Movies in Finnkino Theatre's

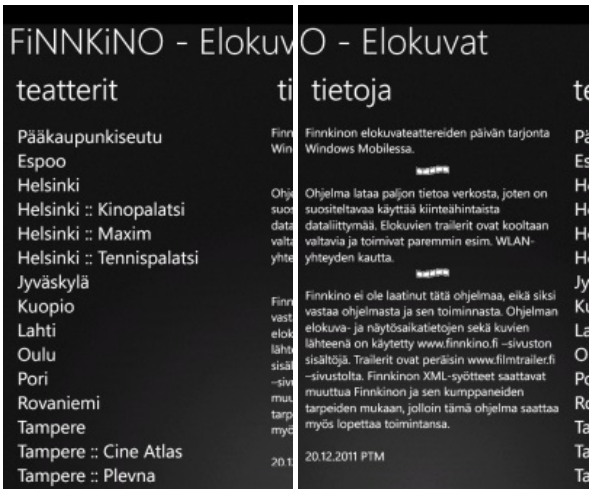
This article explains how to load a lot of XML based data - movies in Finnkino Theatre's - and display Theatre's schedules and Movies data in Windows Phone.

29 Jan 2012

Introduction

In this article, I will show how to create Windows Phone application which loads Movie's information from Finnkino's XML service. Loaded Theatre's and Movie's data will be parsed and displayed in a different Pages in Windows Phone application.

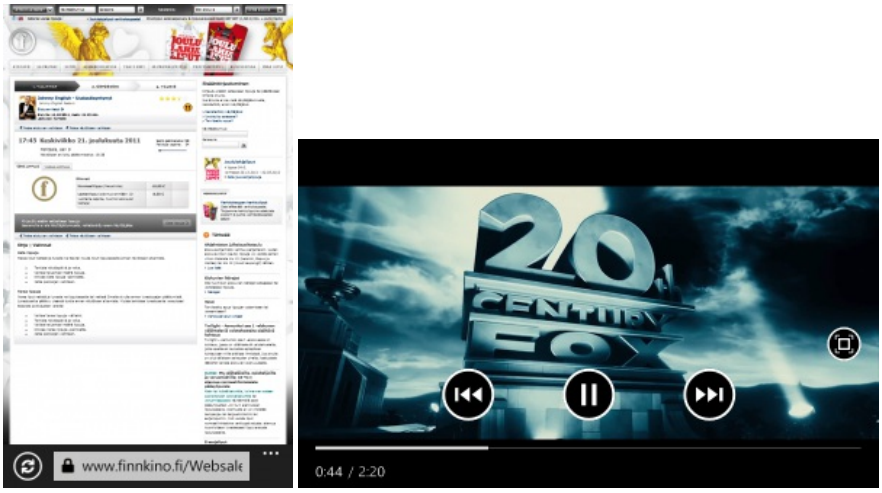
Application is designed to run in Panorama Type in WP. First application connects to Finnkino's XML service and loads all the currently showing events (one XML file). After that it loads all the event's information and schedules of the Theatres (based on the Areas in Finland - many XML files). In Panorama view all the Finnish Theatre's (cities) are displayed in a ListBox Control (Image 1). There are some information about this application displayed in second page of Panorama (sorry this application is targeted to Finland and it only displays Finnish texts).



Todays events will be displayed when user selects Theatre from the Main List (Image 3). User can scroll up and down to see which movies are running today in the selected Theatre. Selected movie details are shown in a new movie detail page (Image 4). Showed data are combined from the Finnkino's XML data and it contains: theatre and movie name, starting time, length, genre, year, and synopsis text.

User can click "Osta Liput" Button to buy tickets to this selected movie. It launches Windows Phone's default browser and opens Finnkino's web site to this specific movie page (Image 5). User can watch the movie trailer if it is available in XML data. "Katso Video" Button launches Windows Phone's default movie application to load and display movie trailer in fullscreen.

(C) Copyright Nokia 2013. All rights reserved.



This applications is now available to download on Windows Marketplace: Finnkino Elokuvat

In this article, I will also write a few lines about testing application in Microsoft Visual Studio.

Note! Finnkino is not the author of this program, and is therefore not responsible for the program and its activities.

Demo Video - Application in Action

The media player is loading...

Finnkino's XML data

Finnkino offers nice XML data set from available Theatres and movie events in Finland (look more details here Finnkino - XML).

Following XML sources are used in this example:

- List of Theatre Areas http://www.finnkino.fi/xml/TheatreAreas
- Movie Schedules for a date http://www.finnkino.fi/xml/Schedule
- Theatre Area Schedules List (now 17 different Areas) http://www.finnkino.fi/xml/Schedule/?area=1015 (for example Jyväskylä)
- List of currently showing Events http://www.finnkino.fi/xml/Events

Like you can see there is a lot of different data's available in those XML feeds and we are going to load all of them!

Note! Finnkino can/may change the structure of XML data at anytime and you might need to make modifications to this example.

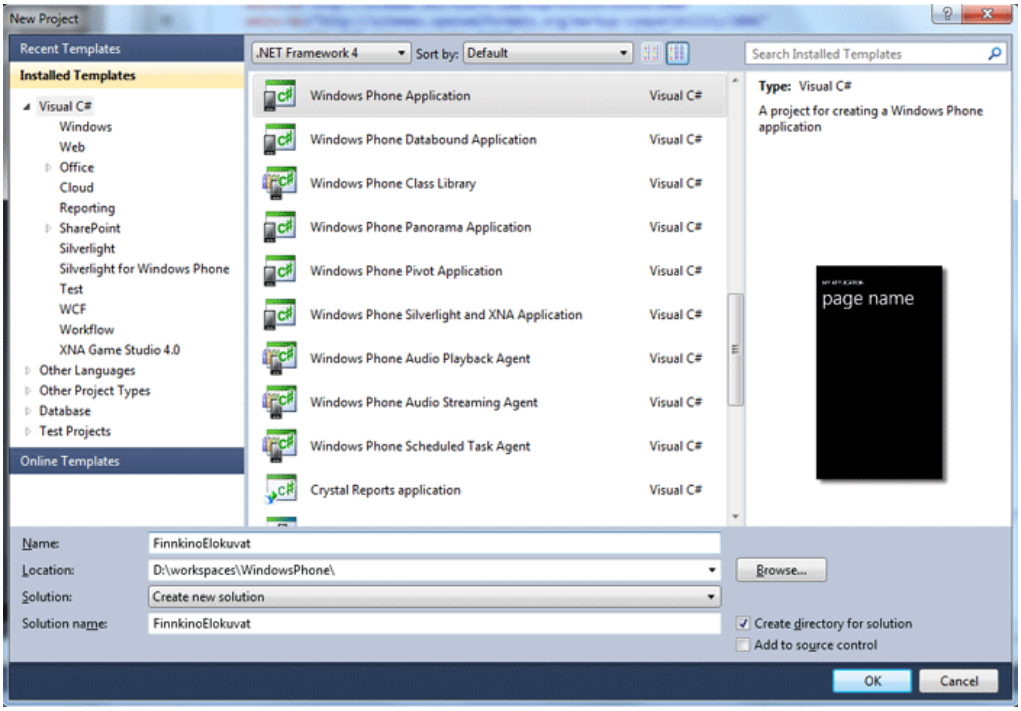
Windows Phone 7.1 SDK

To Develop application for Windows Phone 7 Devices, you need to install Windows Phone 7.1 SDK. You can download latest SDK for Windows Phone here

Windows Phone Application

To start creating a new Windows Phone Application, start Microsoft Visual studio then create a new Project and select Windows Phone Application Template. There is a Panorama Template also available but select Windows Phone Application Template. In this way we keep our application as clean as possible and we can do later those panoramas with coding.

(C) Copyright Nokia 2013. All rights reserved.



In this example I have used C# as code behind language.

XML Parsing in Windows Phone

There are many different ways to load and parse XML data in Windows Phone . In this example I will use XML Deserialization to load XML document and pass the data to a deserializer. You will need to add references to `System.Xml.Serialization` and `System.Xml.Linq` to your project. Right click "References" in your project Solutions Explorer and select "Add new Reference..."

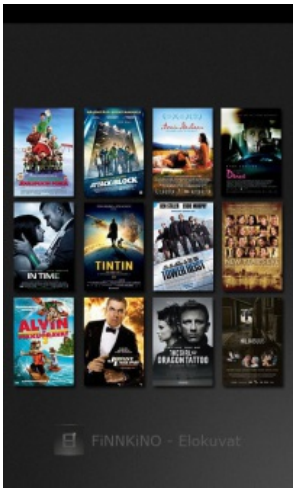
Loading and displaying many images from net

There can be a small issues when you try to load lot of images from the web at the same time in Windows Phone. You can find a few assemblys from the net to fix this. One good one is PhonePerformance assembly (you can download compiled PhonePerformance assembly with source codes here). You have to download and unzip it. Copy PhonePerformance.dll to your project and add a new reference to it with Solution Explorer.

You might get warning with to unblock PhonePerformance.dll assembly (because it is downloaded from web). Close your Visual Studio. Open Windows Explorer and browse your PhonePerformance.dll file. Open file properties and Unblock.

Splash Screen Image

You can find default SplashScreenImage.jpg image in your project folder. Easiest way to add your own customized Splash Screen images is just to edit this image in your favorite image edition application tool. Image has to be 400x480 size and it's Build Actions has to be set to Content so it winds up in the XAP file at deployment time.



The above image is shows as a splash screen in Finnkino Elokuvat application.

Classes behind the Application

(C) Copyright Nokia 2013. All rights reserved.

To create a New Classes in to your project:

1. Right click your project in Solutions Explorer
2. Select Add and then Class...

Events.cs

List of currently showing events are available at http://www.finnkino.fi/xml/Events/. I only use `EventID`, `Synopsis` and `Video` url in this example. `EventID` and `Synopsis` are used as a Strings. `Video` url is `List<XElement>` collection because there can be a video Url or not. So, it will be parsed later in the code (to activate "Katso Video" Button in screen).

```
using System.Collections.Generic;
using System.Xml.Linq;

namespace FinnkinoElokuvat
{
    public class Event
    {
        public string EventID { get; set; }
        public string Synopsis { get; set; }
        public List<XElement> Video { get; set; }
    }
}
```

Show.cs

Every Theatre movie is described as a `Show` node in XML data (look here : http://www.finnkino.fi/xml/Schedule/?area=1015). There are a lot of information available but only those what are listed in `Show` Class are used in this example. Some of the images are handled same way as the Video in earlier.

```
using System.Xml.Linq;
using System.Collections.Generic;

namespace FinnkinoElokuvat
{
    public class Show
    {
        public string ID { get; set; }
        public string EventID { get; set; }
        public string Title { get; set; }
        public string EventSmallImagePortrait { get; set; }
        public string EventLargeImagePortrait { get; set; }
        public string EventSmallImageLandscape { get; set; }
        public List<XElement> Image { get; set; }
        public string ShowStart { get; set; }
        public string LengthInMinutes { get; set; }
        public string Genres { get; set; }
        public string ProductionYear { get; set; }
        public string ShowURL { get; set; }
        public string TheatreAndAuditorium { get; set; }
    }
}
```

TheatreArea.cs

`TheatreArea` Class is clean and simply. It only stores theatre `id` and `name`.

```
namespace FinnkinoElokuvat
{
    public class TheatreArea
    {
        public string id { get; set; }
        public string name { get; set; }
    }
}
```

Panorama View

Design (MainPage.xaml)

Finnkino Elokuvat application uses two Pages in Panorama View. First page displays XML loading process and all the available Theatre's (areas) in Finland when the XML loading process is done. The second page is used to display some (legal and credits) information of the application.

The Grid of the Phone Application Page contains Panorama Control element (`controls:Panorama`) which contains Panorama Title, Background and two Panorama Items.

```
<controls:Panorama Title="FiNNKiNO - Elokuvat">
  <!-- controls:Panorama.TitleTemplate -->
  <!-- controls:Panorama.Background -->
  <!-- controls:PanoramaItem Header="teatterit" -->
  <!-- controls:PanoramaItem Header="tietoja" -->
</controls:Panorama>
```

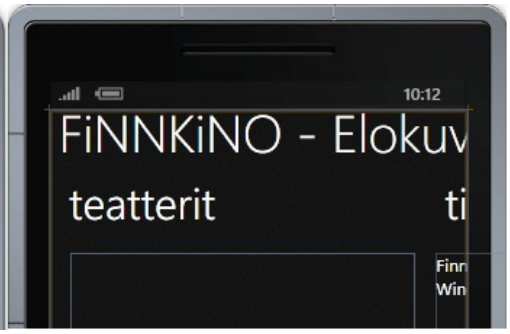
By default Panorama's Page and Title are shown using a quite big font size. You can change this by using Panorama's `TitleTemplate` Element

```
<controls:Panorama.TitleTemplate>
 <DataTemplate>
```


(C) Copyright Nokia 2013. All rights reserved.

```
  <TextBlock Text="{Binding Content, RelativeSource={RelativeSource TemplatedParent}}"
     Foreground="White"
     FontSize="60"
     Margin="0,60,0,0"/>
 </DataTemplate>
</controls:Panorama.TitleTemplate>
```

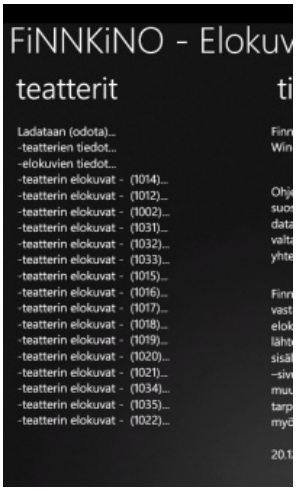
I have used font size 60 (which seems nice size to me in this application) and text's foreground is set the white because I will use black gradient image as a background.

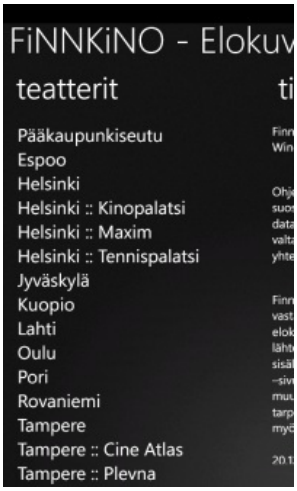


You can set your own background image to the Panorama View. Create Images folder to your Project (in Solution Explorer) and copy your image to this folder in Windows. Select Add, Existing Item.. and browse your image. In this example I have used 1024x800 sized image as a background of the Panorama.

```
<controls:Panorama.Background>
 <ImageBrush ImageSource="Images/bg.png"/>
</controls:Panorama.Background>
```

First view of the Panorama displays loading process and Theatre (area) names when loading process is finished.





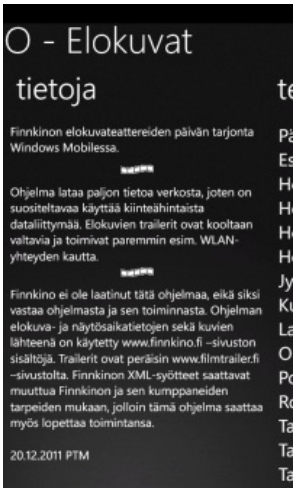
By default the header is displayed with too big font size (for this example) and it is modified same way as Panorama Title earlier using with `PanoramaItem.HeaderTemplate` Element. Actually the first panorama item is a grid with `TextBlock` (to display loading process) and the `ListBox` of the Theatre names (areas) in Finland. `TextBlock` will be removed from the Grid after the loading process is done.

```
<controls:PanoramaItem Header="teatterit">
 <controls:PanoramaItem.HeaderTemplate>
  <DataTemplate>
   <TextBlock Text="{Binding Content, RelativeSource={RelativeSource TemplatedParent}}"
       FontSize="50"
       Foreground="White"
       Margin="0,0,0,0"/>
  </DataTemplate>
 </controls:PanoramaItem.HeaderTemplate>

 <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
  <TextBlock x:Name="InfoTextBlock" Foreground="White" />
  <ListBox x:Name="TheatreListBox" FontSize="30" Foreground="White"/>
 </Grid>
</controls:PanoramaItem>
```

The second view of the Panorama shows informations of the application. Header is modified like earlien in this example. PanoramaItem contains few `TextBlocks` and `Images`.

(C) Copyright Nokia 2013. All rights reserved.




```
<controls:PanoramaItem Header="tietoja">
<controls:PanoramaItem.HeaderTemplate>
 <DataTemplate>
  <TextBlock Text="{Binding Content, RelativeSource={RelativeSource TemplatedParent}}"
      FontSize="50" Foreground="White"
      Margin="0,0,0,0"/>
 </DataTemplate>
</controls:PanoramaItem.HeaderTemplate>
<Grid>
 <TextBlock TextWrapping="Wrap"
      Text="Finnkinon elokuvateattereiden päivän tarjonta Windows Mobilessa."
      Foreground="White"
      />
 <Image Source="Images/film.png"
     Width="54"
     Height="15"
     VerticalAlignment="Top"
     Margin="0,70"/>
 <TextBlock Margin="0,100"
      Foreground="White"
      Text="Ohjelma lataa paljon tietoa verkosta... " TextWrapping="Wrap"/>
 <Image Source="Images/film.png"
     Width="54"
     Height="15"
     VerticalAlignment="Top"
     Margin="0,240"/>
 <TextBlock Margin="0,269,0,38"
      TextWrapping="Wrap"
      Foreground="White"
      Text="Finnkino ei ole laatinut tätä ohjelmaa... " Height="297" />
</Grid>
</controls:PanoramaItem>
```

Programming (MainPage.xaml.cs)

All the XML loading process is happening here in the `MainPage.xml.cs` file. First when I created this example I stored all the data here in this `MainPage.xaml.cs` file but later I noticed that the data is better store in to `Appl.xaml.cs` file because it is easier to share data between Pages and handle Tombstone situation. So, look more information about `theatreAreas`, `theatreEvents` and `theatreShows` Lists from `App.xaml.cs` file later in this article. I also have one `dataLoaded` variable in `App.xaml.cs` which is set to `true` when all the data is loaded for a first time.

This application handles a lot of dynamic data. In Windows Phone your application can be passed to background for example when a new phone call is coming or user pressed the windows button in the device and runs a different application. If you are lucky your application can keep its data in the memory (your application is in Dormant mode) and your application might work right when it is running again in the device (going foreground). But Windows Phone might send your application to Tombstoned mode also and then you have to store your application data and load it back when your application is running again. I will discuss more about this memory handling later in this article.

In MainPage Class I have used a following variables

```
int showsLoaded = 0;           // how many theatre's informations (shows) are loaded
App app = App.Current as App;  // reference to application instance (there are all the List Collections for XML data)
```

After splash screen or when user comes back from movie details page `OnNavigatedTo` method will be called. Here we fist check is data already loaded or not. If we are starting the application then data is not loaded yet and we first check is there a network connection available in `GetIsWorking()` method.

```
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
 // XML is not loaded yet, check if phone is connected to network
 if (!app.dataLoaded) GetIsWorking();
 // data is loaded, show theatres in listbox
 else
 {
          // this method will be described later in this example
          AddTheatres();
 }
}
```

Check is there network connection available:

```
public void GetIsWorking()
{
 // is there network connection available
```


(C) Copyright Nokia 2013. All rights reserved.

```
 if (!System.Net.NetworkInformation.NetworkInterface.GetIsNetworkAvailable())
 {
  MessageBox.Show("Ei verkkoyhteyttä");
  InfoTextBlock.Text = "Elokuvateattereiden tietoja ei saada ladattua.\nSovellus vaatii verkkoyhteyden toimiakseen.\n\nTarkista matk
  return;
 }

 // network is available, start loading XML data
 InfoTextBlock.Text = "Ladataan (odota)...\n-teatterien tiedot...";
 GetTheatreAreas();
}
```

If network connection is available, we start to load Theatre Area information. In Windows Phone you can use `WebClient` Class to load data asynchronously. `TheatresDownloaded` method will be called when the data is loaded or if there are errors occurred.

```
public void GetTheatreAreas()
{
 WebClient downloader = new WebClient();
 Uri uri = new Uri("http://www.finnkino.fi/xml/TheatreAreas/", UriKind.Absolute);
 downloader.DownloadStringCompleted += new DownloadStringCompletedEventHandler(TheatresDownloaded);
 downloader.DownloadStringAsync(uri);
}
```

In `TheatresDownloaded` method we first check if there are errors occured and display some information with `MessageBox` Control. If there are no errors, all the Theatres ID's and names will parsed from loaded XML data. `TheatreArea` objects will be created from XML data and those objects are stored to `List<TheatreArea> theatreAreas` Collection (in `App.xaml.cs` file).

```
public void TheatresDownloaded(object sender, DownloadStringCompletedEventArgs e)
{
 if (e.Result == null || e.Error != null)
 {
  MessageBox.Show("Virhe Finnkinon XML-tietojen latauksessa!");
 }
 else
 {
  // parse XML data
  XDocument document = XDocument.Parse(e.Result);
  // create TheatreArea objects from loaded XML data
  var data = from query in document.Descendants("TheatreArea")
       select new TheatreArea
       {
        id = (string)query.Element("ID"),
        name = (string)query.Element("Name")
       };
  // store all TheatreArea objects to List (in App.xaml.cs file)
  app.theatreAreas = data.ToList<TheatreArea>();
  // we don't use the first element at TheatreAreas data
  app.theatreAreas.RemoveAt(0);
  // start loading events
  GetEvents();
 }
}
```

All the Movie Events will be loaded next with `WebClient` Class.

```
public void GetEvents()
{
 InfoTextBlock.Text += "\n-elokuvien tiedot...";
 WebClient downloader = new WebClient();
 Uri uri = new Uri("http://www.finnkino.fi/xml/Events/", UriKind.Absolute);
 downloader.DownloadStringCompleted += new DownloadStringCompletedEventHandler(EventsDownloaded);
 downloader.DownloadStringAsync(uri);
}
```

`EventsDownloaded` method will be called when the Events data is loaded or if there are errors occurred. If there are no errors, Events `ID`'s, `Synopsis` and `Video` Url will parsed from loaded XML data. First the Event objects will be created from XML data and those objects are stored to `List<Event> theatreEvents` Collection (in `App.xaml.cs` file).

```
public void EventsDownloaded(object sender, DownloadStringCompletedEventArgs e)
{
 if (e.Result == null || e.Error != null)
 {
  MessageBox.Show("Virhe Finnkinon XML-tietojen latauksessa!");
 }
 else
 {
  // parse XML data
  XDocument document = XDocument.Parse(e.Result);
  // create Event objects from loaded XML data
  var data = from query in document.Descendants("Event")
       select new Event
       {
        EventID = (string)query.Element("ID"),
        Synopsis = (string)query.Element("Synopsis"),
        Video = query.Descendants("Videos").Descendants("EventVideo").ToList<XElement>()
       };
  // store all Event objects to List (in App.xaml.cs file)
  app.theatreEvents = data.ToList<Event>();
  // start loading schedules
  GetSchedule();
 }
}
```

All the Theatre Schedules will be loaded next with `WebClient` Class. This class will be called as many times as there are Theatres available.

```
public void GetSchedule()
{
 // get the theatre id
 string area = app.theatreAreas[showsLoaded].id;
 InfoTextBlock.Text += "\n-teatterin elokuvat -  (" + area + ")...";
```


(C) Copyright Nokia 2013. All rights reserved.

```
 WebClient downloader = new WebClient();
 // start loading this theatre schedule
 Uri uri = new Uri("http://www.finnkino.fi/xml/Schedule/?area=" + area, UriKind.Absolute);
 downloader.DownloadStringCompleted += new DownloadStringCompletedEventHandler(ScheduleDownloaded);
 downloader.DownloadStringAsync(uri);
}
```

`ScheduleDownloaded` method will be called when the Schedule data is loaded or if there are errors occurred. If there are no errors, `Show` objects will be created from XML data and those objects are stored to `List<List<Show>> theatreShows` Collection (in `App.xaml.cs` file).

There are not always images included to shows in XML data and this causes a small problems. We have to first query Images descendants (and create `Show` objects) and later check if there are image or not. In this example I will add different size No*.png images to `Show` objects if image node is empty in XML data. You have to add your own images to project.

```
void ScheduleDownloaded(object sender, DownloadStringCompletedEventArgs e)
{
 if (e.Result == null || e.Error != null)
 {
  MessageBox.Show("Virhe Finnkinon XML-tietojen latauksessa!");
 }
 else
 {
  // parse XML data
  XDocument document = XDocument.Parse(e.Result);
  // create Show objects from loaded XML data
  var data = from query in document.Descendants("Show")
       select new Show
       {
        ID = (string)query.Element("ID"),
        EventID = (string)query.Element("EventID"),
        Title = (string)query.Element("Title"),
        ShowStart = (string)query.Element("dttmShowStart"),
        LengthInMinutes = (string)query.Element("LengthInMinutes"),
        Genres = (string)query.Element("Genres"),
        ProductionYear = (string)query.Element("ProductionYear"),
        ShowURL = (string)query.Element("ShowURL"),
        TheatreAndAuditorium = (string)query.Element("TheatreAndAuditorium"),
        Image = query.Descendants("Images").ToList<XElement>()
       };
  // create a new list (for this theatre shows)
  List<Show> shows = new List<Show>();
  shows = data.ToList<Show>();

  // check if there are images in XML - shows - (add No*.png if not)
  for (int i = 0; i < shows.Count; i++)
  {
   // is there EventSmallImagePortrait image
   try
   {
    shows[i].EventSmallImagePortrait = (string) shows[i].Image.Descendants("EventSmallImagePortrait").ElementAt(0);
   }
   catch (ArgumentOutOfRangeException)
   {
    shows[i].EventSmallImagePortrait = "Images/NoEventSmallImagePortrait.png";
   }

   // is there EventSmallImageLandscape image
   try
   {
    shows[i].EventSmallImageLandscape = (string) shows[i].Image.Descendants("EventSmallImageLandscape").ElementAt(0);
   }
   catch (ArgumentOutOfRangeException)
   {
    shows[i].EventSmallImageLandscape = "Images/NoEventSmallImageLandscape.png";
   }

   // is there EventLargeImagePortrait image
   try
   {
    shows[i].EventLargeImagePortrait = (string) shows[i].Image.Descendants("EventLargeImagePortrait").ElementAt(0);
   }
   catch (ArgumentOutOfRangeException)
   {
    shows[i].EventLargeImagePortrait = "Images/NoEventLargeImagePortrait.png";
   }
  }
  // store all Show objects to List (in App.xaml.cs file)
  app.theatreShows.Add(shows);
  // show is loaded
  showsLoaded++;
  // get another theatre shows (if there are any left)
  if (showsLoaded < app.theatreAreas.Count()) GetSchedule();
  // all shows are loaded, create menu of Theatres
  else if (showsLoaded == app.theatreAreas.Count())
  {
   app.dataLoaded = true;
   AddTheatres();
  }
 }
}
```

Now all the massive XML data is loaded and it is time to create main menu to the user. In this example and this Page I just add Theatre names to the `ListBox` at the all time when application comes back to this Page.

```
public void AddTheatres()
{
 // remove info textblock (displays loading information)
 ContentPanel.Children.Remove(InfoTextBlock);
 // remove previous names from the list
 TheatreListBox.Items.Clear();
 // create listbox items from theatreAreas
 for (int i = 0; i < app.theatreAreas.Count(); i++)
 {
  TheatreListBox.Items.Add(app.theatreAreas[i].name);
 }
 // add eventlistener to listbox
 TheatreListBox.SelectionChanged += new SelectionChangedEventHandler(TheatreListBox_SelectionChanged);
}
```


(C) Copyright Nokia 2013. All rights reserved.

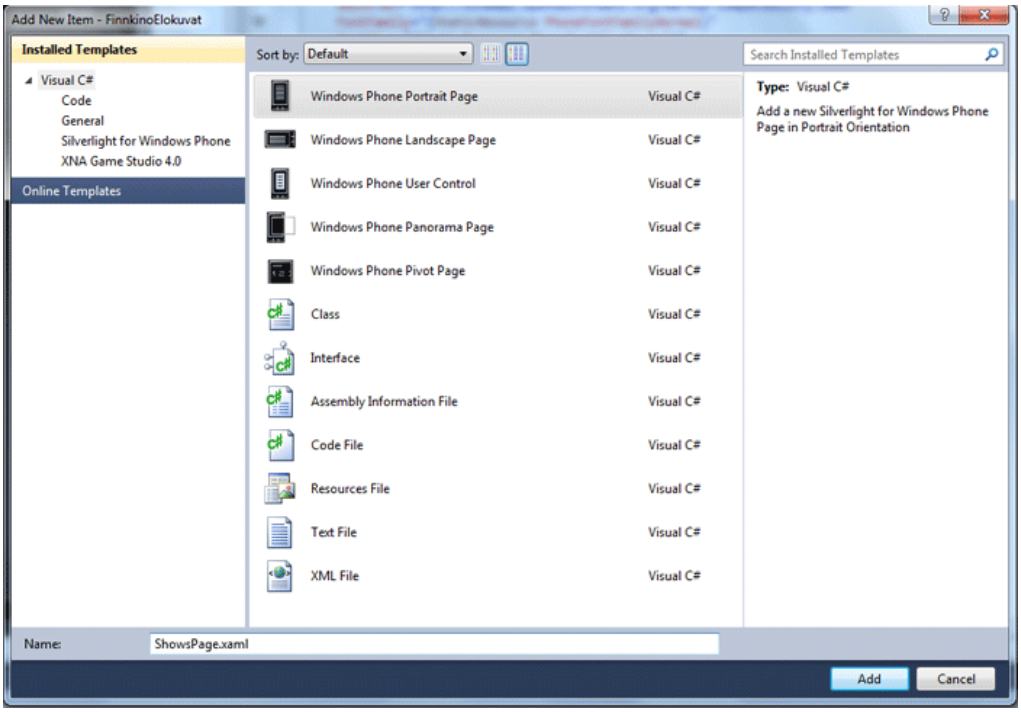
A new Shows Page will be displayed when the user selects the Theatre from the `TheatreListBox`. All the data is stored to `App` Class, so we need to pass the Theatre's index to the new page. We are clearing the `SelectionChanged` method from the list (it will be created again when the list is shown again) and finally we are opening a new page with sending one parameter to it with `GET`-method (this is one way to share data between Pages in Windows Phone Applications).

```
void TheatreListBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
 // get the selected theatre index
 int selectedIndex = (sender as ListBox).SelectedIndex;
 // remove listener from the listbox
 TheatreListBox.SelectionChanged -= new SelectionChangedEventHandler(TheatreListBox_SelectionChanged);
 // open a shows page with selected index
 this.NavigationService.Navigate(new Uri("/ShowsPage.xaml?SelectedIndex="+selectedIndex, UriKind.Relative));
}
```

Shows Page

Design (ShowsPage.xaml)

To create a new Page to your project, right click your project in Solution Explorer and select Add, New Item... Select Windows Phone Portrait Page and name it to `ShowsPage.xaml`



Shows Page function is to display selected Theatre's shows/movies in a `ListBox` Control. The page grid contains background, title and listbox.

```
<Grid x:Name="LayoutRoot">
        <Grid.Background>
            <ImageBrush ImageSource="Images/bg2.png"/>
        </Grid.Background>
        <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="FiNNKiNO - Kaupunki"
                       Foreground="White"
                       Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="elokuvat" Margin="9,-7,0,0"
                       Foreground="White"
                       Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!-- Shows ListBox Control -->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <ListBox x:Name="ShowsListBox"
                     Foreground="White"
                     ItemTemplate="{StaticResource ShowDataTemplate}"/>
        </Grid>
    </Grid>
```

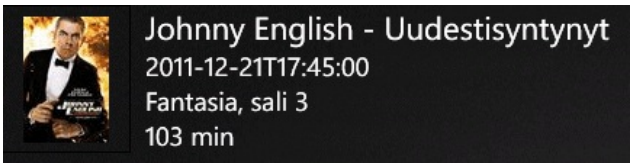
Background image is handled as it is done earlier in this example. The interesting part in this Page is in the `ListBox` and it's `ItemTemplate`.

In Windows Phone programming you can make a templates and use those templates in different pages to show same kind of information. I made this example first using the Pivot Application Template and all the Theatre's shows are then displayed in different Pivots. Then I had more than 15 different pages (pivots) with same kind of lists of shows in different Theatre Areas (and forced to create template to display shows in `ListBox`). I moved to Panorama view because of the memory management in Pivots (I really not need to show all the Theatre Area Shows at the same time, right?).

`ShowsDataTempate` is described in {Icode|App.xaml}} file. In left there are 100 px space for the image and right side are used with `TextBlock` Controls.

(C) Copyright Nokia 2013. All rights reserved.

Data are binded to UI from ShowsPage Class.




```
<Application.Resources>
 <DataTemplate x:Key="ShowDataTemplate">
  <Grid>
   <Grid.ColumnDefinitions>
    <ColumnDefinition Width="100"/>
    <ColumnDefinition Width="*"/>
   </Grid.ColumnDefinitions>
   <Border Margin="5" BorderBrush="Black" BorderThickness="1">
    <Image delay:LowProfileImageLoader.UriSource="{Binding EventSmallImagePortrait}" Width="65" Height="87"/>
   </Border>
   <StackPanel Grid.Column="1" Margin="5">
    <TextBlock Text="{Binding Title}" FontSize="22" Foreground="White" />
    <TextBlock Text="{Binding ShowStart}" FontSize="18" Foreground="White"/>
    <TextBlock Text="{Binding TheatreAndAuditorium}" FontSize="18" Foreground="White"/>
    <TextBlock Text="{Binding LengthInMinutes, StringFormat='\{0\} min'}" FontSize="18" Foreground="White" />
   </StackPanel>
  </Grid>
 </DataTemplate>
</Application.Resources>
```

There can be like a 100 images loading in this page and thats why we are using PhonePerformance Reference. Remember add reference to that dll in your project references and following XML Namespace to your `App.xml` file.

```
xmlns:delay="clr-namespace:Delay;assembly=PhonePerformance"
```

Programming (ShowsPage.xaml.cs)

`OnNavigatedTo` method will be called when Shows page is displayed. Selected Theatre index will be found in this Page `NavigationContext`. Selected Theatre's data are in `theatreAreas` and `theatreShows` Lists in `App.xaml` Class.

```
App app = App.Current as App; // reference to App Class
int TheatreSelectedIndex;     // selected Theatre Index

protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
 base.OnNavigatedTo(e);

 // find Theatre's selected index (sent from ShowsPage.xaml)
 IDictionary<string, string> parameters = this.NavigationContext.QueryString;
 if (parameters.ContainsKey("SelectedIndex"))
 {
  // Theatre SelectedIndex (ListBox)
  TheatreSelectedIndex = Int32.Parse(parameters["SelectedIndex"]);
  // Show Selected Theatre name in Page Title
  ApplicationTitle.Text = "Finnkino - " + app.theatreAreas[TheatreSelectedIndex].name;
  // Bind data to ListBox (all events/shows in this theatre)
  ShowsListBox.ItemsSource = app.theatreShows[TheatreSelectedIndex];
  // Handle Selection of the ListBox
  ShowsListBox.SelectionChanged += new SelectionChangedEventHandler(ShowsListBox_SelectionChanged);
 }
}
```

A new Show Details Page will be displayed when the user selects the Show from the `ShowsListBox`. Selected Theatre and Show indexes will be passed to this new class as a `GET` method.

```
int selectedIndex = (sender as ListBox).SelectedIndex;
this.NavigationService.Navigate(new Uri("/ShowDetailPage.xaml?TheatreSelectedIndex=" + TheatreSelectedIndex+"&ShowSelectedIndex="+se
```

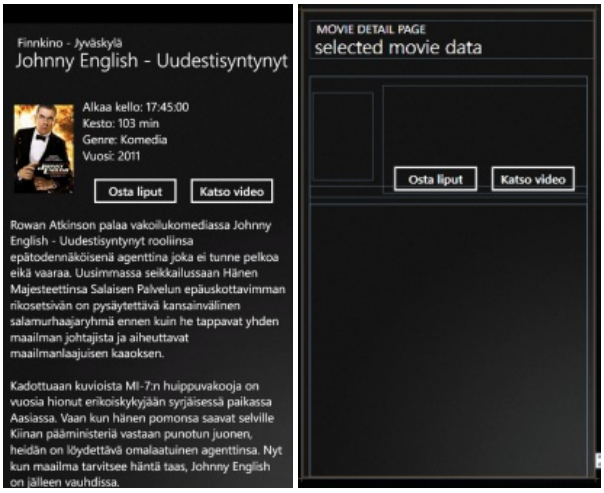
Show Detail Page

Design (ShowDetailPage.xaml)

To create a new Page to your project, right click your project in Solution Explorer and select Add, New Item... Select Windows Phone Portrait Page and name it to `ShowDetailPage.xaml`

This Show Detail Page display selected movies details (see picture in left). In Design Mode the grid is devided to Page Title, Image and basic informations with TextBlocks and Buttons. Movie's synopsis text is displayed at the bottom of the screen (see picture in right).

(C) Copyright Nokia 2013. All rights reserved.



Show Detail Page's background and title is done same way as a previous pages in this example. Only a smaller text font in title is used. The main content of the page is more different.

There is a space for image below title and movie details (and Buttons) are grouped in to StaticPanel blocks (right side of the image).

```
<Grid.ColumnDefinitions>
 <ColumnDefinition Width="110"/>
 <ColumnDefinition Width="*"/>
</Grid.ColumnDefinitions>
<Image x:Name="Image"
       Grid.Column="0"
       Width="99"
       Height="146"
       VerticalAlignment="Center"
       HorizontalAlignment="Center"
       Margin="6"/>
<StackPanel Margin="10,15,0,0" Grid.Column="1" Height="200" VerticalAlignment="Top">
 <StackPanel Grid.Row="0" Height="120">
  <TextBlock x:Name="ShowStart" FontSize="20" Foreground="White"/>
  <TextBlock x:Name="LengthInMinutes" FontSize="20" Foreground="White" />
  <TextBlock x:Name="Genres" FontSize="20" Foreground="White"/>
  <TextBlock x:Name="ProductionYear" FontSize="20" Foreground="White"/>
 </StackPanel>
 <StackPanel Margin="0,0,0,0" Grid.Row="1" Height="70" VerticalAlignment="Top" HorizontalAlignment="Left" Orientation="Horizontal">
  <Button Content="Osta liput" FontSize="20" Height="64" Width="160" BorderBrush="White" Foreground="White" Click="OstaLiput_Click"/
  <Button Content="Katso video" FontSize="20" Height="64" Width="160" BorderBrush="White" Foreground="White" x:Name="PlayVideo" Clic
 </StackPanel>
</StackPanel>
</Grid>
```

Movie's synopsis text is displayed in `TextBlock` which is in `ScrollView`.

```
<Grid Height="480" VerticalAlignment="Top" Margin="0,200,0,0">
 <ScrollViewer Height="400" Grid.ColumnSpan="2" Margin="0,0,0,0">
    <TextBlock x:Name="Synopsis"
        FontSize="20"
        Foreground="White"
        TextWrapping="Wrap"/>
  </ScrollViewer>
</Grid>
```

Programming (ShowDetailPage.xaml)

Like in previous Page `OnNavigatedTo` method will be called when Show Detail page is displayed. Before that I will mention that we use here a few variables inside `ShowDetailPage` Class.

```
App app = App.Current as App;       // reference to App Class
MediaPlayerLauncher VideoLaucher;   // used to display movie trailer

Show selectedShow;                  // selected Show object
string selectedVideo = "";          // video url of the selected Show object
```

In `OnNavigatedTo` method selected Theatre and Show index is passed via `GET` parameters. After that movie's information (start time, genre, year and length) are displayed. And finally selected show's synopsis and video Url are found from `theatreEvents` List (based on IDs). A new `MediaPlayerLauncher` object is created if there are a movie trailer available in selected show.

```
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
 base.OnNavigatedTo(e);

 IDictionary<string, string> parameters = this.NavigationContext.QueryString;
 if (parameters.ContainsKey("TheatreSelectedIndex") && parameters.ContainsKey("ShowSelectedIndex"))
 {
  // Theatre SelectedIndex (ListBox)
  int TheatreSelectedIndex = Int32.Parse(parameters["TheatreSelectedIndex"]);
  ApplicationTitle.Text = "Finnkino - " + app.theatreAreas[TheatreSelectedIndex].name;
  // selected show
  int ShowSelectedIndex = Int32.Parse(parameters["ShowSelectedIndex"]);
  selectedShow = (Show) app.theatreShows[TheatreSelectedIndex][ShowSelectedIndex];
  PageTitle.Text = selectedShow.Title;

  ShowStart.Text = "Alkaa kello: " + selectedShow.ShowStart;
```


(C) Copyright Nokia 2013. All rights reserved.

```
    Genres.Text = "Genre: " + selectedShow.Genres;
    ProductionYear.Text = "Vuosi: " + selectedShow.ProductionYear;
    LengthInMinutes.Text = "Kesto: " + selectedShow.LengthInMinutes + " min";


    // find selected show -> event id
    for (var i = 0; i < app.theatreEvents.Count(); i++)
    {
     if (app.theatreEvents[i].EventID == selectedShow.EventID)
     {
      // synopsis text
      Synopsis.Text = app.theatreEvents[i].Synopsis;
      // show video
      if (app.theatreEvents[i].Video.Count() > 0)
      {
       selectedVideo = app.theatreEvents[i].Video.Descendants("Location").ElementAt(0).Value;
      }
      break;
     }
    }

    // small image
    Image.Source = new BitmapImage(new Uri(selectedShow.EventLargeImagePortrait, UriKind.RelativeOrAbsolute));

    if (selectedVideo != "")
    {
     VideoLaucher = new MediaPlayerLauncher();
    }
    else
    {
     PlayVideo.IsEnabled = false;
    }

   }
  }
```

This Class has two buttons and those `Button`'s event handlers are the last programming parts here.

Windows Phone's default Browser will be launched when user clicks "Osta liput" `Button`.

```
private void OstaLiput_Click(object sender, RoutedEventArgs e)
{
 WebBrowserTask webbrowser = new WebBrowserTask();
 webbrowser.Uri = new Uri(selectedShow.ShowURL);
 webbrowser.Show();
}
```

Windows Phone's default Video Player will be launched when user clicks "Katso video" `Button`.

```
private void PlayVideo_Click(object sender, RoutedEventArgs e)
{
 VideoLaucher.Controls = MediaPlaybackControls.All;
 VideoLaucher.Location = MediaLocationType.Install;
 VideoLaucher.Media = new Uri(selectedVideo, UriKind.Absolute);
 VideoLaucher.Show();
}
```

Handling Tombstoned Mode

(C) Copyright Nokia 2013. All rights reserved.

Like I wrote before, Windows Phone might send your application to Tombstoned mode and then you have to store your application data and load it back when your application is running again. I decided to do it in the `App` Class where all the application data is now stored in to `List` Collections.

```
public partial class App : Application
{
 public List<TheatreArea> theatreAreas;                          // Theatre Areas
 public List<Event> theatreEvents;                               // Theatre Events
 public List<List<Show>> theatreShows = new List<List<Show>>();  // Theatre Shows
 public bool dataLoaded;                                         // is data loaded from Finnkino's server
        ...
}
```

`App` Class has `Application_Launching`, `Application_Activated` and `Application_Deactivated` methods which will be called automatically by the system.

Application_Launching method

`Application_Launching` method will be called only when the application is launching, it will not be executed when the application is reactivated again. Here I only initialize dataLoaded variable to false.

```
private void Application_Launching(object sender, LaunchingEventArgs e)
{
 dataLoaded = false;
}
```

Application_Activated method

`Application_Activated` method will be called when the application is activated (brought to foreground). Here I first check is application coming back from Dormant or Tombstoned mode. All the application data is loaded back to List Collections if Tombstoned mode is detected.

```
private void Application_Activated(object sender, ActivatedEventArgs e)
{
 if (e.IsApplicationInstancePreserved)
 {
  // DORMANT
 }
 else
 {
  // TOMBSTONED
  if (PhoneApplicationService.Current.State.ContainsKey("TheatreAreas"))
  {
   theatreAreas = (List<TheatreArea>)PhoneApplicationService.Current.State["TheatreAreas"];
  }

  if (PhoneApplicationService.Current.State.ContainsKey("TheatreEvents"))
  {
   theatreEvents = (List<Event>)PhoneApplicationService.Current.State["TheatreEvents"];
  }

  if (PhoneApplicationService.Current.State.ContainsKey("TheatreShows"))
  {
   theatreShows = (List<List<Show>>)PhoneApplicationService.Current.State["TheatreShows"];
  }
 }
 dataLoaded = true;
}
```

Application_Deactivated method

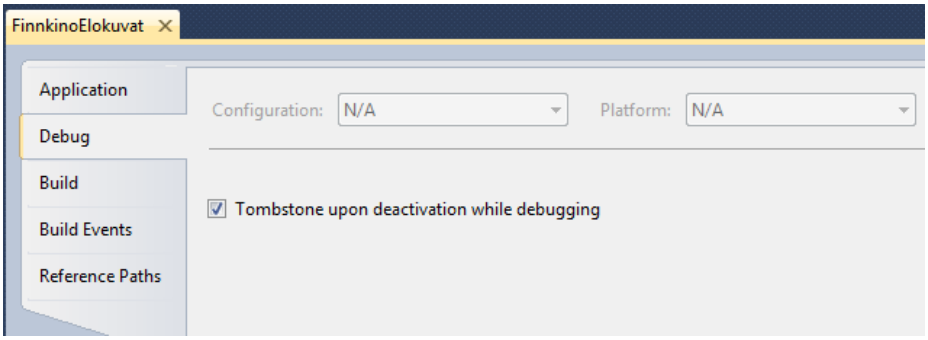
`Application_Deactivated` will be executed when the application is deactivated (sent to background). Here I save the List Collections data to the phone memory.

```
private void Application_Deactivated(object sender, DeactivatedEventArgs e)
{
 if (!PhoneApplicationService.Current.State.ContainsKey("TheatreAreas"))
  PhoneApplicationService.Current.State.Add("TheatreAreas", theatreAreas);
 if (!PhoneApplicationService.Current.State.ContainsKey("TheatreEvents"))
  PhoneApplicationService.Current.State.Add("TheatreEvents", theatreEvents);
 if (!PhoneApplicationService.Current.State.ContainsKey("TheatreShows"))
  PhoneApplicationService.Current.State.Add("TheatreShows", theatreShows);
}
```

Testing Tombstoned Mode in Windows Phone Emulator

It is a quite hard to test Tombstoned mode in real phone but you can test it easily in Windows Phone Emulator. Right Click your project from Solutions Explorer and select Properties. Select Debug Tab and check Tombstone upon deactivation while debugging.

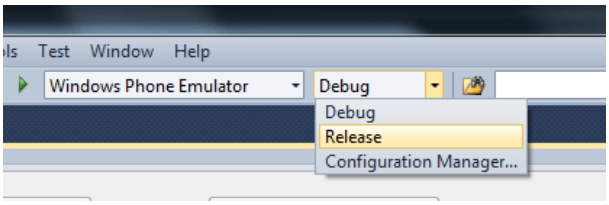
(C) Copyright Nokia 2013. All rights reserved.



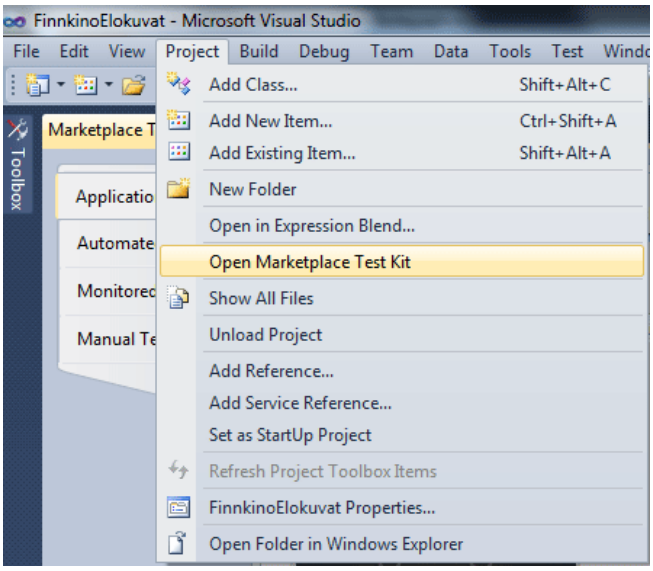
Now run your Finnkino Elokuvat application in emulator and wait it to load all the data from Finnkino's server. Click Windows Button in the emulator when any page is visible in Finnkino Elokuvat application. Now Finnkino Elokuvat application is sent to the background (`Application_Deactivated` will be called and data is saved to device) and all of it's data is removed from the memory. Browse back to Finnkino Elokuvat application and then `Application_Activated` method will be called and all the Theatre's data is loaded back to memory.

Application Testing

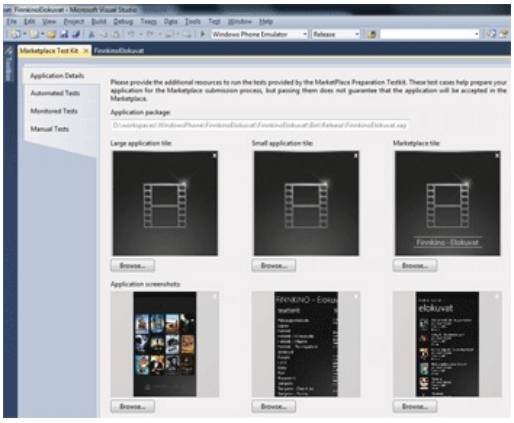
It is good to test your application before you publish it to Windows Market Place. You can open and run Marketplace Test Kit in your Visual Studio, but first you have to build your application in Release mode.



Select Project and then Open Marketplace Test Kit from the main menu.



First you have to add additional resources to run these tests. Basically those are different size of images for the Marketplace. You can run automated test in the second tab from the Test Kit. It will give you information about your XAP file, used Capabilities and are your icons and screenshots correctly sized.



(C) Copyright Nokia 2013. All rights reserved.

Passed: 4 Failed: 0

Result	Test Name	Test Description	Result Details
Passed	XAP Package Requirements	Validation of XAP file size and content files	
Passed	Capability Validation	Validation of application capabilities	[INFORMATION] : Capabilities used by application : ID_CAP_NETWORKING ID_CAP_MEDIALIB
Passed	Iconography	Validation of Application Icons	
Passed	Screenshots	Validation of Screenshots	

Monitored test is used used to analyze your application performance and functionality. You have to start your application, navigate forward and back in the application and finally close application to get test data generated. You should run this in real device connected to Visual Studio to get real test.

There are a lot of different Manual Test also available in the Marketplace Test Kit.

Summary

This is one of the biggest articles what I have ever written. Hope you find this article useful and it helps you work with XML in WP.

You can download source codes here: File:PTMFinnkinoElokuvat.zip

This application is also available in Windows Marketplace: Finnkino Elokuvat

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.