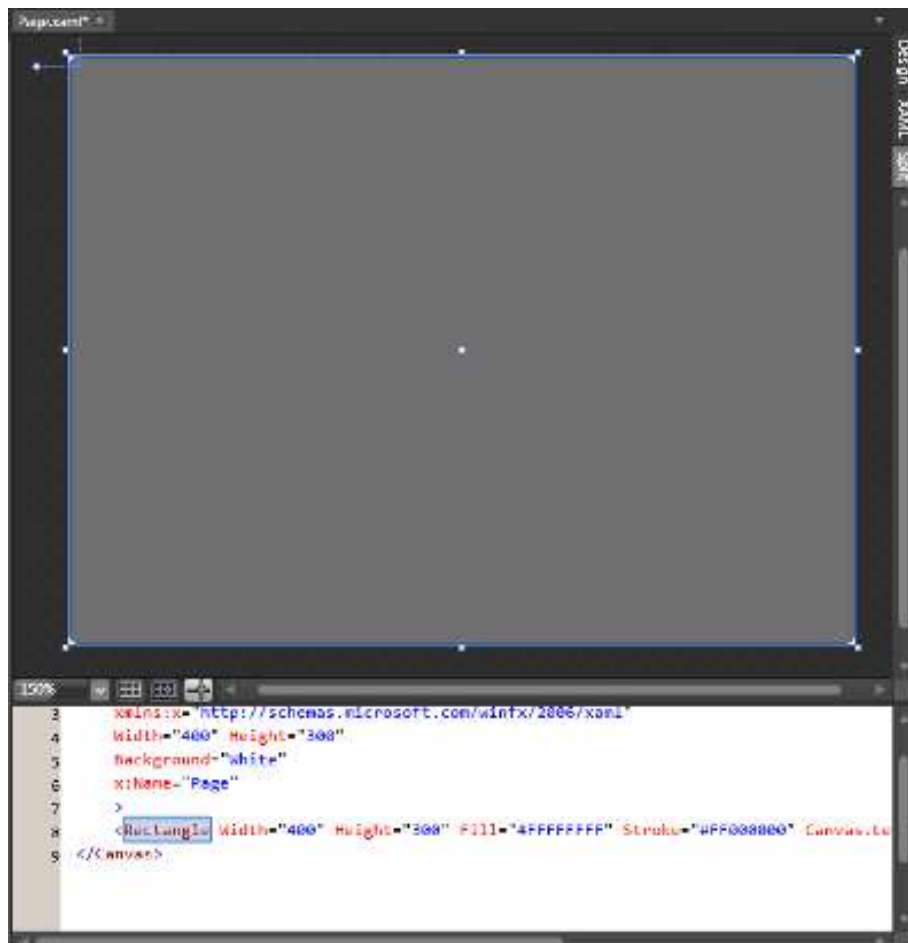
**Note** : en infographie une couleur est souvent définie par sa quantité de rouge, de vert (jaune en peinture) et de bleu. Les valeurs vont de 0 (rien) à 255 (maximum de cette couleur). Certains logiciels ajoutent une composante Alpha définissant la transparence de la couleur (0 à 100%)

En technologie Web ces composantes sont souvent notées de manière hexadécimale c'est-à-dire sur 16 valeurs (0,1,2,3,4,5,6,7, 8,9,A,B,C,D,E,F). On peut ainsi représenter les 256 valeurs sur 2 caractères de 00 (0 en décimal) à FF (255 en décimal) au lieu de 3.

**Note** : le placement des objets en Silverlight se fait toujours en utilisant les propriétés '**Top**' et '**Left**'. Le coin en haut à gauche ayant pour valeur (0,0), et le coin en bas à droite (Largeur / Hauteur). Les valeurs croissent donc en allant vers la droite et vers le bas. Ainsi si je veux déplacer mon objet vers la droite j'augmente sa valeur '**Left**', de même avec sa propriété '**Top**' pour le déplacer vers le bas.

Attention ces propriétés sont relatives au parent de l'objet et non au Canvas racine (sauf si celui-ci est le parent). Ainsi si un objet Rectangle enfant d'un objet Canvas lui-même enfant du Canvas racine utilise les propriétés '**Top**' et '**Left**', sa position (du Rectangle) sera calculée par rapport à la position de l'objet Canvas enfant et non par rapport au Canvas Racine.

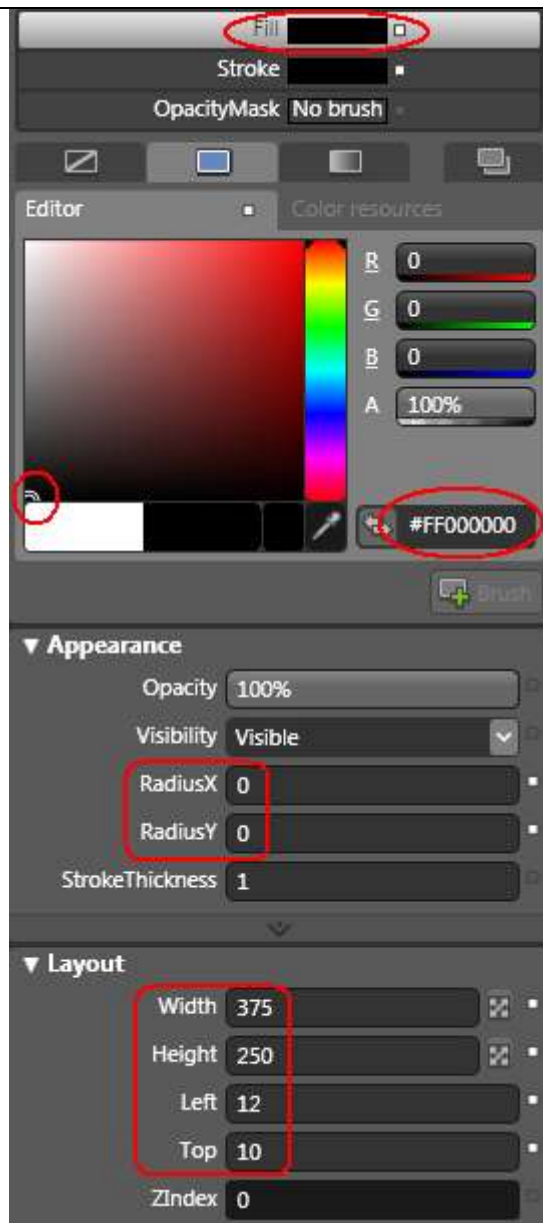
Le résultat attendu est celui là :



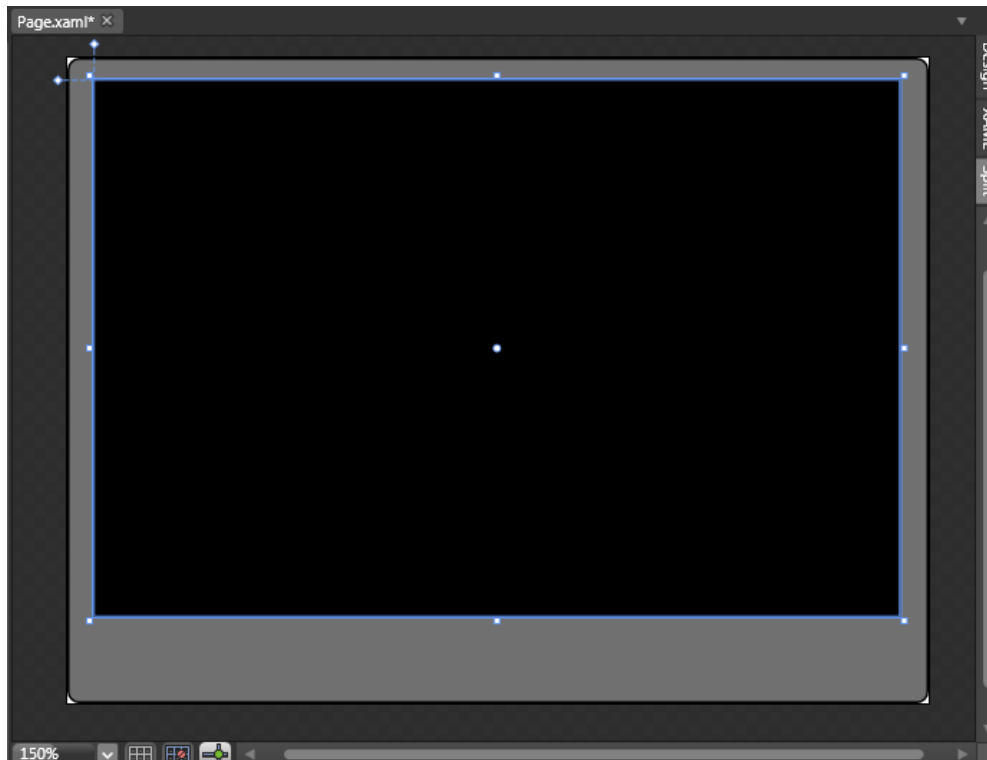
## Etape 5 : Créer le fond de la vidéo

Afin encore une fois d'avoir un rendu plus professionnel, nous allons symboliser l'emplacement de la vidéo par un rectangle noir. Ainsi si la vidéo affichée n'avait pas les bonnes dimensions, des bandes noires apparaîtraient sur les côtés ou sur le haut et le bas.

Assurez-vous de sélectionner le Canvas racine dans le sélecteur d'objets et créez un nouveau rectangle (comme précédemment) mais en utilisant les propriétés suivantes



Le Résultat attendu est celui-là :



### Etape 6 : Ajouter la vidéo

A présent que nous avons fini le design extérieur de notre lecteur vidéo, nous allons pouvoir ajouter la vidéo que nous souhaitons afficher. Pour cela nous utiliserons un objet appelé « MediaElement » qui comme son nom l'indique sert à afficher les médias. Ici nous l'utiliserons pour afficher une vidéo, cependant cet objet peut aussi lire d'autres fichiers comme les flux audio... Etant donné que la barre d'outils ne possède pas d'élément '**MediaElement**' par défaut nous allons devoir utiliser la bibliothèque d'objet pour aller le chercher.

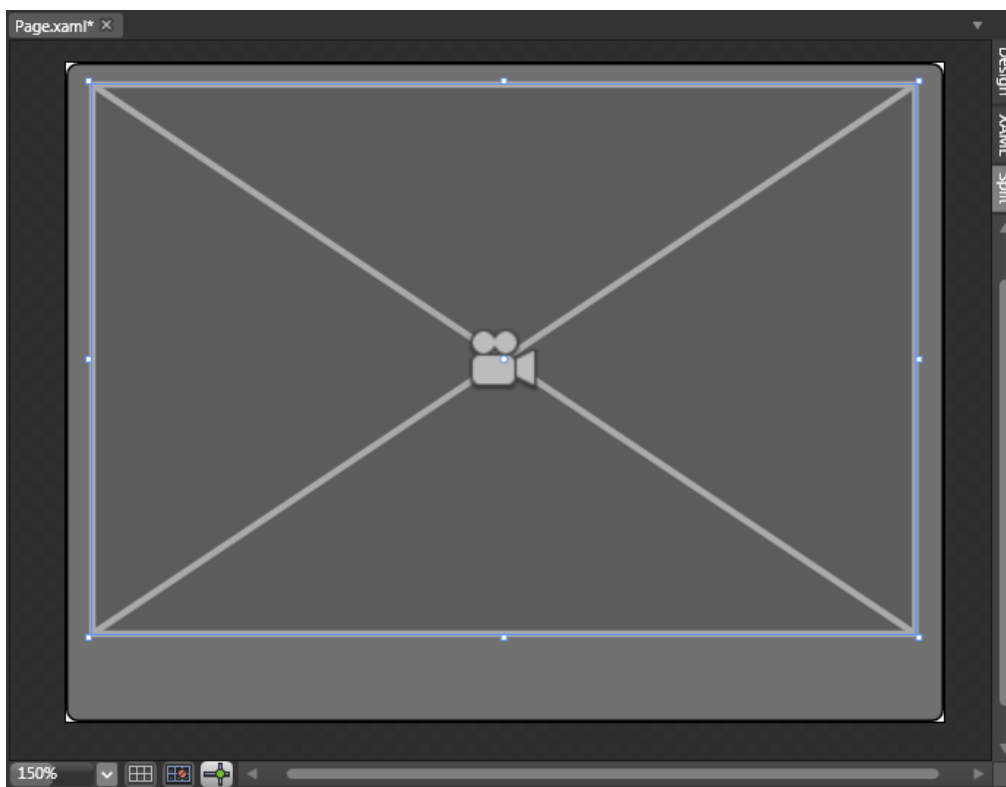
Utilisez le dernier bouton de la barre d'outils et cochez la case '**Show All**' ('Tout montrer' en anglais) puis sélectionnez '**MediaElement**' dans la liste

Dessinez sur le Canvas racine un rectangle, peu importe où et peu importe la taille. Ouvrez la fenêtre des propriétés pour cet objet et saisissez les informations suivantes

▼ Layout

Width	375
Height	250
Left	12
Top	10
ZIndex	0

Le résultat souhaité est le suivant :



Nous allons ensuite associer la vidéo que nous souhaitons afficher avec l'objet **'MediaElement'**

Dans la fenêtre propriétés de l'objet **'MediaElement'**, déployez les options « Media » en cliquant sur la petite flèche à côté du titre **'Media'**. Puis saisissez les informations ci-contre.

Notez que si la propriété **'Stretch'** (« Etirement » en anglais) n'apparaît pas, vous pouvez l'afficher en cliquant sur la flèche noire en dessous de **'Volume'**.

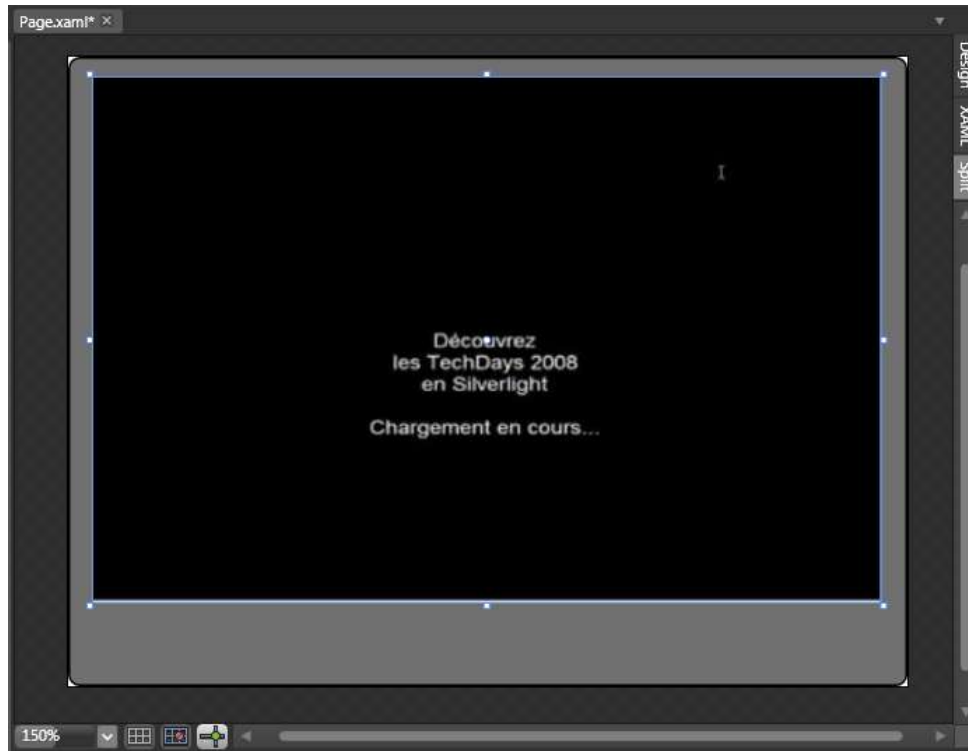
Pour sélectionner la source, cliquez sur le bouton **'...'** pour ouvrir le sélecteur de fichier. Allez sur **'C:\Techdays2008\Workshop\Silverlight\'** et sélectionnez la vidéo **'Techdays.wmv'**

▼ Media

Balance	0
IsMuted	<input checked="" type="checkbox"/>
Source	techdays.wmv
Volume	0,5
Stretch	Fill

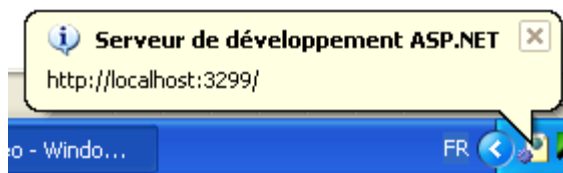
Après peu de temps (comptez 5 à 10 secondes) la vidéo sera importée dans le projet et la première image de cette vidéo apparaîtra à l'emplacement du MediaElement.

**Le résultat attendu est le suivant :**



### Etape 7 : Test du lecteur

A cette étape précise notre lecteur vidéo est déjà fonctionnel. En effet par défaut Silverlight charge et lance automatiquement une vidéo lorsqu'elle apparaît dans la scène. Pour tester votre projet il vous suffit de cliquer sur **'Test Site'** dans le menu **'Projet'** en haut à gauche de l'interface ou d'appuyer sur la touche **'F5'** (qui sera familière aux développeurs Visual Studio). Vous verrez alors Expression Blend lancer le serveur de développement ASP.NET (Cassini) dans la barre des tâches



et Internet Explorer affichera en quelques secondes votre projet à l'écran comme ci-dessous

*(Voir image page suivante)*



Cette partie est à présent terminée. Dans la prochaine partie nous verrons comment ajouter de l'interactivité au moyen d'un bouton et des événements de Silverlight

S'il vous reste du temps vous pouvez vous amuser à modifier les propriétés des objets pour voir comment ceux-ci réagissent.


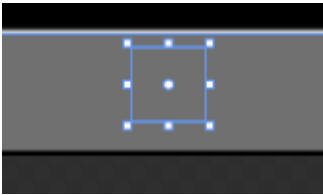
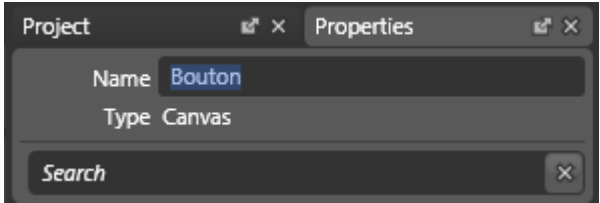
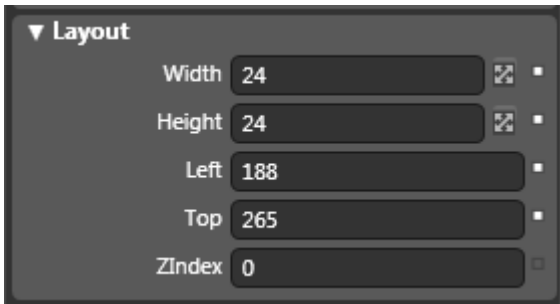
## Deuxième partie : Ajoutons un peu d'interactivité !

Bien que notre lecteur vidéo soit d'ors et déjà fonctionnel, il est pour le moment vraiment très sommaire. L'une des premières (et primordiales) fonctionnalités à lui ajouter est un bouton permettant de déclencher la lecture. En effet il n'est rien de plus désagréable que d'arriver sur un site et qu'une vidéo se mette en marche toute seule. Pour peu que cette vidéo possède du son et que vous écoutiez de la musique c'est la cacophonie assurée ! Nous allons donc bloquer la lecture de notre vidéo et ajouter un bouton permettant de lancer la vidéo.

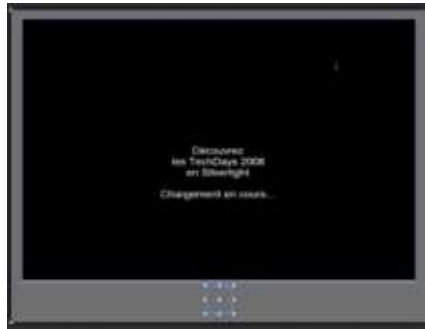
Il n'existe pas en Silverlight 1.0 d'objet bouton tout prêt, cependant tout ce qu'il nous faut pour faire notre bouton est fourni. Presque tous les objets répondent à l'évènement **'MouseLeftButtonDown'** qui est déclenché lorsque l'on clique avec le bouton gauche de la souris sur un élément. Nous allons donc utiliser cette faculté pour créer notre propre bouton.

### Etape 8 : Ajout d'un Canvas bouton

La première étape à réaliser lorsque l'on veut créer un nouveau contrôle est de l'inclure dans son propre Canvas, ainsi si nous voulons lui appliquer des propriétés ou des facultés particulières nous pourrons facilement l'isoler en utilisant ce Canvas.

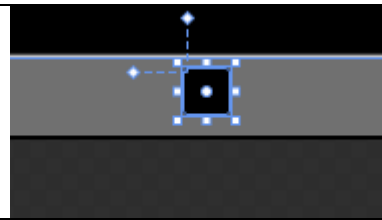
Sélectionnez l'outil Canvas et assurez-vous que le Canvas Racine ( <b>'Page'</b> ) soit bien sélectionné.	
Dessinez un petit <b>Canvas</b> sous la vidéo.	
En utilisant la fenêtre des propriétés, donnez à ce nouveau Canvas le nom <b>'Bouton'</b>	
Assignez-lui les propriétés suivantes	

Le résultat doit ressembler à ceci :

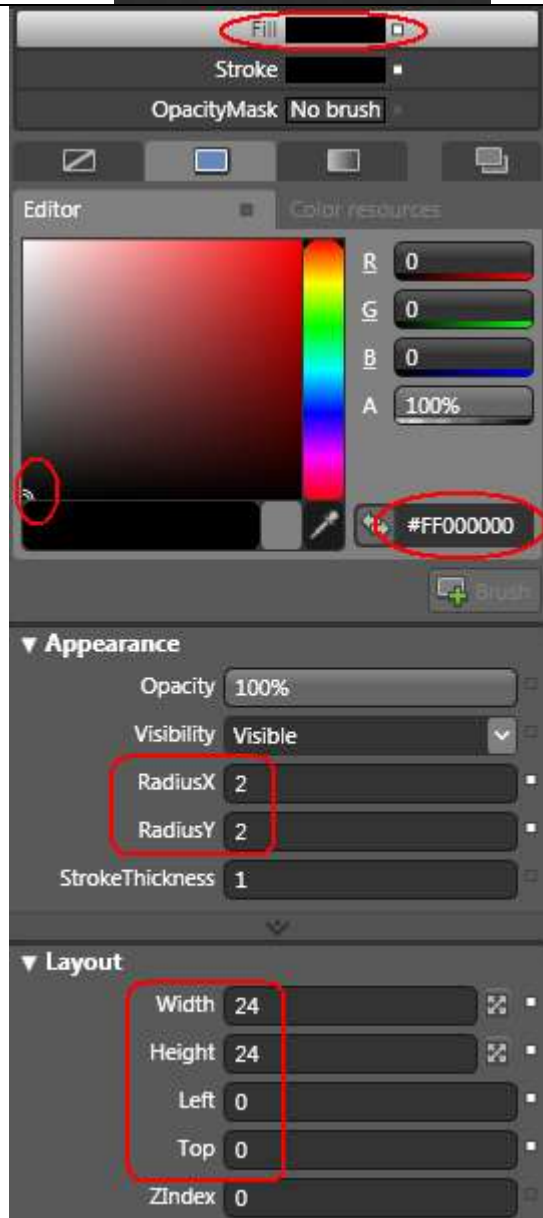


### Etape 9 : Ajout d'un fond pour notre bouton

Assurez-vous que le Canvas **'Bouton'** soit sélectionné et en utilisant l'outil **'Rectangle'** dessinez un rectangle à l'intérieur du Canvas (peu importe la taille et le positionnement)



Définissez à présent les propriétés de ce nouveau rectangle comme sur l'image


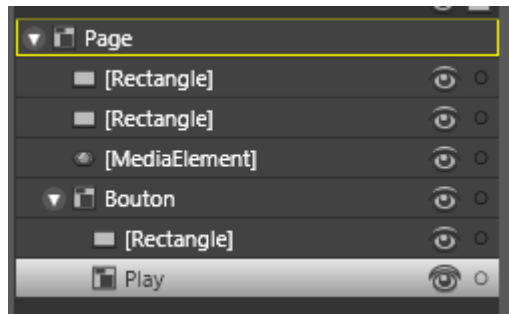


Vous devriez obtenir le résultat suivant :



### Etape 10 : Ajout d'un Canvas 'Play'

Bien que cette étape ne soit pas nécessaire à ce stade vous comprendrez dans la prochaine partie pourquoi nous utilisons encore un nouveau Canvas pour y placer le symbole « Lecture » de notre bouton. La première étape consiste à ajouter un nouveau Canvas dans le Canvas **'bouton'**

<p>Assurez-vous que le Canvas <b>'bouton'</b> soit sélectionné et en utilisant l'outil <b>'Canvas'</b> dessinez un nouveau Canvas au dessus du carré noir. Nommez-le <b>'Play'</b> et donnez-lui les propriétés suivantes :</p>	
<p>Votre hiérarchie devrait ressembler à ceci :</p>	

Nous allons maintenant ajouter sur ce Canvas le symbole « Lecture » usuellement représenté par un triangle. Nous allons utiliser un objet qui n'est pas présent dans la barre d'outils de Blend (pas même en passant pas la bibliothèque). Gageons que ceci vient du fait que c'est une préversion et que la version finale aura cet outil. Comme nous ne pouvons pas utiliser le designer nous allons écrire le code directement dans la fenêtre XAML. L'objet en question est « Polygon » qui permet de créer des... Polygones !

Cependant avant de créer notre triangle nous allons devoir modifier légèrement notre code XAML. En effet si vous ouvrez la fenêtre XAML et que vous recherchez la ligne définissant votre Canvas 'Play' vous trouverez la ligne suivante :

```
<Canvas Width="24" Height="24" x:Name="Play" Canvas.Left="0" Canvas.Top="0" />
```

Ce qui nous empêche de pouvoir ajouter du contenu au Canvas directement (puisque c'est une balise unique). Nous allons donc transformer cette balise unique en balise régulière.

**Note :** Comme vous l'aurez sûrement remarqué le code XAML est basé sur code XML, tout y est indiqué par des balises (ex : <Canvas>). Chaque balise ouverte doit être refermée par une balise équivalente (ex : </Canvas>). Il existe des balises uniques c'est-à-dire ne possédant pas de balise fermante, elles sont symbolisées ainsi : <Canvas />. Pour pouvoir placer du contenu dans un objet on doit utiliser la notation <Balise>Contenu</Balise> ou <Balise Attribut= "Contenu" />

Supprimez le « / » à la fin de la balise et ajouter derrière « > » le code « </Canvas > » (Attention au majuscule). Vous devriez obtenir le résultat suivant :

```
13 <Canvas Width="24" Height="24" x:Name="Play" Canvas.Left="0" Canvas.Top="0"></Canvas>
```

Passez ensuite la balise fermante à la ligne et ajouter une ligne vide comme ceci :

```
<Canvas Width="24" Height="24" x:Name="Play" Canvas.Left="0" Canvas.Top="0">
</Canvas>
```

Ajouter la ligne suivante entre les deux balises <Canvas> et </Canvas> :

```
<Polygon Canvas.Left="6" Canvas.Top="4" Points="0,0 0,16 14,8" Opacity="1" Fill="#FFFFFFF"
Stroke="#FF5B5B5B" />
```

Note : Taper l'ensemble sur 1 ligne

```
<Canvas Width="24" Height="24" x:Name="Play" Canvas.Left="0" Canvas.Top="0">
    <Polygon Canvas.Left="6" Canvas.Top="4" Points="0,0 0,16 14,8" Opacity="1" Fill="#FFFFFFF"
    </Canvas>
```

Le résultat attendu est le suivant



## Etape 11 : Blocage de la lecture automatique et ajout du gestionnaire d'évènement

Tant que nous sommes dans le code XAML nous allons en profiter pour modifier 2 points :

- Bloquer la lecture automatique de la vidéo au démarrage grâce à l'attribut **'AutoPlay'**
- Ajouter un gestionnaire d'évènement pour **'MouseLeftButtonDown'** sur le Canvas **'Play'**, c'est-à-dire une fonction qui se déclenchera lorsque nous cliquerons sur le Canvas **'Play'** avec le bouton gauche de la souris

**Note :** Une des particularités de Silverlight (héritée de WPF) est la possibilité d'avoir des évènements routés. En effet lorsque vous déclenchez un évènement sur un objet (par exemple en cliquant dessus) et que cet objet ne gère pas cet évènement (ne définit pas de gestionnaire d'évènement), cet objet va transmettre l'évènement déclenché à son Parent qui pourra alors le transmettre lui aussi à son Parent s'il ne gère pas cette évènement.

On voit ainsi l'importance de la hiérarchie dont nous parlions dans la première partie. Ici bien que le gestionnaire d'évènement soit placé sur le Canvas **'Play'**, si nous cliquons sur le Polygon **'[Triangle]'** celui-ci ne gérant pas l'évènement le transmettra à son Parent (le Canvas **'Play'**) et le tour est joué ! Nous aurons ainsi bien reçu l'évènement sur le canvas **'Play'** alors que nous aurons cliqué sur le triangle.

### a) Bloquer la vidéo au démarrage

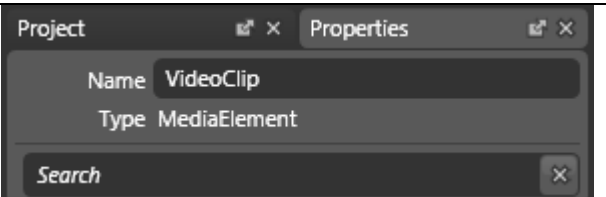
Recherchez la ligne suivante dans le code XAML :

```
<MediaElement Width="375" Height="250" Canvas.Left="12" Canvas.Top="10"
IsMuted="True" Source="techdays.wmv" Stretch="Fill" />
```

Et ajoutez le code suivant : **AutoPlay="False"**. Le résultat est le suivant :

```
<MediaElement AutoPlay="False" Width="375" Height="250" Canvas.Left="12" Canvas.T
```

Tant que l'objet MediaElement est sélectionné, utilisez la fenêtre de propriétés pour le nommer **'VideoClip'**



### b) Associer un gestionnaire d'évènement pour l'évènement MouseLeftButtonDown

Recherchez la ligne suivante dans le code XAML :

```
<Canvas Width="24" Height="24" x:Name="Play" Canvas.Left="0" Canvas.Top="0">
```

Et ajoutez le code suivant : **MouseLeftButtonDown="onPlay"**. Le résultat est le suivant :

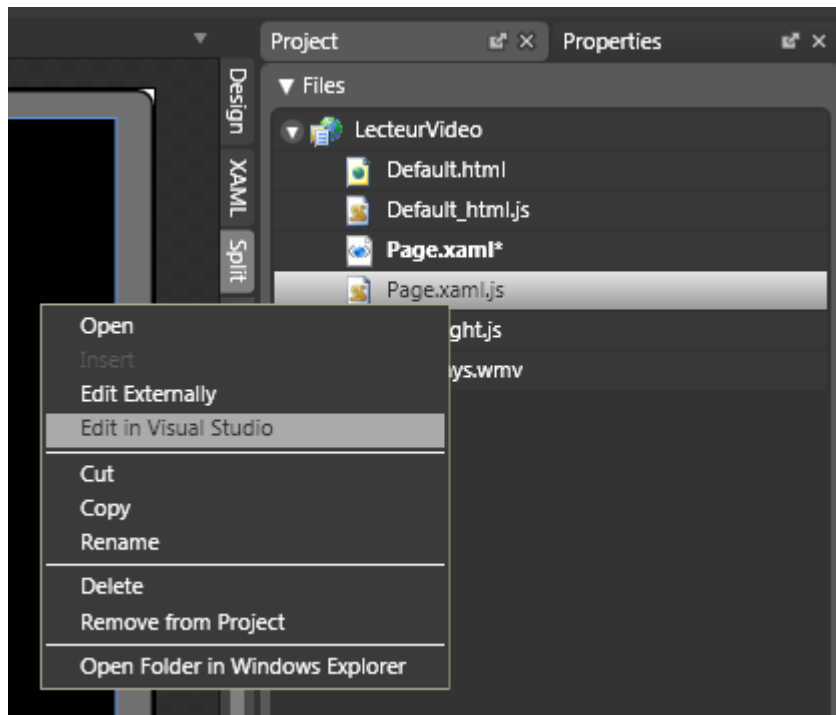
```
<Canvas MouseLeftButtonDown="onPlay" Width="24" Height="24" x:Name="Play" Canva
```

Ce code définit une fonction JavaScript appelée « onPlay » qui sera exécutée chaque fois que nous cliquerons avec le bouton gauche sur le Canvas **'Play'** (ou un des ses enfants ne définissant pas de gestionnaire pour cet évènement)

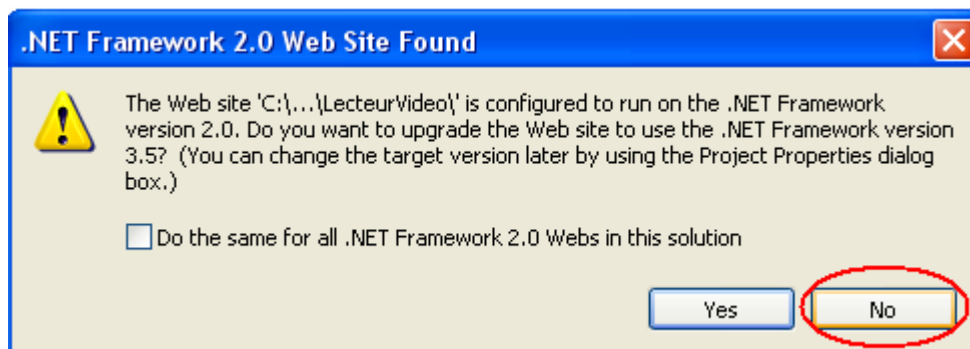
## Etape 12 : Ajouter la fonction JavaScript « onPlay »

Nous allons à présent ajouter la fonction JavaScript « onPlay » qui sera appelée par l'évènement '**MouseButtonDown**' du Canvas '**Play**'. Bien que nous pourrions utiliser l'éditeur de code de Blend, nous allons passer par Visual Studio 2008 et voir à quel point les deux produits sont bien intégrés l'un à l'autre.

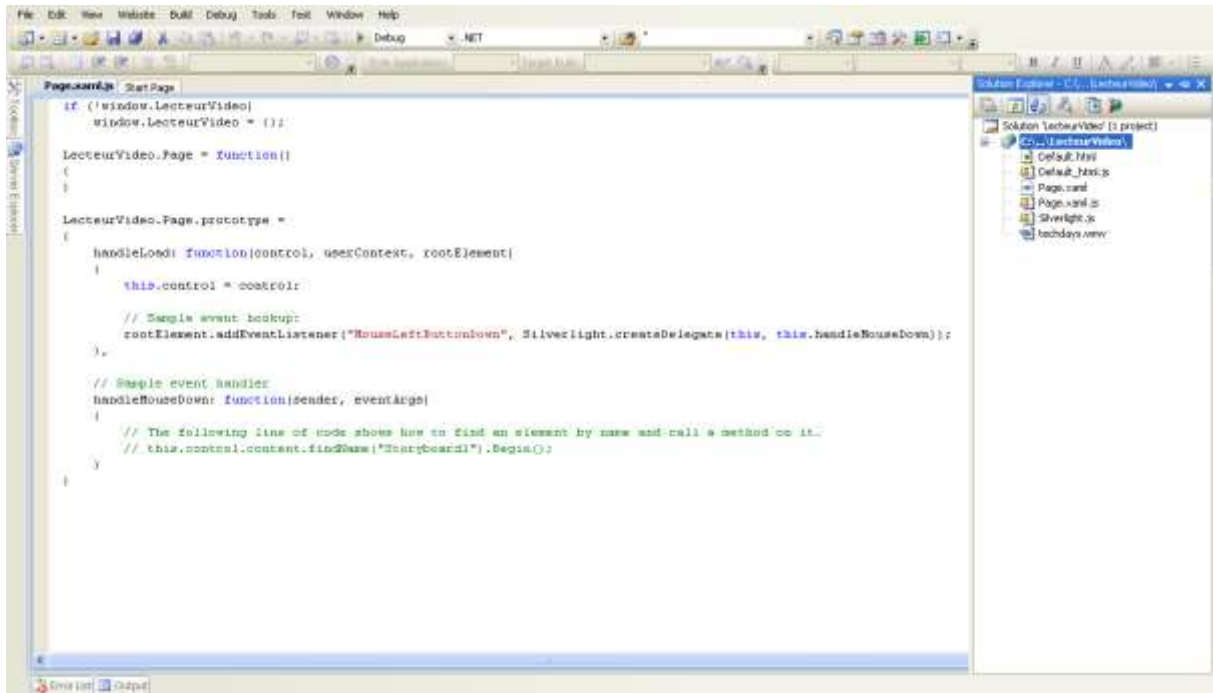
La première étape est d'ouvrir le fichier '**Page.xaml.js**' dans Visual Studio. Il suffit simplement de cliquer sur la fenêtre '**Project**' puis de faire un clic droit sur le fichier '**Page.xaml.js**' et enfin de sélectionner '**Edit in Visual Studio**' dans le menu contextuel.



Le projet s'ouvre alors dans Visual Studio 2008. Il est possible qu'une fenêtre s'ouvre pour vous demander de convertir le projet vers le Framework 3.5. Répondez 'No'



Une fois dans Visual Studio 2008, le fichier '**Page.xaml.js**' est automatiquement ouvert et l'arborescence du projet disponible via le « Solution Explorer » (Cf ci-dessous)



Editez le code JavaScript pour modifier la fonction **'handleLoad :'**, supprimez la fonction **'handleMouseDown :'** et ajoutez la fonction **'onPlay'** comme suit :

```

Page.xaml.js* Start Page
if (!window.LecteurVideo)
    window.LecteurVideo = {};

LecteurVideo.Page = function()
{
}

function onPlay(sender, args) {
    sender.findName("VideoClip").play();
}

LecteurVideo.Page.prototype =
{
    handleLoad: function(control, userContext, rootElement)
    {
    }
}

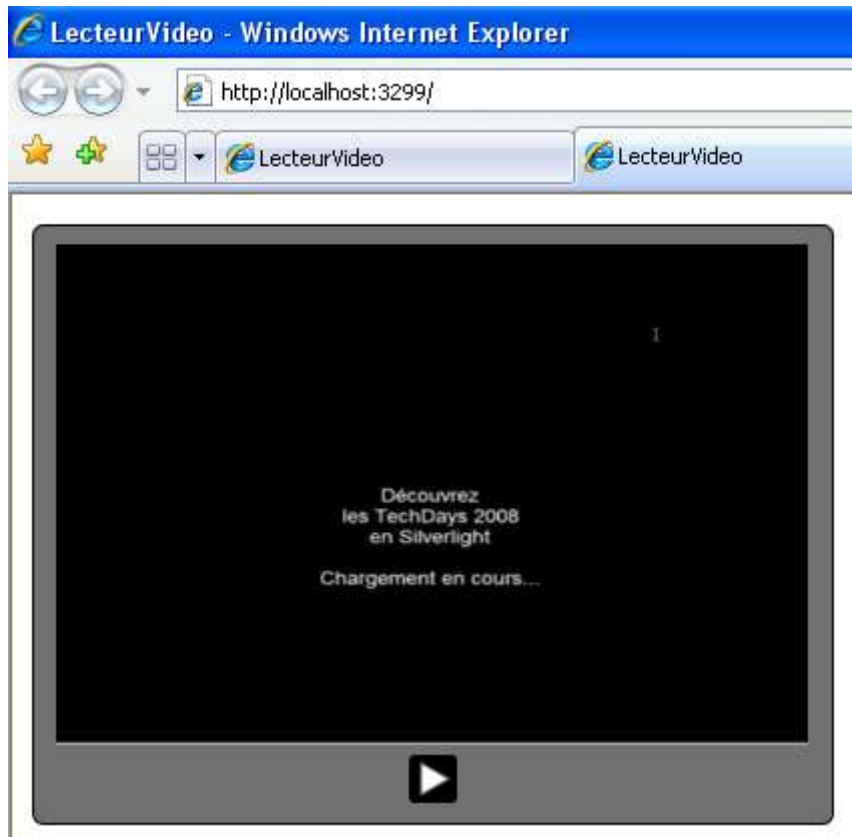
```

La fonction **'onPlay'** attend deux arguments **'sender'** qui représente l'objet ayant déclenché l'évènement (ici le Canvas **'Play'**) et **'args'** qui est un objet permettant de transférer des paramètres à la fonction si nécessaire. Cette fonction réalise une seule action pour le moment : elle trouve via la fonction **'findName'** l'élément **'VideoClip'** (d'où la nécessité de nommer notre **MediaElement**) et exécute la fonction **'play()'** qui permet de démarrer la lecture de la vidéo.

Une fois ce code saisi vous pouvez sauvegarder la page et fermer Visual Studio (si celui-ci demande de sauvegarder le projet répondez **'Yes'** et acceptez le nom de solution par défaut)

### Etape 13 : Test du lecteur

Vous pouvez maintenant tester votre lecteur en appuyant sur 'F5' depuis Blend et tester votre bouton. Le résultat attendu est celui-ci :



## Troisième partie : Ajoutons un peu d'animation !

Notre lecteur vidéo a bien avancé dans la deuxième partie, mais il ne saurait convenir aux utilisateurs du Web avertis que nous sommes. Une des fonctionnalités que nous pourrions lui ajouter est un bouton **'Pause'**. Nous pourrions bien-sûr réitérer la manœuvre effectuée dans la Partie 2 en changeant l'appel de **'play()'** par un appel à **'pause()'** mais ce ne sera pas amusant et instructif ☺.

Je vous propose de créer un bouton Play/Pause animé. Lorsque la vidéo est arrêtée un bouton **'Play'** apparaît ; si l'on clique dessus la vidéo se lance et une animation transforme le bouton **'Play'** en un bouton **'Pause'** et vice versa. Plutôt cool non ?

Dans la deuxième partie je vous avais expliqué qu'il n'était pas strictement nécessaire de faire un Canvas **'Play'** pour réaliser notre bouton. Je vais à présent vous expliquer pourquoi nous avons utilisé ce subterfuge.

Afin de réaliser notre animation nous allons utiliser un deuxième Canvas appelé **'Pause'** sur lequel nous dessinerons deux rectangles côte à côte symbolisant « Pause ». Nous animerons ensuite chacun de ses Canvas pour les faire apparaître ou disparaître aux grés de nos besoins. Et voici pourquoi nous avons besoin de Canvas, il est ainsi plus facile de manipuler les éléments de notre bouton 'Play' et ceux de notre bouton 'Pause' afin de les animer, nous n'aurons qu'à utiliser leurs Canvas parents !

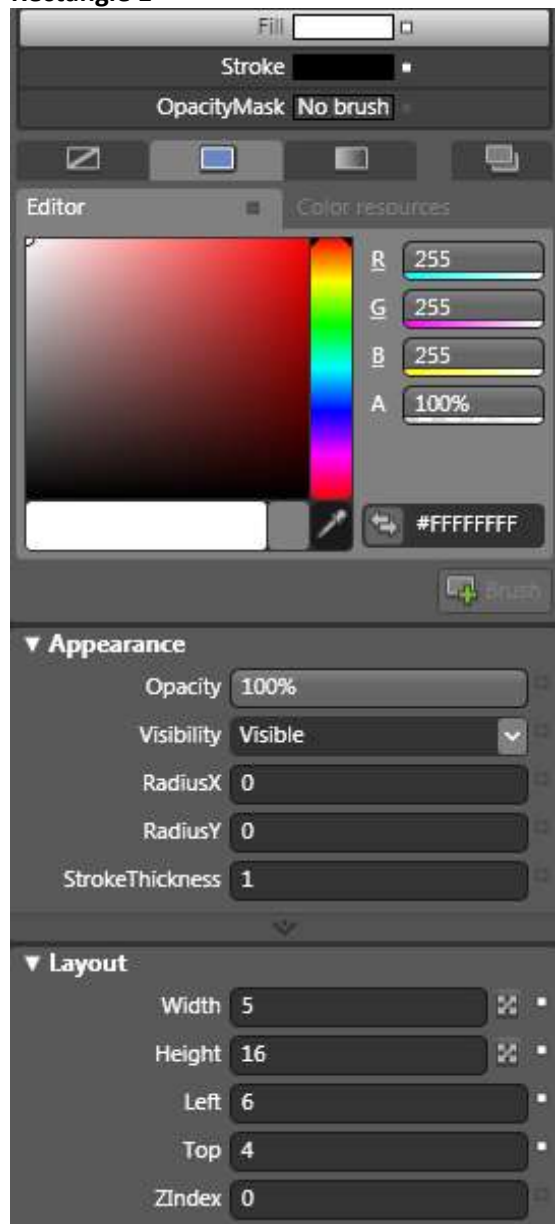
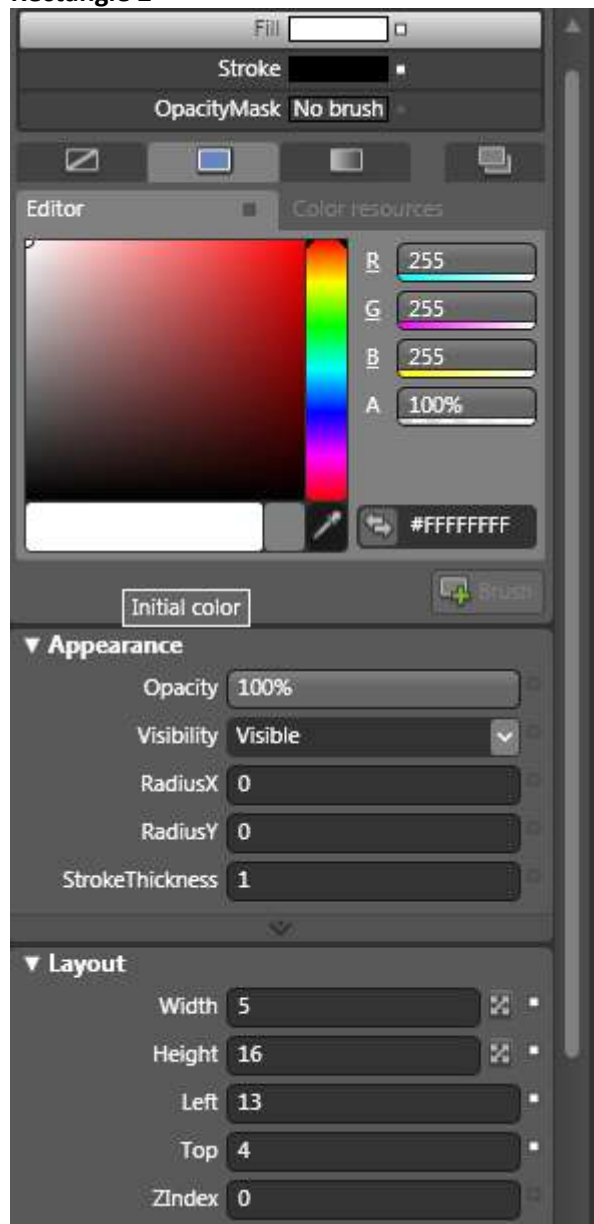
### Etape 14 : Création d'un bouton Pause

La création du bouton **'Pause'** est similaire à celle du bouton **'Play'** bien entendu en remplaçant le triangle par deux rectangles. Voici la démarche ci-dessous.

Créez tout d'abord un nouveau Canvas en tant qu'enfant du Canvas **'Bouton'** et donnez lui les propriétés suivantes (ce sont exactement les mêmes que pour le Canvas 'Play' excepté le nom)



Ajoutez ensuite dans ce Canvas 'Pause' deux nouveaux rectangles avec les propriétés suivantes :

Rectangle 1	Rectangle 2
	

Le résultat attendu est le suivant :



## Etape 15 : Ajouter un comportement au bouton 'Pause'

A présent nous allons coder le comportement de notre bouton **'Pause'**. Pour cela nous allons comme pour le Canvas **'Play'** gérer l'évènement **'MouseLeftButtonDown'** avec une fonction JavaScript appelée « onPause ».

Assurez-vous que le Canvas **'Pause'** soit sélectionné et dans le code XAML correspondant ajoutez le code **MouseLeftButtonDown="onPause"**, le résultat doit être :

```
<Canvas MouseLeftButtonDown="onPause" Width="24" Height="24" x:Name="Pause" Canvas
  <Rectangle Width="5" Height="16" Fill="#FFFFFF" Stroke="#FF0000" Canvas
```

A présent depuis la fenêtre **'Project'** ouvrez le fichier **'Page.xaml.js'** dans Visual Studio 2008 (clic droit -> *Edit in Visual Studio*). Puis dans le code JavaScript juste en dessous de la fonction **'onPlay'**, ajoutez le code suivant pour la fonction **'onPause'** :

```
function onPlay(sender, args) {
    sender.findName("VideoClip").play();
}

function onPause(sender, args) {
    sender.findName("VideoClip").pause();
}
```

**Note** : Sauvegardez votre fichier **'Page.xaml.js'** mais ne fermez pas Visual Studio 2008, nous aurons besoin de modifier ce fichier d'avantage plus tard.

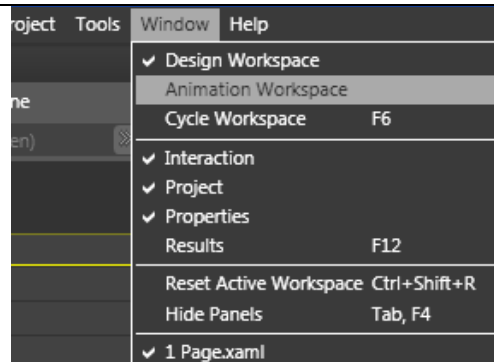
**Note** : Ne tester pas l'application pour le moment, en effet les deux boutons se superposent et le bouton **'Play'** étant en dessous il ne recevra jamais l'évènement **'MouseLeftButtonDown'** donc la vidéo ne sera jamais lancée...

## Etape 16 : Ajouter l'animation de 'Play' à 'Pause'

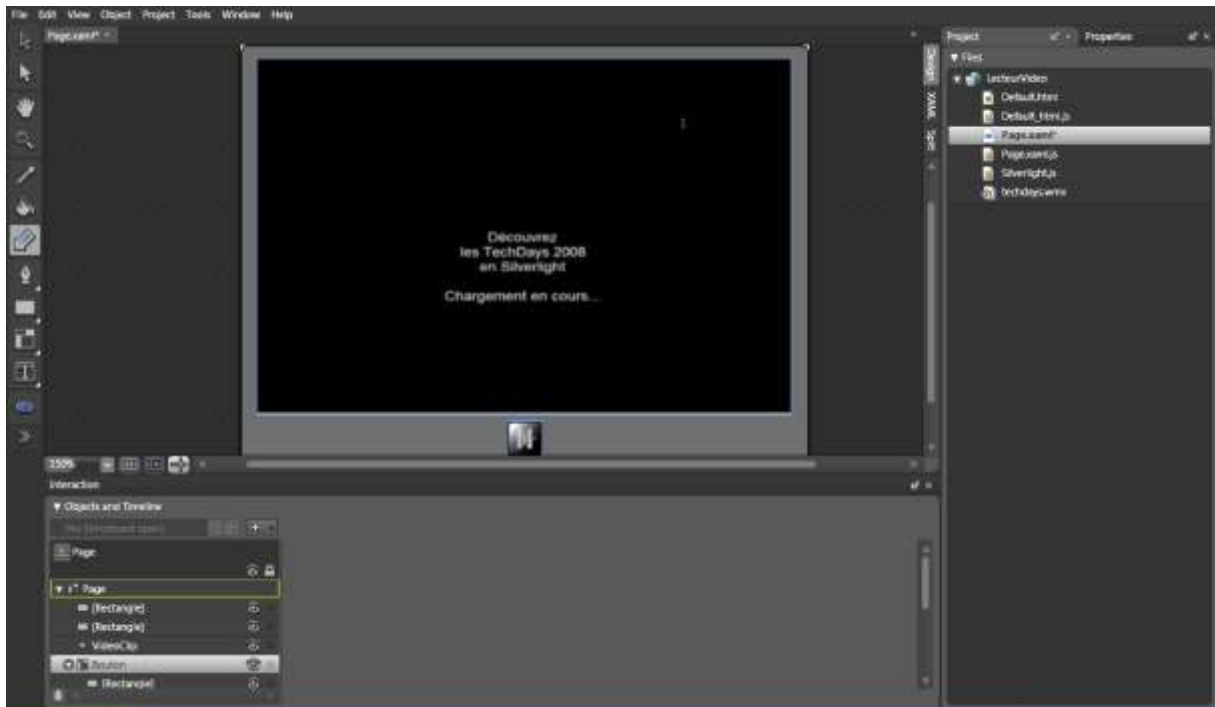
Les animations dans Silverlight fonctionnent avec des **Storyboards** qui sont des objets permettant de définir des modifications sur les objets Silverlight au cours du temps.

Pour créer un **Storyboard** la première des étapes est de passer en mode **'Animation'**, ce n'est pas obligatoire mais cela va organiser les panneaux pour travailler plus facilement avec la règle de temps.

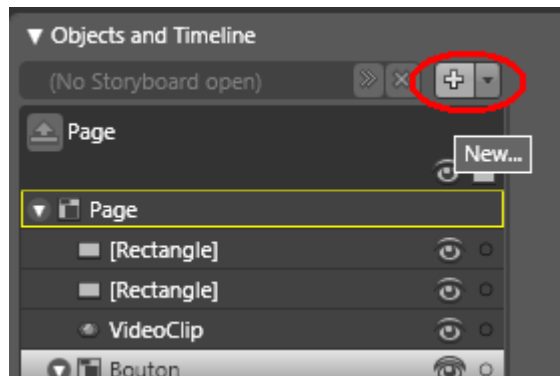
Pour passer en mode **'Animation'** utilisez le menu **'Windows'** et cliquez sur **'Animation Workspace'**



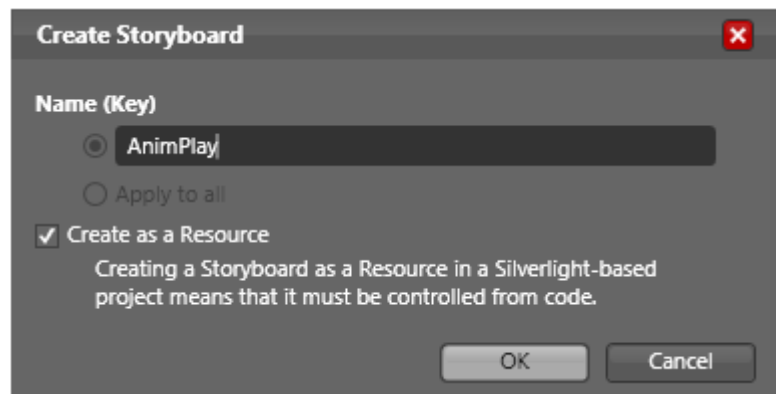
Votre interface va être modifiée pour afficher le sélecteur d'objets et de scénarios en bas :



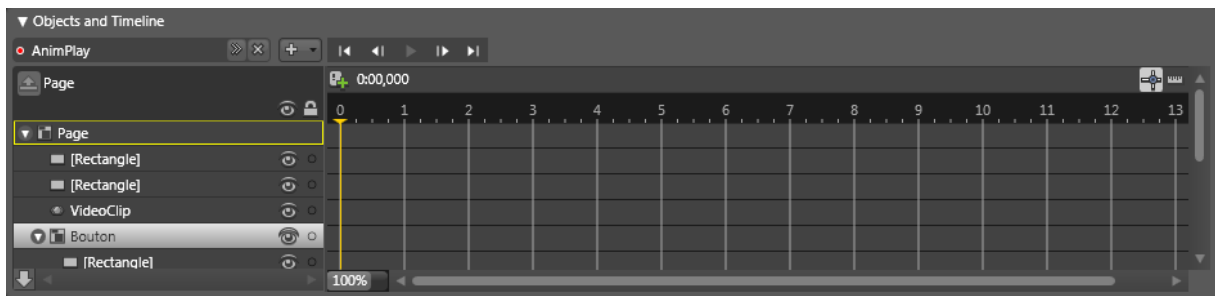
Une fois dans cette interface nous allons pouvoir ajouter un nouveau **'Storyboard'** en utilisant le bouton '+' dans le sélecteur d'objets



Dans la fenêtre de création qui s'ouvre, entrez les informations suivantes et cliquez sur OK



Vous devriez à présent avoir une règle de temps vous permettant de créer votre animation :



De plus l'espace de travail s'entoure d'un liseré rouge symbolisant le mode animation



Les Storyboards fonctionnent avec des **'Keyframes'** (images clés en anglais), ces images clés sont des points dans le temps définissant des informations à propos des objets (comme par exemple leur taille, leur emplacement à ce moment précis, etc...).

Si vous définissez deux images clés pour un même objet à deux temps différents et que ces images clés contiennent des informations différentes (comme par exemple une position différente pour l'élément) alors Silverlight créera une interpolation entre ces images (c'est-à-dire qu'il calculera au moment de l'exécution toutes les images intermédiaires permettant de passer de l'état A à l'état B), créant ainsi une animation par interpolation.


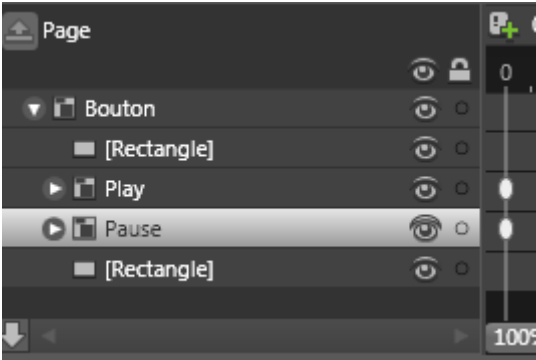
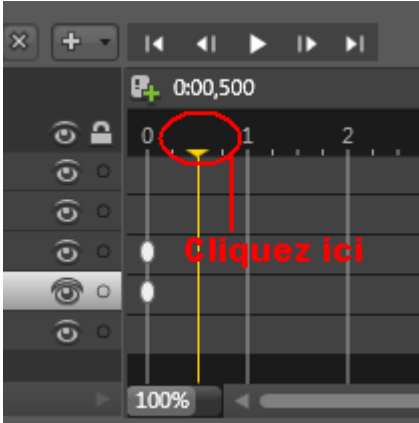
Par défaut cette animation est linéaire ce qui signifie que la transition entre A et B se fera à part égale sur toutes les images intermédiaires.

**Note :** L'interpolation linéaire est le mode d'animation par défaut de Blend, c'est aussi le plus utilisé mais ce n'est pas le seul type d'animation disponible.

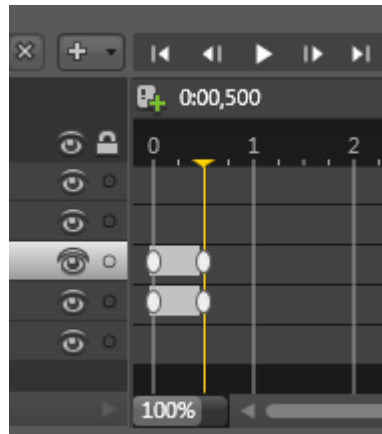
Dans notre cas nous souhaitons que le bouton **'Play'** disparaisse progressivement et que le bouton **'Pause'** apparaisse progressivement. Nous allons donc animer les valeurs d'opacité de ces Canvas.

- A t=0 seconde le bouton **'Play'** aura 100% d'opacité alors que le bouton **'Pause'** en aura 0%
- A t=0.5 secondes le bouton **'Pause'** aura 100% d'opacité alors que le bouton **'Play'** sera à 0%

**Note :** en réalité pour une animation plus fluide les valeurs 100% et 0% seront remplacées par les valeurs 95% et 5% pour les images de départ

<p>Assurez-vous que la ligne jaune surmontée d'une flèche inversée (<i>représentant le curseur de temps</i>) soit positionnée sur 0. Voir ci-contre</p> <p>Dans la liste des objets, sélectionnez le Canvas <b>'Play'</b> et ensuite modifiez (<i>via la fenêtre de propriété</i>) sa valeur d'Alpha à 95%.</p> <p>Un point blanc apparaît sur la ligne de temps symbolisant qu'une image clé est définie pour cet élément à ce temps précis.</p>	
<p>Sélectionnez le Canvas <b>'Pause'</b> et placez sa valeur d'opacité (Alpha) à 5%.</p> <p>Vous devriez obtenir à présent deux points blancs dans la ligne de temps (<i>cf image ci-contre</i>)</p>	
<p>Placer maintenant votre tête de lecture (<i>flèche jaune</i>) au temps 0.5 seconde</p> <p>Définissez pour ce temps une opacité de 0% pour le Canvas <b>'Play'</b> et 100% pour le Canvas <b>'Pause'</b>.</p>	

Le résultat devrait être ceci :



Votre animation pour le passage de **'Play'** à **'Pause'** est terminée vous pouvez cliquer sur le bouton de lecture au dessus du Storyboard pour visualiser l'animation.

### Etape 17 : Animation de Pause à Play

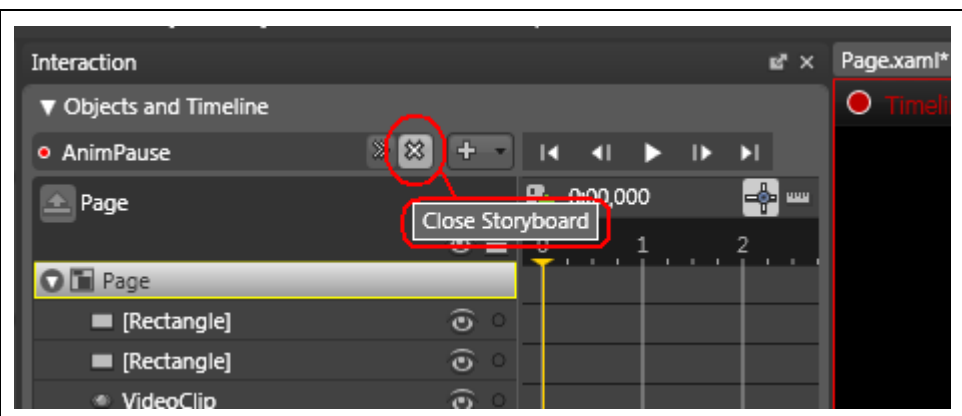
L'animation de **'Pause'** à **'Play'** est très proche de la précédente animation. Ajouter une nouveau **Storyboard** et nommez le **'AnimPause'**

Pour le temps T=0 seconde, définissez une opacité de 5% pour la Canvas **'Play'** et 95% pour le Canvas **'Pause'**. Amener votre tête de lecture sur T=0.5 seconde et définissez une opacité de 0% pour **'Pause'** et 100% pour **'Play'**.

### Etape 18 : Déclenchement des animations

Repasssez en mode **'Design'** via le menu **'Windows'**. Le sélecteur d'objets affiche à présent le dernier **Storyboard** utilisé, comme nous ne souhaitons plus modifier les **Storyboards** et que cet affichage prend de la place nous allons fermer le **Storyboard**.

Cliquez sur la croix à droite du nom du Storyboard pour fermer celui-ci



La première des modifications que nous allons apporter à présent est de masquer au chargement de la page le bouton **'Pause'** qui ne doit apparaître que lorsque la vidéo est en lecture. Nous l'empêcherons ainsi de capter les événements qui pourront alors être envoyés au bouton **'Play'**.

**Note :** La propriété **'Opacity'** d'un objet gère sa transparence. A 100% l'élément est totalement opaque alors qu'à 0% il est totalement transparent donc invisible.  
La propriété **'Visibility'** (valeur **'Visible'** ou **'Collapsed'**) permet de savoir si un élément est visible à l'écran. On pourrait se demander quel est l'intérêt d'une telle propriété lorsqu'il suffit de mettre l'opacité d'un élément à 0% pour le rendre invisible... Outre un gain de performance, l'avantage de la propriété **'Visibility'** est qu'elle empêche un élément invisible (**'Collapsed'**) de recevoir les événements, alors qu'un élément transparent (**'Opacity'** à 0%) continue de capter ces événements.

Ouvrez Visual Studio et sélectionnez le fichier **'Page.xaml.js'**. Modifiez la fonction **'handleLoad :'** pour qu'elle ressemble à ceci :

```
LecteurVideo.Page.prototype =
{
  handleLoad: function(control, userContext, rootElement)
  {
    rootElement.findName("Pause").Visibility = "Collapsed";
  }
}
```

Pour que notre lecteur vidéo fonctionne correctement nous devons aussi nous assurer qu'un seul bouton reçoive les clics de souris à tout moment. Ce bouton devra être différent en fonction de l'état de lecture, si la vidéo est en pause ce devra être le bouton **'Play'**, si la vidéo est en lecture ce sera le bouton **'Pause'**.

Nous devons donc masquer (avec la propriété **'Visibility'**) à la fin de chaque animation le bouton que l'on vient de rendre transparent. De la même façon nous devons réafficher le bouton à chaque fois que nous appellerons une animation dessus (de **'Play'** à **'Pause'** et vice versa).

Pour cela nous allons devoir ajouter un gestionnaire d'événements pour chaque **Storyboard**. Ce gestionnaire répondra à l'évènement **'Completed'** de l'objet **Storyboard**. De plus nous allons devoir ajouter un peu de code sur les gestionnaire d'événements **'onPlay'** et **'onPause'** pour réafficher les boutons avant de les animer.

Dans la fenêtre de code XAML, trouvez le tag correspondant au **Storyboard 'AnimPlay'** et ajouter lui le code **Completed="onAnimPlayCompleted"** associant le gestionnaire d'évènement **'onAnimPlayCompleted'** à l'évènement **'Completed'** de l'objet **Storyboard** :

```
Canvas.Resources >
  <Storyboard x:Name="AnimPlay" Completed="onAnimPlayCompleted">
    <DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard
```

Faites de même pour le **Storyboard 'AnimPause'** avec la fonction **'onAnimPauseCompleted'** :

```
</Storyboard>
<Storyboard x:Name="AnimPause" Completed="onAnimPauseCompleted">
  <DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard
```

Une fois ceci fait, ouvrez la page *'Page.xaml.js'* dans Visual Studio 2008 et modifiez votre code pour qu'il ressemble à ceci :

```
LecteurVideo.Page = function()
{
}

function onPlay(sender, args) {
    sender.findName("Pause").Visibility = "Visible";
    sender.findName("AnimPlay").Begin();
    sender.findName("VideoClip").play();
}

function onPause(sender, args) {
    sender.findName("Play").Visibility = "Visible";
    sender.findName("AnimPause").Begin();
    sender.findName("VideoClip").pause();
}

function onAnimPlayCompleted(sender, args) {
    sender.findName("Play").Visibility = "Collapsed";
}

function onAnimPauseCompleted(sender, args) {
    sender.findName("Pause").Visibility = "Collapsed";
}

LecteurVideo.Page.prototype =
{
    handleLoad: function(control, userContext, rootElement)
    {
        rootElement.findName("Pause").Visibility = "Collapsed";
    }
}
```

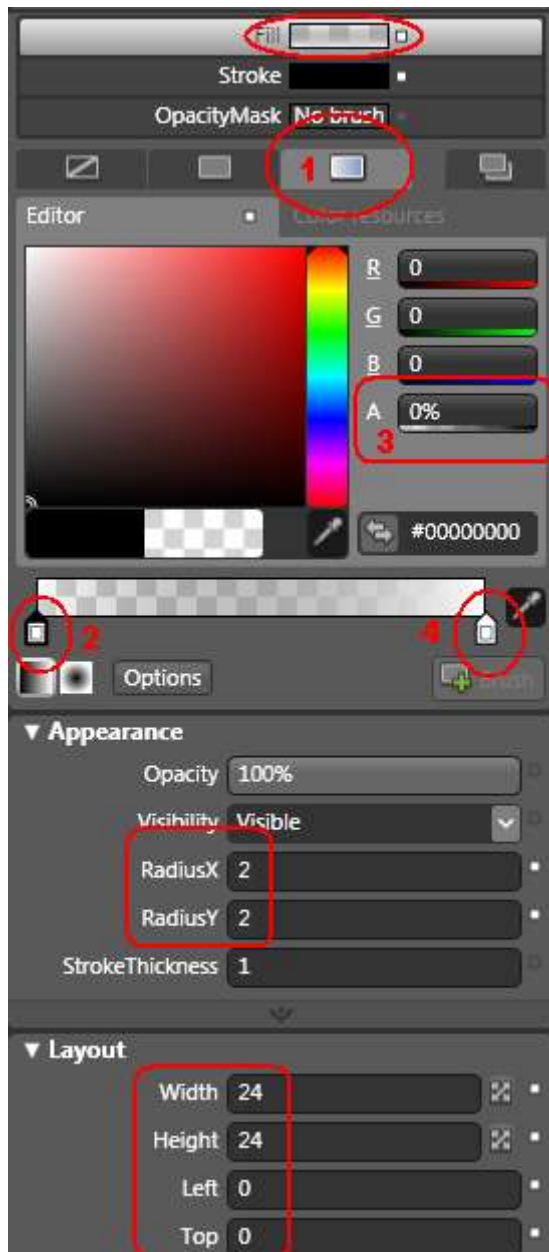
Ce code ajoute une instruction au gestionnaire d'évènements *'onPlay'* et *'onPause'* (pour afficher les boutons masqués), et deux nouveaux gestionnaires d'évènements *'onAnimPlayCompleted'* et *'onAnimPauseCompleted'* qui masquent les boutons à la fin des animations.

### Etape 19 (optionnelle) : Ajouter un effet d'illumination

C'est un effet très facile à faire que l'on voit de plus en plus sur Internet. Pour réaliser cet effet nous allons utiliser un rectangle disposé au dessus du bouton qui aura un remplissage fait d'un dégradé semi-transparent.

Assurez-vous que le Canvas *'Bouton'* soit sélectionné et dessinez dessus un nouveau rectangle ayant les propriétés suivantes :

(Cf image page suivante)


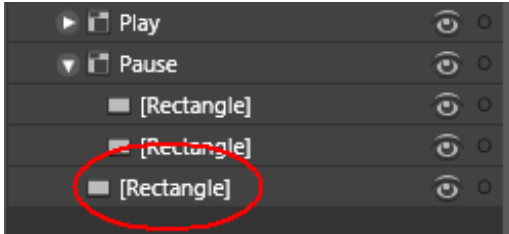



Note : Pour réaliser le dégradé, assurez vous que la propriété **'Fill'** soit sélectionnée (Zone en haut de l'image), puis cliquez sur un remplissage par dégradé (Zone 1 dans l'image). Dans la barre de dégradé sélectionnez le curseur gauche (Zone 2) et placer sa valeur **Alpha à 0%** (Zone 3). Le curseur de droite ne doit pas être modifié (Zone 4) c'est-à-dire **'Blanc avec alpha à 100%'**

**Le résultat attendu est le suivant :**



Nous allons maintenant placer correctement le dégradé sur le rectangle pour donner notre impression d'illumination (Cf page suivante)

Sélectionnez l'outil de modification de dégradé	
Assurez-vous que le rectangle avec le dégradé soit sélectionné	
Sur le rectangle, cliquez sur la queue de la flèche symbolisant le dégradé et faite la tourner jusqu'à obtenir le résultat suivant	

Dernière modification avant de passer à la prochaine étape, nous devons modifier les propriétés de ce rectangle pour qu'il n'accepte pas les événements. En effet ce rectangle étant posé au dessus de tous les autres composants il captera en premier le clic de souris. Comme il ne définit aucun gestionnaire d'évènement il routera cet évènement à son parent (le Canvas **'Bouton'**). Ce faisant, les boutons **'Play'** et **'Pause'** ne recevront jamais le clic de souris. Nous allons donc masquer la zone de clic du rectangle en utilisant la propriété : **'IsHitTestVisible'**. Si cette propriété a la valeur **'False'** alors l'objet ne définit plus de zone de clic.

**Note :** Chaque élément possède une zone de clic qui définit la zone où un clic de souris déclenchera un évènement sur l'objet (comme **'MouseDown'** par exemple).

Assurez-vous que le rectangle contenant le dégradé soit bien sélectionné. Et dans le code XAML tapez ceci :

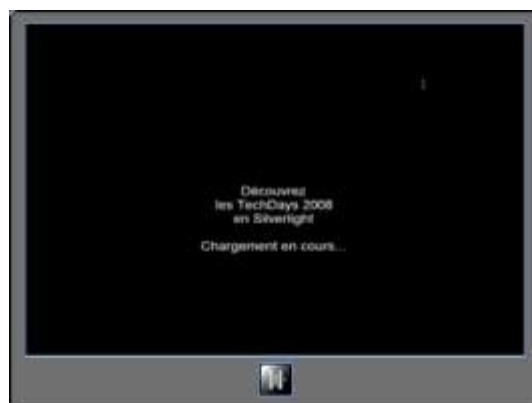
```

</Canvas>
<Rectangle IsHitTestVisible="False" Width="24" H
  <Rectangle Fill>

```

A présent notre rectangle ne répond plus aux événements de souris.

**Votre fenêtre doit ressembler à ceci :**



## Etape 20 : Tester le lecteur vidéo

Votre lecteur vidéo est à présent terminé vous pouvez le tester dans IE ! Il existe beaucoup d'autres fonctionnalités à ajouter à ce bouton, comme la possibilité d'arrêter la vidéo en cours de lecture (et non de la mettre en pause), d'ajouter une estimation du temps totale avec un curseur de lecture permettant de savoir où l'on se trouve etc... Une de ces fonctionnalités fait partie de l'exercice de la quatrième partie. Les autres fonctionnalités restent à faire, alors à vous souris et claviers !!!

## Quatrième partie : Pour aller plus loin...

Pour ceux qui souhaitent aller plus loin, voici un exercice à faire seul :

Ajoutez un bandeau gris semi-transparent à la fin de la vidéo avec un texte précisant de cliquer sur le bouton Play pour rejouer la vidéo. Faites apparaître ce bandeau progressivement par transparence. Voir la solution pour une idée de l'effet souhaité.

### Aide :

- L'objet **MediaElement** possède un événement '**MediaEnded**' envoyé quand le media lu arrive à la fin.
- Lorsqu'une vidéo se termine elle ne revient pas automatiquement au début, si vous appelez la méthode '**play()**' sur la vidéo rien ne se passera.
- Pour replacer une vidéo au début utilisez la propriété '**Position**' de l'objet **MediaElement** et donnez lui la valeur '**'00:00:00'**

Vous pouvez vous inspirer de la solution, pour voir comment cela peut-être implémenté.

## Conclusion

Ce Workshop est à présent terminé, j'espère que vous avez pris plaisir à le suivre, au moins autant que j'ai eu plaisir à le préparer. Rendez-vous dans un prochain épisode ☺