# Mở đầu

## Nội dung bài toán

## Đặc điểm bài toán

## Hướng giải quyết

# Giới thiệu về Windows Phone 7

## Phần cứng thiết bị

Trong phần này chúng ta sẽ tìm hiểu được về tiêu chuẩn phần cứng tối thiểu dành cho hệ điều hành WP7, từ đó chúng ta sẽ có thể nắm được những giới hạn về nội dung và hiệu ứng khi lập trình trong WP7.

**Nền tảng W7 như một máy tính**

Ngày nay thiết bị công nghệ thông tin thay đổi với tốc độ chóng mắt và các thiết bị cầm tay như cellphone cũng không nằm ngoại lệ. Hiện nay các cell phone không những phát triển về phần cứng mà phần mềm cũng thay đổi với tốc độ chóng mặt.

WP7 được coi như một chiếc vi tính thực thụ vì những khả năng không hề thua kém của nó. Một thiết bị WP7 có con chip xử lý tối thiểu khá mạnh, có khả năng lưu trữ cục bộ, hỗ trợ đồ họa 3D và bộ nhớ khá lớn với đơn vị lên tới hàng Gigabyte.

WP7 đã được viết lại từ nhân hệ điều hành mới và hoàn toàn khác với hệ điều hành windows mobile trước. WP7 viết dựa trên nhân chủ yếu từ môi trường Silverlight và XNA. Có thể nói hệ điều hành WP7 là nỗ lực lớn của Microsoft trong việc làm nên một hệ điều hành thực thụ dành cho cellphone hoạt động dựa trên chủ yếu vào màn hình cảm ứng.

**Bàn về nền tảng phần cứng của WP7**

Trước khi lập trình với WP7 chúng ta sẽ phải có cái nhìn sơ lược về phần cứng của nó. Tất cả thiết bị chạy hệ điều hành WP7 điều phải thỏa mãn cấu hình tối thiểu được đề xuất từ Microsoft. Vì vậy khi bạn cầm trên tay một chiếc WP7 bạn chắc chắn rằng sẽ tận hưởng được các hiệu ứng cũng như tốc độ xử lý ở mức độ chấp nhận được của WP7.

Thế nhưng bạn có thể dễ dàng thấy trên thị trường có các nhà sản xuất điện thoại cho ra các sản phẩm mạnh hơn đề xuất của Microsoft rất nhiều nhằm làm thỏa mãn các tín đồ về game, video cần các hiệu ứng đỉnh cao như bộ vi xử lý rất mạnh, màn hình cảm ứng lớn hơn, bộ nhớ nhiều hơn.

**Bộ vi xử lý**

The Central Processing Unit (CPU) là nơi mọi công việc điều được giải quyết tại đây. Hệ điều hành WP7 cần bộ vi xử lý vào khoảng 1GHz. So sánh với CPU thật ra không chỉ khác đơn thuần về tốc độ mà còn khác rất nhiều về cách thức hoạt động của chúng.

* thứ nhất: CPU trên PC có thể có đa nhân hỗ trợ chạy đa nhiệm, còn trên CPU WP7 thường chỉ có 1 nhân và đa nhiệm trên WP7 thường chỉ hỗ trợ ở mức tối thiểu nhằm đảm bảo thích hợp với bộ vi xử lý đi kèm.
* Thứ hai: CPU trên PC có tốc độ xử lý vượt trội với cellphone. Để đánh đổi với điều đó ta sẽ phải trả giá về nhiệt độ và lượng pin và phải cần một bộ nhớ RAM lớn.

Vì vậy khi lập trình với WP7 ta sẽ phải xem xét tới giới hạn của nó để có thể có những hướng phát triển tốt nhất.

**Hệ điều hành WP7**

hệ điều hành WP7 còn được gọi là Windows CE (“Compact Edition”), được thiết kê riêng cho thiết bị di động, tối ưu cho pin và tốc độ chip của thiết bị. Hệ điều hành WP7 dựa trên nhân của silverlight, một nhân tối ưu cho các hiển thị animations một cách mợt mà nhất với các hình ảnh vector không bị bể màu khi nằm trên các thiết bị có kích thước màn hình khác nhau. Vì vậy các chương trình chạy được trên WP7 hoàn toàn có thể chạy được trên PC.

**Đồ họa**

WP7 được thiết kế để hiển thị tốt các hình ảnh có chất lượng cao với mật độ pixel lớn. Trên WP7 chúng ta có thể thấy các dòng chữ được hiển thị với độ mịn cao. Thế nhưng để hiển thị càng nhiều pixel ảnh ta sẽ phải hao tổn bộ nhớ ram cũng như hao tốn pin hơn. WP7 biết sẽ làm cho màn hình hiển thị ảnh thích hợp nhất với màn hình và pin của cellphone. Vì vậy bạn có thể mở những file ảnh lớn mà không lo về chất lượng hiển thị cũng như sử ảnh hưởng tới thời lượng pin.

**Đa chạm**

Các thiết bị di động đời cũ có màn hinh cảm ứng thì thường là cảm ứng điện trở dựa vào lực tác động lên màn hình. Còn các thế hệ cellphone chạy WP7 thì thường dùng màn hình cảm ứng điện dung, nó hoạt động dựa trên các electron âm trên đầu ngón tay của người sử dụng. Vì vậy màn hình điện dung dễ dàng hỗ trợ đa chạm trên màn hình từ đó đem lại cho người dùng trải nghiệm hoàn toàn mới trên thiết bị di động.

**GPS ( Global Positioning System)**

Hầu hết các thiết bị chạy WP7 đều có hệ thống GPS nhằm giúp cho người dùng có thể xác định vị trí và tìm đường thông qua phần mềm hỗ trợ như GoogleMaps hoặc BingMaps....Có thể nói tính năng này dường như không thể thiếu trên các thiết bị di động thông minh hiện nay.

**Bộ gia tốc(Accelerometer)**

bộ gia tốc là một thiết bị phần cứng giúp hệ điều hành nhận biết độ rung,lắc hay hướng di chuyển của thiết bị. Từ thiết bị này xuất hiện rất nhiều chức năng được yêu thích như lắc đổi bài nhạc, chơi game đua xe trên iphone...

**Camera**

hầu hết các thiết bị cellphone hiện nay đều có camera và WP7 không nằm ngoại lệ. Các camera trên WP7 có ít nhất 5Mpx với những bức hình lên tới hơn 5 triệu điểm ảnh. Khi viết chương trình thao tác với camera chúng ta có vài điểm cần quan tâm:

* Chúng ta không thể can thiệp vào camera khi đang quay nếu không được hỗ trợ từ phía phần cứng camera. Tức ở đây bạn không thể chèn hình lên camera đang hoạt động được.
* Thứ 2: chúng ta không thể chụp một bức ảnh mà không có sự đồng ý từ phía người dùng.

**Bộ Nhớ Và Lưu trữ:**

Bộ nhớ là thành phần rất quan trọng trong bất kỳ cellphone nào. Bộ nhớ có 2 loại bộ nhớ “mass storage” dùng để lưu trữ chương trình và bộ nhớ Ram dùng để giúp các phần mềm lưu giữ trạng thái và truy xuất nhanh. Trên WP7 thì chúng ta có ít nhất 256MB bộ nhớ RAM và ít nhất 8GB ổ dữ liệu. Với bộ nhớ máy như trên nên WP7 sẽ tối ưu lại tất cả phần mềm sao cho phần mềm chạy mượt mà nhất có thể trên phần cứng bị giới hạn như vậy.

**Các kết nối mạng được hỗ trợ**

Kết nối internet là thành phần không thể thiếu của các điện thoại thông minh. Hệ điều hành WP7 hỗ trợ cho bạn các kết nối internet thông dụng nhất:

* Wifi: tất cả các thiết bị chạy WP7 đều hỗ trợ kết nối không dây. Nó cho bạn một kết nối tốc độ cao nhưng tốt nhất bạn nên làm việc gần với điểm phát sóng wifi để có một kết nối tốt nhất.
* 3G: có tốc độ truy cập gần như Wifi và có độ phủ sóng tốt hơn Wifi nên khắc phục được nhược điểm của wifi. Bạn có thể đi bất cứ đâu miễn có sóng di dộng bạn sẽ có thể truy cập Internet thông qua 3G.
* GPRS: nếu những nơi quá hẻo lánh không có Wifi hoặc 3G thì bạn có thể sài đến GPRS, thế nhưng tốc độ của nó không thể nào sánh bằng 2 kết nối trên.

Ngoài ra WP7 còn hỗ trợ cho chúng ta kết nối bluetooth để dành cho việc nghe nhạc qua tai nghe bluetooth hoặc gởi những lượng file nhỏ cho nhau.

**Các thách thức của WP7**

WP7 được thiết kế rất mạnh mẽ cho các thiết bị di động thế nhưng nó vẫn bị ràng buộc bởi các vấn đề về thời lượng pin, tốc độ xử lý và bộ nhớ Ram giới hạn. Vì vậy người lập trình viên phải chuẩn bị tốt kiến thức về giới hạn của thiết bị.

## Môi trường WP7

WP7 được Microsoft hỗ trợ rất nhiều với các phần mềm kèm theo để có thể giao tiếp tương tác trên các hệ thống khác nhau.

* Zune Software: Giúp WP7 giao tiếp với PC nhằm quản lý multimedia, trao đổi dữ liệu với nhau. Khi bạn lập trình trên Visual Studio bạn có thể đẩy ứng dụng của bạn lên thiết bị thông qua Zune được tích hợp trong Visual Studio.
* Windows Live và Xbox Live: Nếu bạn sở hữu WP7, bạn hoàn toàn có thể đăng ký tài khoản Live và tham gia các trò chơi trên Xbox Live của Microsoft. Bạn có thể làm việc với các ứng dụng văn phòng với office online được cung cấp bởi Microsoft thông qua tài khoản Live.
* Bing Maps: Bing Maps cung cấp cho bạn biết vị trí hiện tại cũng như bản đồ hầu hết các nước trên thế giới. Bạn có thể sử dụng được hầu hết các dịch vụ của Bing Maps như định vị, tìm kiếm vị trí, dẫn đường, bản đồ 3D...

## Hệ điều hành

## Phát triển phần mềm

## Tóm tắt

# Giới thiệu về Silverlight trên Windows Phone 7

## Lập trình giao diện với Siverlight

## Hiểu về XAML

## Tạo ứng dụng với Silverlight trên Windows Phone 7

## Tóm tắt

# Giới thiệu về XNA trên Windows Phone 7

## XNA là gì

## Tạo chương trình với XNA

## Sử dụng bộ gia tốc trong XNA

## Xử lý âm thanh trong XNA

## Làm việc với màn hình trong XNA

# Lập trình trên Windows Phone 7

## Các thành phần cơ bản trong WP7

### Ứng dụng đầu tiên với WP7

Mã code WP7 được viết trên nền .NET nên nó rất quen thuộc với lập trình viên .NET. Chúng ta có thể download Visual Studio Express For WP7 phiên bản miễn phí để có thể lập trình cho thiết bị WP7. Ngoài ra Microsoft còn cung cấp bộ công cụ Blend miễn phí để lập trình viên có thể dễ dàng thiết kế giao diện trên WP7.

Tóm lại ban đầu ta cần phải cài các bộ công cụ sau để có thể lập trình trên WP7:

* visual studio 2010 express for window phone

(<http://www.microsoft.com/express/Phone/> )

* Windows Phone Developer Tools

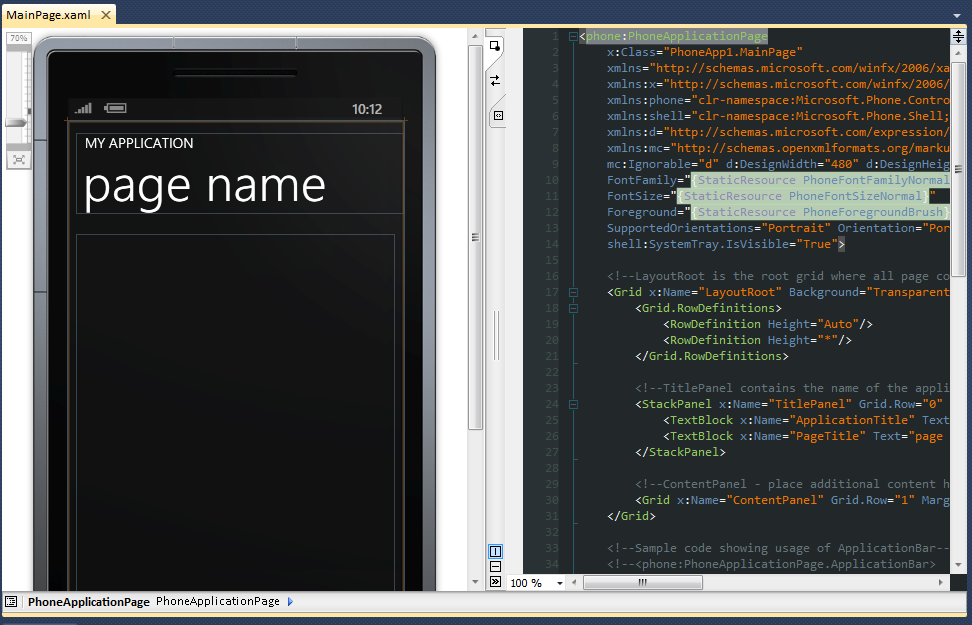
(<http://www.microsoft.com/downloads/en/details.aspx?FamilyID=49b9d0c5-6597-4313-912a-f0cca9c7d277> )

* Silverlight For WP7 Toolkit

(<http://silverlight.codeplex.com/> )

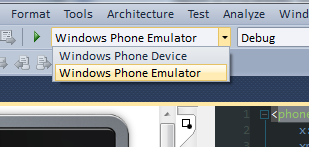
Bây giờ chúng ta đã có thể bắt đầu ứng dụng đầu tiên với WP7. Ban đầu trong template của visual studio 2010 bây giờ đã có template “Silverlight For Windows Phone” của ứng dụng WP7.

Khi khởi tạo project xong, ta sẽ thấy xuất hiện hình ảnh WP7 như sau:



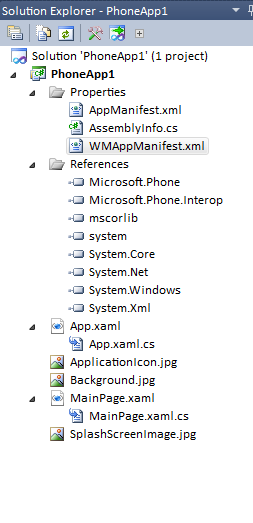
Hình 1: Khởi tạo project

Trên màn hình design trên ta có thể kéo các control từ Toolbox bar để có thiết kế giao diện một cách dễ dàng. Điều quan trọng ta cần lưu ý là khi bạn lập trình cho WP7 bạn có thể chạy chương trình của bạn trên WP7 emulator hoặc trên thiết bị thật thông qua dây cắm USB. Để cấu hình khi chạy ứng dụng bạn có thể chọn như hình sau:



Hình 2: Chọn chế độ chạy phần mềm.

Để có thể hiểu phần nào cấu trúc phần mềm ta sẽ nhin vào cây thư mục của phần mềm:



Hình 3: Cấu trúc project WP7

Ta sẽ tìm hiểu cấu trúc cũng như nhiệm vụ của các thành phần trong project như sau:

File App.xaml, App.xaml.cs và MainPage.xaml, MainPage.xaml.cs: Nếu là lập trình viên .NET ta hoàn toàn có thể biết ngay rằng các file có đuôi cs là những file “code-behind” còn những file có đuôi xaml là những file Extensible Application Markup Language (XAML) tương tác với các file cs tương ứng.

Mở file App.xaml.cs ta sẽ cấu trúc quen thuộc như sau:

namespace SilverlightHelloPhone

{

public partial class App : Application

{

public App()

{

…

InitializeComponent();

…

}

}

}

Ở đây ta cần chú ý các class được đặt là “partial” vì Microsoft đã chuẩn bị cho nó khả năng kêu thêm một file cs mới có các thành viên của class App này.

Cấu trúc file App.xaml như sau:

<Application

x:Class="PhoneApp1.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>

File App.xaml thường được sử dụng để lưu trữ các resource của toàn bộ ứng dụng. Resource của ứng dụng thường bao gồm giá trị của các biến màu, giá trị các brush hoặc các style và animation.

Đúng cú pháp của ngôn ngữ XAML trong Silverlight bạn phải có element root và sau đó phải khai báo 4 XML Namespace.

Property “xmlns” là namespace tiêu chuẩn rất phổ biến trong Silverlight và property xmlns đầu tiên giúp cho compiler định nghĩa được class khi thực thi.

Propery XML thứ 2 thì tham chiếu tới những element và đặc tính khác đã được trong nhân Silverlight.

Trong project WP7 ta có thể gặp thường xuyên 2 property tương tự nhau là x:name và name. Thực ra x:name được tham chiếu từ:

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml".

Còn name property tham chiếu từ:

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

Thực ra không hề có sự khác biệt nhau ở 2 property này cho lắm. Nhưng khi bạn dùng để viết template silverlight cho visual studio thì bạn phải bắt buộc sử dụng x:name để tránh các lỗi về compile cũng như lỗi hiển thị. (1)

Khi chương trình bắt đầu chạy, từ App Class sẽ tạo ra object PhoneApplicationFrame. PhoneApplicationFrame sử lý như một web browser và sẽ gọi class MainPage.

MainPage.xaml và MainPage.xaml.cs là thành phần bắt buộc trong Silverlight. Đây là trang chủ của chương trình tức khi chương trình được chạy thành công sẽ đến trang này. Xét qua MainPage.xaml.cs class ta sẽ thấy cấu trúc của nó rất đơn giản:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Animation;

using System.Windows.Shapes;

using Microsoft.Phone.Controls;

namespace PhoneApp1

{

public partial class MainPage : PhoneApplicationPage

{

// Constructor

public MainPage()

{

InitializeComponent();

}

}

}

Việc kết thừa class PhoneApplicationPage mang lại cho class MainPage kế thừa đầu đủ các tính năng của một trang ứng dụng WP7. Phần giao diện sẽ do file MainPage.xaml quyết định. Mở file MainPage.xaml ta sẽ thấy cấu trúc như sau:

<phone:PhoneApplicationPage

x:Class="PhoneApp1.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="\*"/>

</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="MY APPLICATION" Style="{StaticResource PhoneTextNormalStyle}"/>

<TextBlock x:Name="PageTitle" Text="page name" Margin="9,-7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>

</StackPanel>

<!--ContentPanel - place additional content here-->

<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>

</Grid>

</phone:PhoneApplicationPage>

Thoạt nhìn qua ta sẽ nhìn thấy 4 khai báo xml đầu giống với file App.xaml. Còn lại có một số keyworld ta cần nắm là namespace “d” là viết tắt cho chữ “design” và “mc” là viết tắt của “markup compatibility”(2) và 2 từ khóa trên là thế mạnh của ngôn ngữ XAML. Xét về cơ bản XAML cũng là ngôn ngữ giống như HTML thế nhưng XAML lại được Microsoft trang bị cho những khả năng đồ họa tuyệt vời với cú pháp chặt chẽ hơn.

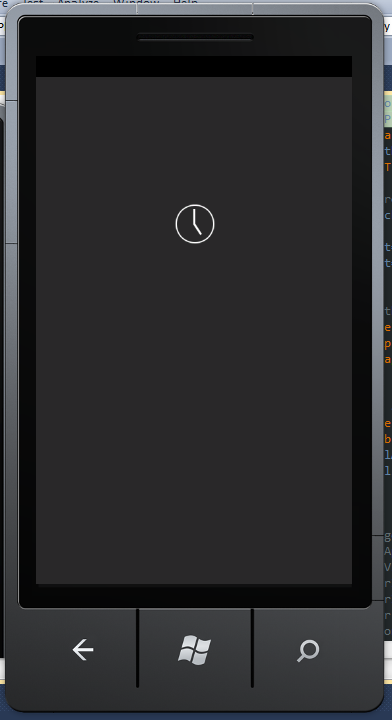
Ta có thể nhìn thấy đoạn mã XAML trên đã xét các property FontSize, FontFamily và Fontground thông qua một StaticResource. Ta sẽ nghiên cứu kĩ hơn về StaticResource trong chương sau nhưng qua đây bạn có thể nào hình dung được khả năng quản lý resource rất tốt của Silverlight For WP7.

Phần thiết kế trang thông qua các control Grid, StackPanel... các control này sẽ được tìm hiểu kỹ càng hơn ở các chương sau.

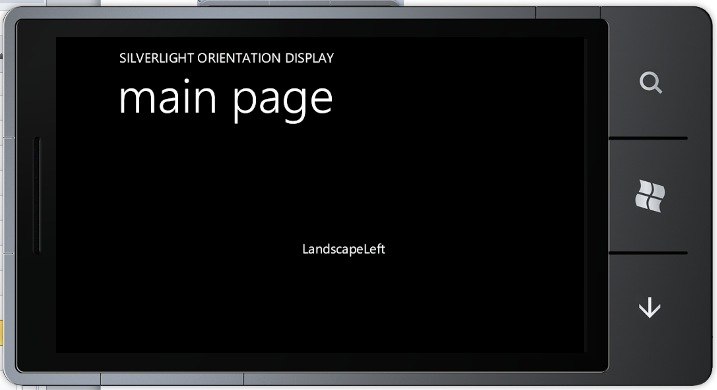
Phần cuối cùng trong chương này, chúng ta sẽ phải nắm rõ rằng khi bạn build project WP7 thì file thực thi chứa tất cả mã lệnh và resource của chương trình sẽ được nằm trong file có đuôi là XAP và file XAP đã được tối ưu hóa để làm sao có thể chạy nhanh nhất khi chạy. Thực ra file XAP chỉ tương tự như file ZIP mà thôi. Ta hoàn toàn có thể giải nén nó như một file ZIP bình thường. Ta có thể tìm thấy file XAP trong thư mục Debug/Bin. Khi bạn đẩy ứng dụng lên thiết bị thật, ta chỉ cần deploy file XAP là đủ.

### Sự điều hướng trong WP7

Mặc định khi khởi tạo project WP7 thì mặc định chương trình được thiết kế theo chiều thẳng đứng (chế độ Portrait ). Thế nhưng ta có thể dễ dàng thay đổi giao diện cho phù khi người dùng để máy nằm ngang (chế độ Landscape).



Hình 4: Chế độ Portrait



Hình 5: Chế độ Landscape.

Mặc định thì project sẽ không hỗ trợ tự động chuyển đổi giao diện khi bạn thay đổi hướng sử dụng thiết bị từ Portrait sang Landscape. Vì vậy bạn cần phải cấu hình cho thiết bị tự động chuyển đổi giao diện bằng cách sau. Rất đơn giản, bạn tìm trong file xaml giao diện của trang WP7, tìm property

SupportedOrientations="Portrait"

Sau đó thay thế bằng:

SupportedOrientations="PortraitOrLandscape"

Sau đó bạn sẽ có thể thấy rằng giao diện người dùng sẽ tự động hỗ trợ người dùng một cách tự động giữa 2 chế độ Portrait và Landscape.

### Giới thiệu về Touch Event (3)

Hiện nay các dòng điện thoại thông minh phần lớn đều hoạt được dựa trên màn hình cảm ứng đa chạm có kích thước lớn nhỏ khác nhau. Vì vậy các nhân hệ điều hành thiết bị di động thường hỗ trợ tối đa việc xử lý trên các màn hình cảm ứng đa chạm. WP7 cũng không nằm ngoài số đó. Đối với các ứng dụng bình thường, có thể ta chưa thể hình dung hay so sánh khả năng cảm ứng mà hệ điều hành hỗ trợ. Thế nhưng khi ta chơi game cần tới những động tác phức tạp và cần độ nhạy cao, ta sẽ có thể cảm nhận được khả năng xử lý của một hệ điều hành có tốt hay không.

Trong chương này, ta sẽ làm quen với các sự kiện cảm ứng trên thiết bị WP7. Ta sẽ có thể hình dung một cách dễ dàng thông qua các hình vẽ sau:

* Cảm ứng đơn chạm:

Tap Touch and Hold

Double Tap Pan

Flick



* Cảm ứng đa chạm:

Pinch and Stretch



Sau đây ta sẽ có thể bắt đầu ứng dụng làm quen với việc xử lý sự kiện cảm ứng trên WP7. Để giúp chúng ta làm việc dễ dàng với màn hình cảm ứng, Microsoft đã cung cấp cho ta class GestureService . Thông qua nó ta se dễ dàng bắt được sự kiện và xử lý chúng một cách dễ dàng.

Chú ý: class GestureSerive được cung cấp thông qua bộ toolkit WP7 chứ không phải ở trong bộ Windows Phone 7 Developer.

Class GestureService được sử dụng bằng cách một GestureListener  vào element cần xử lý. Ban đầu ta cần add file dll *Microsoft.Phone.Controls.Toolkit.dll* vào project (ta có thể tìm theo đường dẫn sau “C:\Program Files (x86)\Microsoft SDKs\Windows Phone \v7.0 \Toolkit \Nov10\Bin\Microsoft.Phone.Controls.Toolkit.dll”).

Sau đó ta cần thêm namespace trong file xaml cần xử lý như sau:

*xmlns:toolkit="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"*

Ta sẽ viết một ví dụ đơn giản như sau. Ban đầu ta sẽ có một hình chữ nhật màu cam. Nếu người dùng Tap trên màn hình thì hình chữ nhật thành màu trắng, còn nếu người sử dụng Hold trên màn hình thì hình chữ nhật thành màu đỏ.



Hình 6: Ví dụ về cảm ứng chạm trên WP7.

Ban đầu ở file giao diện ta cần thêm đoạn code sau:

<Rectangle Fill="Orange" x:Name="rect">

    <toolkit:GestureService.GestureListener>

        <toolkit:GestureListener  Tap="GestureListener\_Tap" Hold="GestureListener\_Hold"  />

    </toolkit:GestureService.GestureListener>

</Rectangle>

Sau đó là code-behind:

private void GestureListener\_Tap(object sender, GestureEventArgs e)

{

    this.rect.Fill = new SolidColorBrush(Colors.White);

}

private void GestureListener\_Hold(object sender, GestureEventArgs e)

{

    this.rect.Fill = new SolidColorBrush(Colors.Red);

}

Qua ví dụ trên ta có thể thấy được sự đơn giản khi ta muốn bắt sự kiện trong WP7 và qua đó ta thấy được sự rõ ràng trong cấu trúc một project WP7.

### Bitmap hay Textures trên WP7 (4)

Hệ điều hành WP7 hỗ trợ cho ta 3 loại định dạng chính:

* JPEG (Joint Photography Experts Group)
* PNG (Portable Network Graphics)
* GIF (Graphics Interchange File)

Để làm quen với việc sử dụng image trong WP7, ta sẽ phải làm quen với thẻ image trong xaml và một số khái niệm build một image trong WP7. Có hai loại build chinh cho một image trong project WP7

* Build loại Content: tức khi này file ảnh sẽ được include vào file XAP nhưng nằm ở ngoài file dll.
* Build loại Resource: tức khi này file ảnh sẽ được nhúng vào trong file DLL.

Vì cách build hoàn toàn khác nhau như vậy nên ta cũng sẽ có các thao tác khác nhau đối với từng loại build. Với build Content ta sẽ làm như sau:

File XAML :

<Image Stretch="None" Source="/images/appbar.cancel.rest.png"/>

File CS :

//Content

Uri uri = new Uri("/images/appbar.cancel.rest.png", UriKind.Relative);

BitmapImage imgSource = new BitmapImage(uri);

this.image.Source = imgSource;

Còn đối với build là Resource ta sẽ làm như sau:

File XAML:

<Image Source="/WP7SampleProject3;component/images/appbar.feature.email.rest.png"/>

File CS:

//Resource

Uri uriR = new Uri("/WP7SampleProject3;component/images/appbar.feature.email.rest.png", UriKind.Relative);

BitmapImage imgSourceR = new BitmapImage(uriR);

this.imageR.Source = imgSourceR;

Có thể ban đầu ta hơi thắc mắc về Relative hoặc Absolute UriKind. Ta có thể phân biệt như sau:

* Relative UriKind: Thường được dùng để chỉ ra vị trí các file thuộc ứng dụng của mình.
* Absolute UriKind: Thường được dùng để chỉ ra vị trí các file cần được download từ nguồn ngoài ứng dụng. VD: <http://www.contoso.com/resources/OnDemand.png>.
* RelativeOrAbsolute: Khi ta không chắc chắn về link UriKind ta sẽ dùng biến này để WP7 tự động thiết lập cho phù hợp.

### Đôi nét về kiến trúc ứng dụng

Project WP7 có các class cơ bản sau đây:

* App class
* Một đối tượng của PhoneApplicationFrame
* Một đối tượng của PhoneApplicationPage

Một project WP7 chỉ có 1 class App duy nhất để chứa các giá trị toàn cục. Sau đó ta cần một đối tượng PhoneApplicationFrame để cung cấp các framework cũng như các hàm cơ bản cho ứng dụng. Sau đó ta có thể có nhiều đối tượng PhoneApplicationPage như là một trang của trang web và ta có thể di chuyển qua lại giữa chúng.

Trong chương này ta sẽ làm quen với kiến trúc của WP7 cũng như các kĩ thuật cơ bản để làm việc với WP7.

#### Sự chuyển trang (Basic Navigation)

Sự chuyển trang trong WP7 có thể nói là sự sáng tạo táo bạo trong việc thiết kế giao diện cũng như cách quản lý của WP7. Thực chất của sự chuyển trang trong WP7 là một chuỗi trang được ghi lại và được quản lý bởi một class NavigationService một cách linh hoạt. Sau đây là một đoạn mã đơn giản dành cho việc chuyển sang trang ứng dụng khác.

this.NavigationService.Navigate(new Uri("/SecondPage.xaml", UriKind.Relative));

Tham số đầu vào là một Uri class với địa chỉ trang muốn đến, ở đây ví dụ là “SecondPage.xaml”. Còn property Relative ta đã đề cập ở chương trên.

Tương tự ta có thể quay lại trang trước cũng bằng cách trên, ta chỉ cần đưa địa chỉ trang trước vào. Thế nhưng ta có cách làm đơn giản hơn nhiều như sau nếu ta muốn quay lại trang trước:

this.NavigationService.GoBack();

Điều này sẽ làm cho chương trình không phải khai báo thêm class PhoneApplicationPage vào trong stack chứa trang, hơn nữa trang được GoBack() sẽ được lưu trạng thái khi nó bị chuyển trang.

#### Chuyển đổi dữ liệu qua các trang

Về vấn đề này thường có 2 câu hỏi đặt ra:

* Làm sao ta có thể chuyển dữ liệu sang trang tiếp theo ?
* Làm sao ta có thể trả dữ liệu về trang trước?

Trong phần này ta sẽ tìm hiểu cách hoạt động của WP7 về vấn đề này. Ban đầu ta sẽ sét vấn đề chuyển dữ liệu sang trang kế tiếp. Thực ra rất quen thuộc với người lập trình web, giống như phương thức GET ta từ trang ban đầu sẽ kêu trang tiếp sau bằng dòng lệnh như ví dụ sau:

this.NavigationService.Navigate(new Uri("/SecondPage.xaml? "?Red=32&Green=255&Blue=255", UriKind.Relative));

Phía bên trang nhận dữ liệu sẽ thực hiện đoạn code sau để lấy dữ liệu:

IDictionary<string, string> parameters = this.NavigationContext.QueryString;

if (parameters.ContainsKey("Red"))

{

}

byte R = Byte.Parse(parameters["Red"]);

byte G = Byte.Parse(parameters["Green"]);

byte B = Byte.Parse(parameters["Blue"]);

ContentPanel.Background = new SolidColorBrush(Color.FromArgb(255, R, G, B));

Nhìn vào đoạn code trên ta thấy để có thể lấy được dữ liệu ta sẽ phải thông qua class NavigationContext với hàm Query. Hàm Query sẽ đưa các giá trị vào các biến tương ứng.

#### Sự chia sẽ dữ liệu giữa các trang

Để có thể chia sẽ dữ liệu giữa nhiều trang khác nhau, ta không thể dùng cách trên được. Vì vậy Microsoft đã cung cấp cho chúng ta một cách là dùng đối tượng App để dùng làm nơi lưu trữ dữ liệu chung.

Đơn giản như sau, ta viết trong file App.xaml.cs một ví dụ sau:

public partial class App : Application

{

// public property for sharing data among pages

public Color? SharedColor { set; get; }

...

}

Ta sẽ định nghĩa một property có thể null là Sharedcolor và biến này sẽ lưu trữ giữ liệu như một vùng nhớ chung cho tất cả các trang. Vì vậy khi ta đứng ở bất cứ trang nào mà muốn lấy dữ liệu hay đặt dữ liệu vào, ta chỉ cần thực hiện đoạn mã sau:

if (ContentPanel.Background is SolidColorBrush)

(Application.Current as App).SharedColor = (ContentPanel.Background as SolidColorBrush).Color;

if (sharedColor != null)

ContentPanel.Background = new SolidColorBrush(sharedColor.Value);

Ta thấy rằng khi đứng ở bất cứ mọi trang, ta đều có đối tượng App class được tạo sẵn và tồn tại song song với chương trình. App class lưu giữ tất cả các giá trị được sử dụng cho cả chương trình.

#### Sự lưu giữ dữ liệu thông qua đối tượng:

Ở đây ta xét trường hợp ta muốn lưu giữ dữ liệu, trạng thái của một trang giao diện WP7. Khi ở phiên làm việc đầu tiên ta đã có được một số thông tin dữ liệu, ta muốn khi ta quay lại trang này, tất cả thông tin trạng thái sẽ được lưu trữ để ta có thể tiết kiệm thời gian thao tác lại các thao tác ta đã làm việc trước đó.

Để giải quyết vấn đề này ta thường suy nghĩ ngay đến việc lưu ngay dữ liệu có được và để sài lúc sau. Để làm việc này ta có thể lưu trên “isolated storage”. “isolated storage” được thiết kế giống như ổ cứng vật lý trong PC và nó có thể chứa được số lượng dữ liệu rất lớn. Nó khá phức tạp và ta sẽ bàn đến ở chương sau.

Ở đây ta sẽ bàn đến class PhoneApplicationService, nó được định nghĩa trong Microsoft.Phone.Shell namespace. Nó được khai báo trong XAML như sau:

<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>

Qua đoạn mã trên ta thấy tiếp theo sau thẻ PhoneApplicationService là 4 property đi theo sau. Đoạn mã trên sẽ tạo cho chúng ta một đối tượng của class PhoneApplicationService, sau đó ta sẽ một đối tượng static thông qua PhoneApplicationService.Current property.

Một PhoneApplicationService class sẽ chứa một property là State thuộc loại IDictionary<string, object>. Ta sẽ lưu giữ đối tượng thông qua một string key. Ta chú ý ở đây, dữ liệu này chỉ tồn tại khi chương trình đang chạy mà thôi. Nếu thoát ra khỏi chương trình thi dữ liệu sẽ được xóa đi. Bất kỳ object nào trong State đều phải có khả năng serializable tức là có khả năng chuyển object đó vòa XML.

Chúng ta sẽ tham khảo một đoạn ví dụ sau đây để xem cách lấy dữ liệu của class PhoneApplicationService:

protected override void OnNavigatedFrom(NavigationEventArgs args)

{

if (ContentPanel.Background is SolidColorBrush)

{

Color clr = (ContentPanel.Background as SolidColorBrush).Color;

if (args.Content is MainPage)

(args.Content as MainPage).ReturnedColor = clr;

// Save color

PhoneApplicationService.Current.State["Color"] = clr;

base.OnNavigatedFrom(args);

}

}

protected override void OnNavigatedTo(NavigationEventArgs args)

{

// Retrieve color

if (PhoneApplicationService.Current.State.ContainsKey("Color"))

{

Color clr = (Color)PhoneApplicationService.Current.State["Color"];

ContentPanel.Background = new SolidColorBrush(clr);

}

base.OnNavigatedTo(args);

}

Ta sẽ chú ý đến đoạn mã

PhoneApplicationService.Current.State["Color"] = clr;

Ta dễ dàng đặt dữ liệu vào một State một cách đơn giản và sau đó khi ta cần dùng tới chỉ cần lấy dữ liệu bằng đoạn mã sau:

Color clr = (Color)PhoneApplicationService.Current.State["Color"];

#### Isolated Storage

Đây là thành phần rất quang trọng trọng WP7. Isolated Storage được coi như là nơi lưu trữ tất cả giữ liệu của cả thiết bị WP7. Để truy xuất dữ liệu ta sẽ dùng System.IO.IsolatedStorage namespace để lấy dữ liệu từ bộ nhớ WP7.

Không phải tình cờ mà trong file App.xaml lại có khai báo một đoạn mã như sau:

<shell:PhoneApplicationService Launching="Application\_Launching"

Closing="Application\_Closing"

Activated="Application\_Activated"

Deactivated="Application\_Deactivated"/>

Ở đây Microsoft cung cấp cho chúng ta một nơi để load dữ liệu cho các biến toàn cục từ Isolated Storage. Ví dụ ta muốn load các tham số về ngôn ngữ chương trình hay theme của chương ngay ngay lúc khởi tạo. Ta sẽ dựa vào các sự kiện trên để làm việc với Isolated Storage:

private void Application\_Launching(object sender, LaunchingEventArgs e)

{

LoadSettings();

}

private void Application\_Activated(object sender, ActivatedEventArgs e)

{

LoadSettings();

}

private void Application\_Deactivated(object sender, DeactivatedEventArgs e)

{

SaveSettings();

}

private void Application\_Closing(object sender, ClosingEventArgs e)

{

SaveSettings();

}

Ta thực hiện các hàm LoadSettings() và SaveSettings() như sau:

void LoadSettings()

{

IsolatedStorageSettings settings = IsolatedStorageSettings.ApplicationSettings;

Color clr;

if (settings.TryGetValue<Color>("backgroundColor", out clr))

BackgroundBrush = new SolidColorBrush(clr);

}

void SaveSettings()

{

IsolatedStorageSettings settings = IsolatedStorageSettings.ApplicationSettings;

if (BackgroundBrush is SolidColorBrush)

{

settings["backgroundColor"] = (BackgroundBrush as SolidColorBrush).Color;

settings.Save();

}

}

Thao tác với IsolatedStorage gần giống với lưu dữ liệu với State nhưng ta nên nhớ rằng thời gian sống của chúng hoàn toàn khác nhau. Vì vậy ta phải chọn lựa nên dùng phương án nào cho thích hợp.

## Ngôn ngữ XAML

Ta đã được biết hệ điều hành WP7 được viết trên nhân của Siverlight. Vì vậy trong chương này ta sẽ điểm qua cơ bản về ngôn ngữ XAML cũng như thế mạnh và giới hạn của nó. Đối với những người đã từng lập trình html thì có lẽ XAML đã khá quen thuộc. XAML cũng là ngôn ngữ đánh dấu giống như HTML thế nhưng nó lại được Microsoft trang bị cho các khả năng tuyệt vời mà HTML không có.Trong XAML sẽ chịu trách nhiệm tất cả về phần UI bao gồm các control, animation, màu sắc, binding dữ liệu.... Có thể nói XMAL và code-behind C# là nền tảng của WP7.

Để tìm hiểu XAML ta sẽ có ví dụ đơn giản như sau:

<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">

<TextBlock Text="Hello, Windows Phone 7!"

HorizontalAlignment="Center"

VerticalAlignment="Center" />

</Grid>

Element trong XAML thực ra cũng là một class mà thôi. Trên code-behind ta cũng có class TextBox như trên XAML. Thực ra chúng chỉ là một mà thôi và đó cũng chính là điểm mạnh khi lập trình WP7. Ta có thể thấy phần giao diện và mã xữ lý được tách riêng hẳn ra nhưng vẫn rất gắn kết với nhau. Các propertys của TextBox trên XAML cũng chính là propertys của class. Vì vậy ta có thể dễ dàng chỉnh sửa nó trên code-behind một cách dễ dàng khi runtime.

Nếu bạn không cần để mã XAML phần giao diện, ta có thể khởi tạo lúc runtime một cách đơn giản như sau:

TextBlock newTextBlock = new TextBlock();

newTextBlock.Text = "Hello, Windows Phone 7!";

newTextBlock.HorizontalAlignment = HorizontalAlignment.Left;

newTextBlock.VerticalAlignment = VerticalAlignment.Top;

newTextBlock.Margin = new Thickness(

(ContentPanel.ActualWidth - txtblk.ActualWidth) \* rand.NextDouble(),

(ContentPanel.ActualHeight - txtblk.ActualHeight) \* rand.NextDouble(),

0, 0);

ContentPanel.Children.Add(newTextBlock);

ở đây ta không đề cập đến element Grid. Đây là thành phần khá quan trọng giúp cho việc thiết kế giao diện trong WP7, vì vậy ta sẽ dành ra một chương cho việc nói về vần các thẻ layout này.

### Kế thừa property:

Ở đây, giả sử ta vào trường hợp nhiều element khác nhau muốn dùng các property giống nhau. Khi đó ta sẽ phải để property đó thành Property Inheritance. Khi đó các element nào trong XAML muốn dùng nó sẽ gọi đến Property Inheritance mà thôi.

Ở đây ta chỉ cần thêm một attribute là một instacne của Style class. Ví dụ ở đây ta thêm một định nghĩa FontStyle vào trong đối tượng PhoneApplicationPage theo cú pháp sau:

<phone:PhoneApplicationPage ... FontStyle="Italic" ...

Sau đó ta chỉ cần set fontstyle cho đối tượng cần dùng. Ví dụ:

<TextBlock Text="Hello, Windows Phone 7!"

FontStyle="Italic" />

### Colors and Brushes(5)

Color trong WP7 được viết theo một chuỗi red, green, blue với mỗi giá trị theo đơn vị hex. Giá trị hex của Color bắt đầu từ 00 đến FF. Ví dụ ta set font màu của chữ của một đối tượng bất kỳ ta có thể viết:

Foreground="#FF0000" //red color

Một đối tượng Color khi khởi tạo sẽ có giá trị là #000000. Ngoài Color, Microsoft còn cung cấp cho chúng ta là Brushes class. Burshes class phức tạp hơn rất nhiều so với Color class. Brushes hỗ trợ ta tô màu theo radian, vector.... Ta có các loại Brush cơ bản sau:

#### SolidColorBrush:

Là cách tô màu đơn sắc không có hiệu ứng. Ta sẽ có ví dụ sau:

XAML:

<Ellipse Stroke="Black" StrokeThickness="3"

Width="64" Height="64" Canvas.Left="8" Canvas.Top="8">

<Ellipse.Fill>

<SolidColorBrush Color="Navy" />

</Ellipse.Fill>

</Ellipse>

Kết quả:



#### LinearGradientBrush:

Là cách tô màu hiệu ứng sáng chung được trộn từ 2 màu từ 2 phía của element. Ví dụ:

XAML:

<Ellipse Stroke="Black" StrokeThickness="3"

Width="64" Height="64" Canvas.Left="8" Canvas.Top="8">

<Ellipse.Fill>

<LinearGradientBrush>

<GradientStop Color="Navy" Offset="0" />

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Ellipse.Fill>

</Ellipse>

Kết quả:



#### RadialGradientBrush

Là cách tô màu tương tự như LinearGradientBrush thế nhưng lại không tô màu theo hướng từ 2 phía mà từ theo tâm điểm. Ta có ví dụ:

XAML:

<Ellipse Width="75" Height="75" Stroke="Black">

<Ellipse.Fill>

<RadialGradientBrush>

<GradientStop Color="Black" Offset="0"/>

<GradientStop Color="Black" Offset="1"/>

<GradientStop Color="Gray" Offset="0.5"/>

</RadialGradientBrush>

</Ellipse.Fill>

</Ellipse>

Kết quả:



#### ImageBrush

Là cách tô màu bằng cách chọn một hình ảnh làm background. Ta có ví dụ:

XAML:

<Ellipse Width="60" Height="60" Stroke="Black">

<Ellipse.Fill>

<ImageBrush ImageSource="http://www.silverlightinaction.com/man.png" />

</Ellipse.Fill>

</Ellipse>

Kết quả:



#### VideoBrush:

VideoBrush tương tự như ImageBrush thế nhưng ta có thể có một background động với video được chỉ định trước.

XAML:

<MediaElement x:Name="myMediaElement" Opacity="0"

Source="http://www.silverlightinaction.com/video2.wmv" />

<Ellipse Width="100" Height="100" Stroke="Black">

<Ellipse.Fill>

<VideoBrush SourceName="myMediaElement" />

</Ellipse.Fill>

</Ellipse>

Kết quả:



### Style and Theme:

Có thể nói Style là một tập các thuộc tính đã được ta định nghĩa sẵn để áp dụng cho đối tượng đã được nhắm tới. Vì vậy một style phải có property là TargetType để chỉ loại đối tượng có thể áp dụng Style này.

<Style x:Key="txtblkStyle"

TargetType="TextBlock">

...

</Style>

Ví dụ như đoạn mã trên thì style txtblkStyle chỉ có thể áp dụng cho TextBlock mà thôi.

Cách viết Style cũng khá đơn giản, ta chỉ cần đặt tên thuộc tính cần gán giá trị vào thẻ Property và giá trị muốn gán vào thẻ Value. Ta có thể ví dụ như sau:

<Style x:Key="txtblkStyle" TargetType="TextBlock">

<Setter Property="HorizontalAlignment" Value="Center" />

<Setter Property="VerticalAlignment" Value="Center" />

<Setter Property="Margin" Value="12 96" />

<Setter Property="FontSize" Value="48" />

</Style>

Phức tạp hơn ta có thể cấu hình cho một thuộc tính ngay trong Style như sau:

<Style x:Key="txtblkStyle"

TargetType="TextBlock">

<Setter Property="HorizontalAlignment" Value="Center" />

<Setter Property="VerticalAlignment" Value="Center" />

<Setter Property="Margin" Value="12 96" />

<Setter Property="FontSize" Value="48" />

<Setter Property="Foreground">

<Setter.Value>

<LinearGradientBrush>

<GradientStop Offset="0" Color="Pink" />

<GradientStop Offset="1" Color="SkyBlue" />

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Style>

Sau đó để áp dụng style này cho một textbox, ta sẽ phải gán property Style của textbox vào style mà chúng ta tạo ra ở trên.

<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">

<TextBlock Text="Whadayasay?"

Style="{StaticResource txtblkStyle}"

HorizontalAlignment="Left"

VerticalAlignment="Top" />

<TextBlock Text="Fuhgedaboudit!"

Style="{StaticResource txtblkStyle}"

HorizontalAlignment="Right"

VerticalAlignment="Bottom" />

</Grid>

Ở đây có vấn đề xảy ra là nếu TextBox đã định nghĩa property mà trong Style cũng có rồi thì sao? Ở đây WP7 sẽ ưu tiên lấy giá trị của property được định nghĩa trong element TextBox, vì độ ưu tiên của Style thấp hơn của Textbox. Thứ tự ưu tiên được sắp xếp như sau:

* Local Setting
* Style Setting
* Property Inheritance
* Defaulse Value.

Theme trong WP7 là một tập các Style được WP7 định nghĩa sẵn trong hệ thống. Khi lập trình với WP7. Ta dễ dàng thấy các đoạn mã tương tự như sau:

FontFamily="{StaticResource PhoneFontFamilyNormal}"

FontSize="{StaticResource PhoneFontSizeNormal}"

Foreground="{StaticResource PhoneForegroundBrush}"

Ta sẽ thấy các Style PhoneFontFamilyNormal, PhoneFontSizeNormal... đã được định nghĩa trước sẵn trong hệ thống. Ta chỉ cần chỉnh sữa lại các Style lại cho phù hợp với phần mềm của mình.

## Các Control trong WP7

### Basic Sharp

Như ta đã biết WP7 có đồ họa theo vector đồ họa, vì vậy ta có thể có những hình ảnh chất lượng khi màn hình phóng to. Khi thao tác với các hình vẽ, ta sẽ phải thông qua System.Windows.Sharp namespace. Có hai class chính trong namespace này, đó là Ellipse và Rectagle.

Ví dụ:

Ta có đoạn mã như sau:

<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">

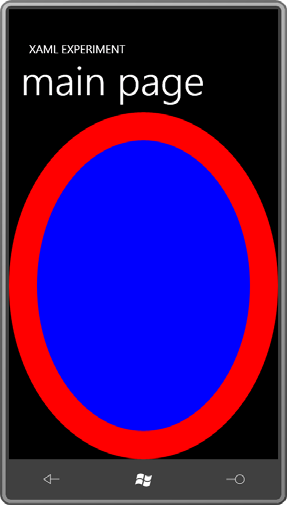
<Ellipse Fill="Blue"

</Grid>

Stroke="Red"

StrokeThickness="50" />

Khi đó ta sẽ có hình vẽ sau:



Tương tự tương ứng với đoạn mã:

<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">

<Ellipse Width="300"

Height="300">

<Ellipse.Fill>

<RadialGradientBrush Center="0.4 0.4"

GradientOrigin="0.4 0.4">

<GradientStop Offset="0" Color="White" />

<GradientStop Offset="1" Color="Red" />

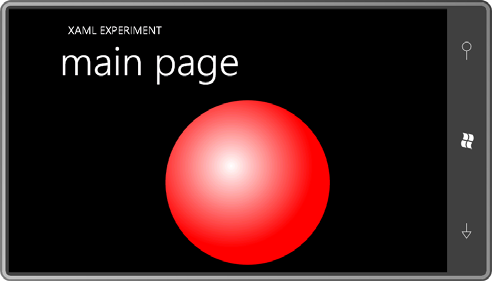
</RadialGradientBrush>

</Ellipse.Fill>

</Ellipse>

</Grid>

Ta cũng sẽ có kết quả sau:



### Animation:

Animation là sự di chuyển của nhiều đối tượng hình ảnh. Animation trong WP7 hoạt động chủ yếu dựa trên các Key-Frame và Storyboard để thực hiện được các animation. Một Storyboard sẽ quản lý tất cả các animation con với một timeline và chỉ rõ vị trí các nguồn tài nguyên dùng cho các animation con. Còn Key-Frame sẽ chịu trách nhiệm quản lý các đối tượng tham gia vào các animation dựa vào thuật toán nội suy(6).

### Border Element

Các control như TextBox hay ListBox không được trang bị những tính năng tự tạo đường viền bao xung quanh nó. Để làm điều này, ta sẽ phải dùng đến element Border để bao xung quanh control mà ta muốn tạo đường viền. Border element sẽ có property Child là một đối tượng của class UIElement. Có nghĩa bạn sẽ đặt element nào ta muốn làm viền vào property này.

Ví dụ sau ta sẽ làm đường viền cho một Textbox.

<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">

<Border Background="Navy"

BorderBrush="Blue"

BorderThickness="16"

CornerRadius="25">

<Border.Child>

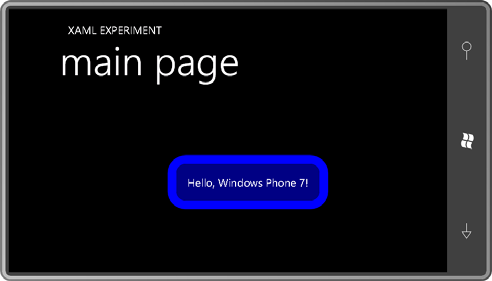
<TextBlock Text="Hello, Windows Phone 7!" />

</Border.Child>

</Border>

</Grid>

Công việc còn lại ta chỉ cần chỉnh sửa lại Border element cho phù hợp mà thồi.



### TextBlock Element:

TextBlock thực ra cũng khá giống như TextBox, thế nhưng nhiệm vụ của mỗi control lại khá khác nhau. TextBox element giành cho việc nhập liệu, còn TextBlock dành cho việc trình diễn thông tin. Một TextBlock có 5 property liên quan đến font: FontFamily, FontSize, FontStretch, FontStyle, and FontWeight.

Ngoài ra TextBlock còn có TextDecorations property dùng để trang trí cho TextBlock.

TextDecorations="Underline"

Đoạn mã trên sẽ cho ta một block có chữ gạch chân. Ngoài ra ta còn có một số property khác TextWrapping dùng để tự canh xuống hàng văn bản và hơn hết nó còn hỗ trợ nhận dạng thẻ. Ta có thể xem ví dụ sau đây:

<TextBlock FontSize="36"

TextWrapping="Wrap">

This is

some <Run FontWeight="Bold">bold</Run> text and

some <Run FontStyle="Italic">italic</Run> text and

some <Run Foreground="Red">red</Run> text and

some <Run TextDecorations="Underline">underlined</Run> text

and some <Run FontWeight="Bold"

FontStyle="Italic"

Foreground="Cyan"

FontSize="72"

TextDecorations="Underline">big</Run> text.

</TextBlock>

Ta thấy ta chưa hề định nghĩa thẻ Run nên đây có thể là một lỗi. Thế nhưng trình biên dịch của WP7 vẫn chạy được vì TextBlock vẫn chấp nhận và tự sửa chữa cho phù hợp. Ta sẽ có kết quả sau:



### Image Element

Hiện nay WP7 chỉ hỗ trợ 2 loại ảnh là JPEG và PNG.

## Layout

## AppBar và Control trong WP7

## DataBinding

## Vector Graphics

## Raster Graphics

## Animations

## Template

## Items Controls

## Pivot and Panorama

# Framework MVVM (Model-View-ViewModel) lập trình trên Windows Phone 7

(<http://msdn.microsoft.com/en-us/magazine/dd419663.aspx>)

## Lịch sử ra đời và quá trình phát triển MVVM

## Tại sao dùng MVVM

## Ứng dụng Demo

# Đánh giá và kết luận

## Tóm tắt

## Thử nghiệm

## Đánh giá

## Hướng phát triển

Phụ Lục:

* Các thành phần cơ bản website du lịch
* Các chức năng chính của WP7 Travel

{1} <http://stackoverflow.com/questions/589874/in-wpf-what-are-the-differences-between-the-xname-and-name-attributes>

{2} Trang 14 sách Microsoft press Ebook Programing Windows Phone 7

{3} <http://www.windowsphonegeek.com/articles/WP7-GestureService-in-depth--key-concepts-and-API>

{4} <http://www.windowsphonegeek.com/tips/wp7-working-with-images-content-vs-resource-build-action>

{5} Trang 526 sách Mainning Silverlight 4 in Action.

{6}Microsoft Vietnam LLC – DPE team | Chuong VI: Media và Animation