How to create a Namedays application for Windows Phone 7

This article explains how Namedays application is made for the Windows Phone 7. It shows how to load XML data from the server side and bind data to UI. It also illustrates how to localize the application for different languages and how to use Scheduled Task Agent to update application.

Introduction

This article describes how the Namedays application is programmed for Windows Phone 7. The app is available in Windows Phone Marketplace here.

Namedays application show selected country's name days. Name days throughout the year are visible in application and today's names are displayed in Live Tile. So it is really easy to see and remember if your friends have a name day. There are 15 countries included: Austria, Czech, Finland, Finland Swedish, France, Germany, Hungary, Italy, Latvia, Poland, Slovak, Spain, Sweden, UK and USA.

Here is a small demo video, which shows how the application works:

The media player is loading...

This article will cover following aspects:

- basic structures of XML files in the application back end
- how to use Windows Phone SDK and creating Windows Phone Project
- how to localize your application to use different languages
- how to load data from the server side and use XML deserialization to parse XML data to own made objects
- bind loaded XML data to ListBox control in the UI
- passing variables between application pages with Isolated Storage
- use ShellTile class to modify application Live Tile
- use Scheduled Task Agent behind the application to update application Live Tile

Note: This code example demonstrates how this kind of application can be done. Windows Phone 7 source code is available, but the real back end with PHP and XML files are not included.

Back end of this application

This application uses PHP and XML files as an back end. Each country's namedays data are stored in the XML file. Use following structure to describe your favourite countries namedays.

1. root element is `resources`
2. each month is as `array` element and there is `name` attribute which value is month name
3. each `item` describes one day namedays, separated with comma (,)

```
<!-- nameday_Finland.xml -->
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <array name="january">
                <item>-</item>
                <item>Aapeli</item>
                <item>Elmeri,Elmo,Elmer</item>
                <item>Ruut</item>
                <item>Lea,Leea</item>
    ...
                <item>Alli</item>
```


(C) Copyright Nokia 2013. All rights reserved.

```
        </array>
        <array name="february">
                <item>Riitta</item>
                <item>Jemina,Lumi,Aamu</item>
  ...
        </array>
        ...
</resources>
```

There is only one small PHP file (**namedaysWP7.php**) used in this code example. This PHP file returns all the available countries to Windows Phone application.

```
<?php
  echo "Austria,Czech,Finland,FinnishSwedish,France,Germany,Hungary,Italy,Latvia,Poland,Slovak,Spain,Sweden,Uk,Usa";
?>
```

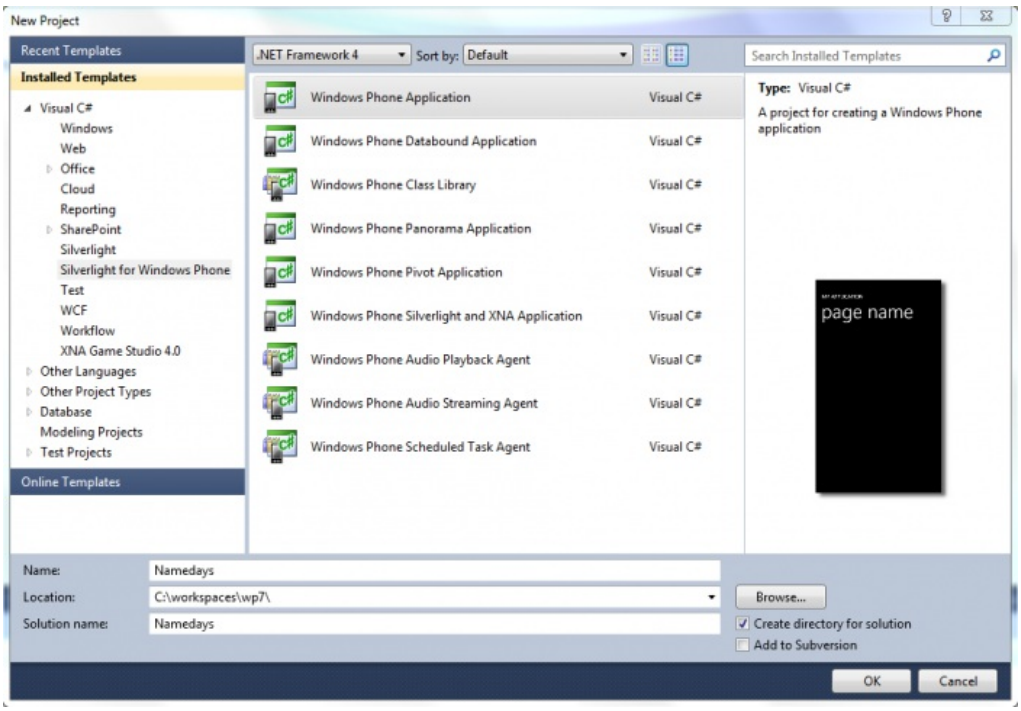
So it is very easy to add a new countries to this application. Create a new XML data and include country in PHP file.

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

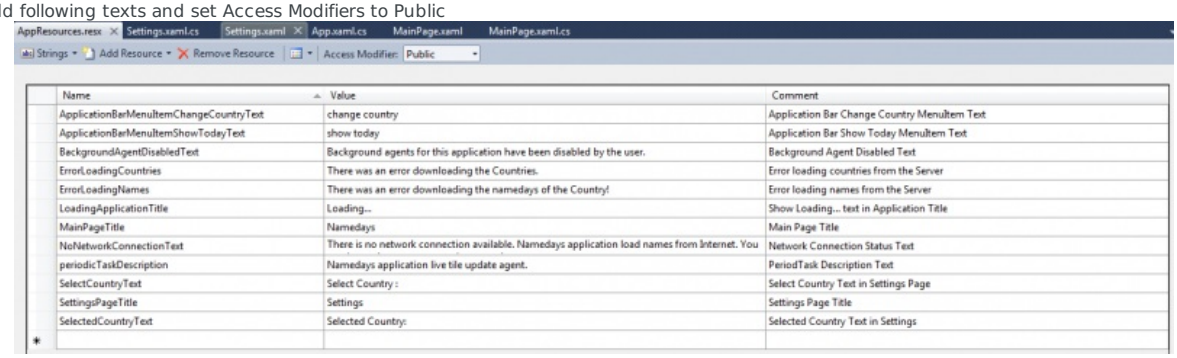
To start creating a new Windows Phone application, start Microsoft Visual Studio then create a new Project and select Windows Phone Application Template.



Localization

This application uses English and Finnish texts in the UI. First add a resource file (.resx) for each language:

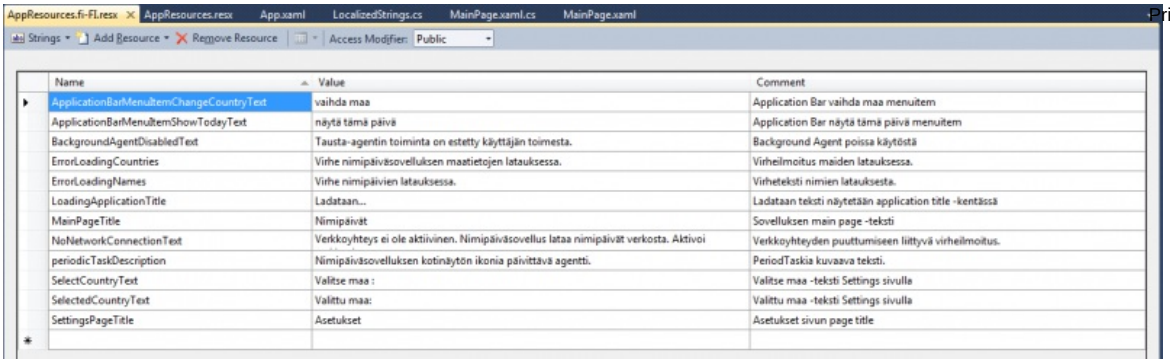
1. Select **Add** > **new item** from project Solution Explorer
2. Select General from Installed Templates and create a new Resource File
3. Name it for example **AppResources.resx**
4. Add following texts and set Access Modifiers to Public



Name	Value	Comment
ApplicationBarMenuItemChangeCountryText	change country	Application Bar Change Country MenuItem Text
ApplicationBarMenuItemShowTodayText	show today	Application Bar Show Today MenuItem Text
BackgroundAgentDisabledText	Background agents for this application have been disabled by the user.	Background Agent Disabled Text
ErrorLoadingCountries	There was an error downloading the Countries.	Error loading countries from the Server
ErrorLoadingNames	There was an error downloading the namedays of the Country!	Error loading names from the Server
LoadingApplicationTitle	Loading...	Show Loading... text in Application Title
MainPageTitle	Namedays	Main Page Title
NoNetworkConnectionText	There is no network connection available. Namedays application load names from Internet. You	Network Connection Status Text
periodicTaskDescription	Namedays application live tile update agent.	PeriodTask Description Text
SelectCountryText	Select Country :	Select Country Text in Settings Page
SettingsPageTitle	Settings	Settings Page Title
SelectedCountryText	Selected Country:	Selected Country Text in Settings

5. create a new resource for Finnish (other language) and name it to **AppResources.fi-FI.resx**

(C) Copyright Nokia 2013. All rights reserved.

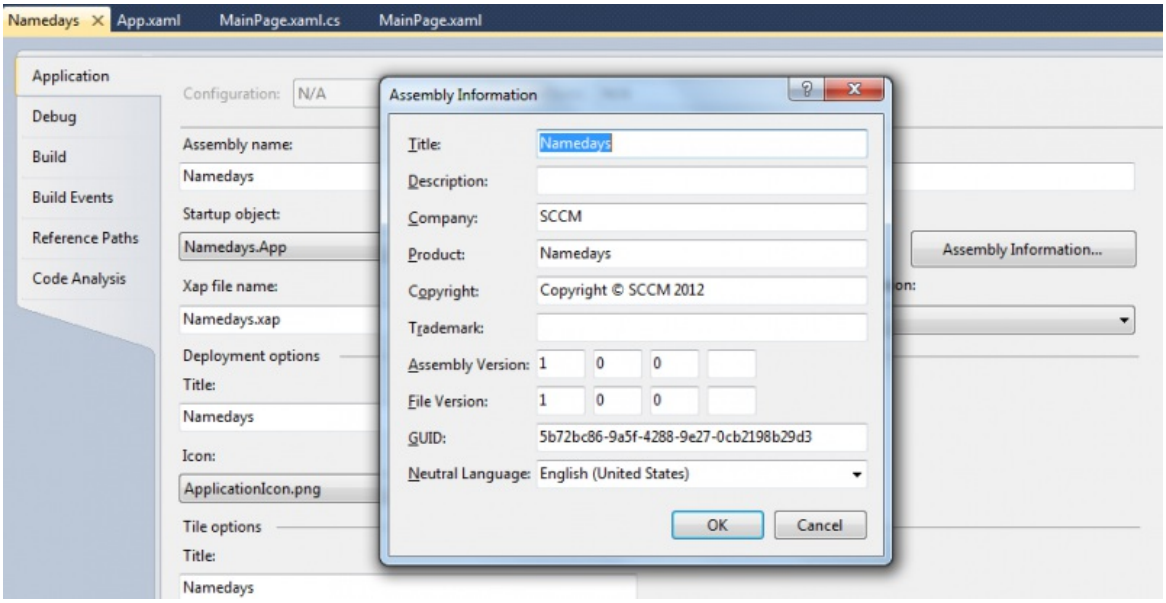


Close your project and add supported localization to project csproj file. Use your favourite editor to add following `<SupportedCultures>fi-FI</SupportedCultures>` line.

```
...
<TargetFrameworkIdentifier>Silverlight</TargetFrameworkIdentifier>
<SilverlightApplication>true</SilverlightApplication>
<SupportedCultures>fi-FI</SupportedCultures>
<XapOutputs>true</XapOutputs>
<GenerateSilverlightManifest>true</GenerateSilverlightManifest>
<XapFilename>Namedays.xap</XapFilename>
...
```

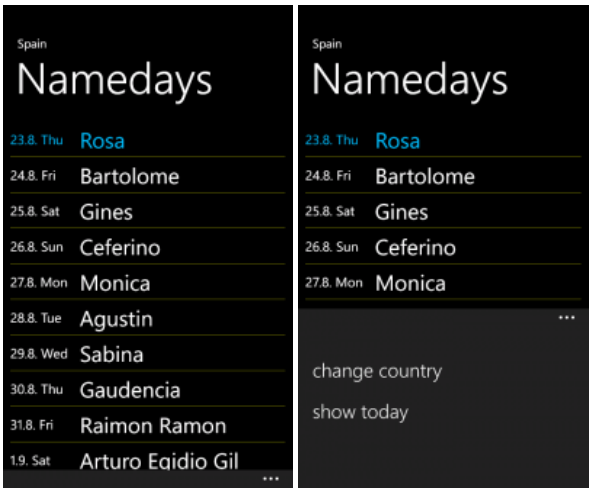
You can check your project's default language from project properties:

1. select your project from Solutions Explorer
2. select Assembly Information... from Application Tab
3. select your Neutral Language (now it is English).



Main Page of the application

Main Page shows selected country's namedays in a `ListBox`. User can change country via Application Bar (it opens a new Settings Page).



(C) Copyright Nokia 2013. All rights reserved.

Design (MainPage.xaml)

The `ContentPanel` grid contains `ListBox` to show namedays data. Each list item displays the date and names of the day. Data is binded from `DayItemObject` class to NamesList (will be discussed later in this code example).

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
 <ListBox x:Name="NamesList">
  <ListBox.ItemTemplate>
   <DataTemplate>
    <Border BorderBrush="Yellow" Margin="0" BorderThickness="0,0,0,0.5">
     <StackPanel Orientation="Horizontal" Margin="0,5,0,5" VerticalAlignment="Top">
      <TextBlock Text="{Binding date}" Margin="0,0,15,0"
       FontSize="22"
       Foreground="{Binding fgColor}" VerticalAlignment="Center"
       Width="100"
       TextAlignment="Left"/>
      <TextBlock Text="{Binding names}"
          FontSize="36"
          Foreground="{Binding fgColor}" VerticalAlignment="Top"
          TextAlignment="Left"
          TextWrapping="Wrap"
          Width="340"/>
     </StackPanel>
    </Border>
   </DataTemplate>
  </ListBox.ItemTemplate>
 </ListBox>
</Grid>
```

Application Bar shows selection only in text (no IconButtons used). Menu Item texts will be replaced from the code using localization data from Application Resources. Now there are only sample texts visible.

```
<phone:PhoneApplicationPage.ApplicationBar>
 <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True" Mode="Minimized">
  <shell:ApplicationBar.MenuItems>
   <shell:ApplicationBarMenuItem x:Name="ChangeCountry"
    Text="Change country" Click="Settings_Click"/>
   <shell:ApplicationBarMenuItem x:Name="ShowToday"
    Text="Show Today" Click="ShowToday_Click"/>
  </shell:ApplicationBar.MenuItems>
 </shell:ApplicationBar>
</phone:PhoneApplicationPage.ApplicationBar>
```

Programming (MainPage.xaml.cs)

It is time to test localization first. Create a class to load localized strings from Application Resources.

1. select your project from Solution Explorer
2. select add, class..
3. and name it to **LocalizedStrings.cs**

```
namespace Namedays
{
    public class LocalizedStrings
    {
        public LocalizedStrings()
        {
        }

        // connect and return resources
        private static Namedays.AppResources localizedResources = new Namedays.AppResources();
        public Namedays.AppResources LocalizedResources { get { return localizedResources; } }
    }
}
```

Next connect created class to `App.xaml`, add a `LocalizedStrings` key to `LocalizedString` class.

```
 <!--Application Resources in App.xaml -->
    <Application.Resources>
        <local:LocalizedStrings xmlns:local="clr-namespace:Namedays" x:Key="LocalizedStrings" />
    </Application.Resources>
```

Now you can test localization for example in Main Page's `OnNavigatedTo` method. Add following method to `MainPage.xaml.cs` class.

```
// this page is navigated to
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
 // test localization to show main page title in messagebox
 MessageBox.Show(AppResources.MainPageTitle);
}
```

You should see following informations in `MessageBox`. Change your phone langugage settings to use Finnish and English.

(C) Copyright Nokia 2013. All rights reserved.

This application loads data from network, so it is best first to check, if network connection is available. After that application stores running information (true) to Isolated Storage (This application uses Scheduled Agent to update Live Tile. Now application is running and Live Tile is not updated from Scheduled Agent). Application stores running information (false) later in OnNavigatedFrom method when application is closed. Finally earlier stored country info is loaded from Isolated Storage and country's namedays data is loaded from the server.

Add following variables to Main Page.

```
// namedays countries name data
private List<List<string>> names = new List<List<string>>();

// isolated storage data
private IsolatedStorageSettings appSettings;
const string CountrySettingsKey = "CountryKey";
const string NamesSettingsKey = "NamesKey";
const string RunningSettingsKey = "RunningKey";

// default values
private string country = "Finland";
private int todayPosition = 0;
private bool isNet = false;
```

Add connection to Isolated Storage in `MainPage` constructor.

```
// Constructor
public MainPage()
{
 InitializeComponent();
 // connect to isolated storage
 appSettings = IsolatedStorageSettings.ApplicationSettings;
}
```

`OnNavigatedTo` method will do all the magic work to start loading data from the server side.

```
// this page is navigated to
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
 // test localization to show main page title in messagebox
 //MessageBox.Show(AppResources.MainPageTitle);

 if (!System.Net.NetworkInformation.NetworkInterface.GetIsNetworkAvailable())
 {
  // load and show error text from AppResources (localization)
  MessageBox.Show(AppResources.NoNetworkConnectionText);
  return;
 }
 // network is available
 isNet = true;

 // app is now running, save running information to isolated storage
 appSettings.Remove(RunningSettingsKey);
 appSettings.Add(RunningSettingsKey, "true");

 // read country from isolated storage, if there is no country -> Finland is used by default
 if (appSettings.Contains(CountrySettingsKey))
 {
  country = (string)appSettings[CountrySettingsKey];
 }

 // set application title (do not use Finland from server data, use Suomi :-)
 if (country == "Finland" && CultureInfo.CurrentCulture.Name == "fi-FI") ApplicationTitle.Text = "Suomi";
 else ApplicationTitle.Text = country;

 // set page title from resources
 PageTitle.Text = AppResources.MainPageTitle;

 // set application menu items
 ApplicationBarMenuItem mi1 = (ApplicationBarMenuItem)ApplicationBar.MenuItems[0];
 ApplicationBarMenuItem mi2 = (ApplicationBarMenuItem)ApplicationBar.MenuItems[1];
 // set menu item text from resources (localization)
 mi1.Text = AppResources.ApplicationBarMenuItemChangeCountryText;
 mi2.Text = AppResources.ApplicationBarMenuItemShowTodayText;

 // start loading names from server
 LoadNames();
}
```

`LoadNames` method will be called when `OnNavigatedTo` method is ending. Here WebClient Class is used to load data from the server side. Remember change your own server address to load XML data.

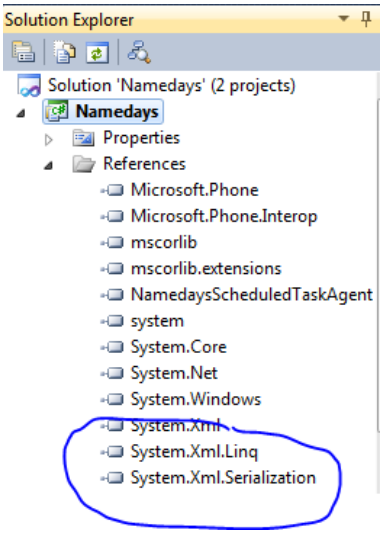
```
// load names data from server
private void LoadNames()
{
 // use WebClient class to load data
 WebClient downloader = new WebClient();
```


(C) Copyright Nokia 2013. All rights reserved.

```
 Uri uri = new Uri("http://your_server_here/nameday_" + country + ".xml", UriKind.Absolute);
 downloader.DownloadStringCompleted += new DownloadStringCompletedEventHandler(NamesDownloaded);
 downloader.DownloadStringAsync(uri);
 // show Loading... text from resources (localization)
 PageTitle.Text = AppResources.LoadingApplicationTitle;
}
```

There are many different ways to load and parse XML data in Windows Phone 7. This example uses 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:

1. right click "References" in your project Solutions Explorer
2. select "Add new Reference..."



`NamesDownloaded` method will be called and names are parsed from loaded XML-file to List with XDocument Class. First names list object will be cleared and all the names data are returned as an array. Each array contains the names of the month which are stored in the List collection (`monthNames`) and finally all the loaded month names are stored to names list. This method calls `AddListItem` method to show data in the UI.

```
▼<resources>
  ▶<array name="january">...</array>
  ▶<array name="february">...</array>
  ▼<array name="march">
     <item>Alpo,Alvi,Alpi</item>
     <item>Virve,Virva,Fanni</item>
     <item>Kauko</item>
     <item>Ari,Arsi,Atro</item>
     <item>Leila,Laila</item>
     <item>Tarmo</item>
     <item>Taria,Taru</item>
```

```
private void NamesDownloaded(object sender, DownloadStringCompletedEventArgs e)
{
 // check possible errors
 if (e.Result == null || e.Error != null)
 {
  // show error text from resources (localization)
  MessageBox.Show(AppResources.ErrorLoadingNames);
 }
 else
 {
  // clear names list
  names.Clear();
  // loaded XML document data
  XDocument document = XDocument.Parse(e.Result);
  // get all month arrays
  var resources = from item in document.Descendants("array") select item;
  // loop to get all the names
  foreach (var month in resources)
  {
   List<string> monthNames = new List<string>();
   var arrays = from item in month.Descendants("item") select item;
   foreach (var name in arrays)
   {
    monthNames.Add(name.Value);
   }
   names.Add(monthNames);
  }
  // add ListItems
  AddListItems();
 }
 // set application title from resources (localization)
 PageTitle.Text = AppResources.MainPageTitle;
}
```

XML data is binded to `NamesList` in the UI with `DayItemsObjects` class. Create a new class to your project with following data: names, month, day, date and fgColor.

```
namespace Namedays
{
    public class DayItemObject
    {
        public string names { get; set; }
        public int month { get; set; }
        public int day { get; set; }
```


(C) Copyright Nokia 2013. All rights reserved.

```
        public string date { get; set; }
        public string fgColor { get; set; }
    }
}
```

This object includes the names of the day, the day data itself and text color used in UI.

Now names are separated with commas (,) in the XML data (this data is used in many other applications also) and this time here in Windows Phone names are displayed with blank ( ). RegularExpressions is used to replace characters. AddListItems method first loops through loaded XML data and generates `DayItemObject`'s which are binded to list in UI. Today's name will be highlited with different color (DeepSkyBlue). After all `DayItemObjects` are generated, those will be binded to UI with `NamesList ItemSource` property. Finally `NamesList` is positioned to today and `UpdateTile` method is called to update it's data. Updating and showing application Tile will be covered later in this article.

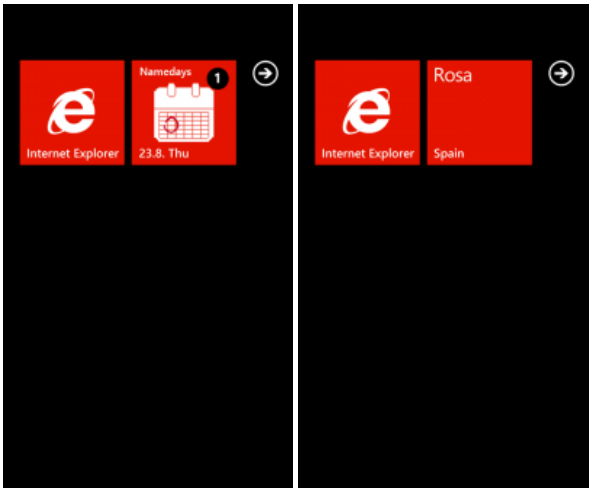
```
private void AddListItems()
{
 // convert "," to " "
 string pattern = ",";
 string replacement = " ";
 Regex rgx = new Regex(pattern);

 // get today from device
 DateTime dNow = DateTime.Now;
 int position = 0;
 string todayNames = "";
 string todayDate = "";

 // store DayItemObject objects
 List<DayItemObject> dios = new List<DayItemObject>();

 // loop throught XML data, and generate DayItemObjects
 for (int i = 0; i < names.Count(); i++)
 {
  for (int k = 0; k < names[i].Count(); k++)
  {
   // create a new DayItemObject object
   DayItemObject dio = new DayItemObject();

   // day and month in xml data -> date
   dio.day = k + 1;
   dio.month = i + 1;
   DateTime dt = new DateTime(dNow.Year, dio.month, dio.day);

   dio.date = dio.day + "." + dio.month + ". " + dt.ToString("ddd");
   dio.names = rgx.Replace(names[i][k], replacement);

   // highlite today
   if (dNow.Month == dio.month && dNow.Day == dio.day)
   {
    dio.fgColor = "DeepSkyBlue";
    todayPosition = position;
    //todayNames = dio.names;
    todayNames = names[i][k];
    todayDate = dio.date;
   }
   else
   {
    dio.fgColor = "White";
   }
   // name position in list
   position++;

   // add dio item to dios
   dios.Add(dio);
  }
 }
 // bind DayItemObjects to NamesList
 NamesList.ItemsSource = dios;

 // show today position in list
 showToday();

 // UpdateTile
 UpdateTile(todayNames, todayDate);
}
```

`showToday` method set the `NamesList` position to `todayPosition` and updates it's layout. `VisualTreeHelper` class is used to get `ScrollViewer` object from the List and finally list is positioned to point today with `ScrollToVerticalOffset` method.

```
private void showToday()
{
 // show today position
 NamesList.SelectedIndex = todayPosition;
 NamesList.UpdateLayout();
 ScrollViewer scrollViewer = VisualTreeHelper.GetChild(NamesList, 0) as ScrollViewer;
 scrollViewer.ScrollToVerticalOffset((double)todayPosition);
}
```

`UpdateTile` method finds the first tile of the application (the only one now) and update it's data. Tile's title shows today's date, background image is one of the available images in application (different images are used in Finland and other countries) and the names are displayed in the back of the tile.

(C) Copyright Nokia 2013. All rights reserved.




```
private void UpdateTile(string todayNames, string todaydate)
{
 string[] names = todayNames.Split(',');
 string tileNames = "";

 string tileBackground = "Images/Namedays173Eng.png";
 string tileCountry = country;

 // if Finland, use different background and Suomi text instead of Finland :-)
 if (country == "Finland" && CultureInfo.CurrentCulture.Name == "fi-FI")
 {
  tileCountry = "Suomi";
  tileBackground = "Images/Namedays173Fi.png";
 }

 // if there are more than 4 names, we cannot display oll the names (no space in tile),
 // show only 3 first and then ... (to show info that there are more names)
 if (names.Count() <= 4)
 {
  foreach (string name in names)
  {
   tileNames += name + "\n";
  }
 }
 else
 {
  for (int i = 0; i < 3; i++)
  {
   tileNames += names[i] + "\n";
  }
  tileNames += "...";
 }

 // get the application tile and update it's data
 ShellTile currentTiles = ShellTile.ActiveTiles.First();
 var tilesUpdatedData = new StandardTileData
 {
  Title = todaydate,
  BackgroundImage = new Uri(tileBackground, UriKind.Relative),
  Count = names.Count(),
  BackTitle = tileCountry,
  BackContent = tileNames,
  BackBackgroundImage = new Uri("Images/Background173.png", UriKind.Relative)
 };
 currentTiles.Update(tilesUpdatedData);
}
```

`OnNavigatedFrom()` method will be called when users ends this application. Now it is time to save names to Isolated Storage to later use and save application running information (false). This info is used with Scheduled Task agent later in this example.

```
protected override void OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs e)
{
 // save names to isolated storage
 appSettings.Remove(NamesSettingsKey);
 appSettings.Add(NamesSettingsKey, names);

 // save not running information to isolated storage
 appSettings.Remove(RunningSettingsKey);
 appSettings.Add(RunningSettingsKey, "false");
}
```

Main Page contains two other smaller methods which are called from the Application Bar.

Settings page can be opened if the network is available.

```
private void Settings_Click(object sender, EventArgs e)
{
 if (isNet) this.NavigationService.Navigate(new Uri("/Settings.xaml", UriKind.Relative));
}
```

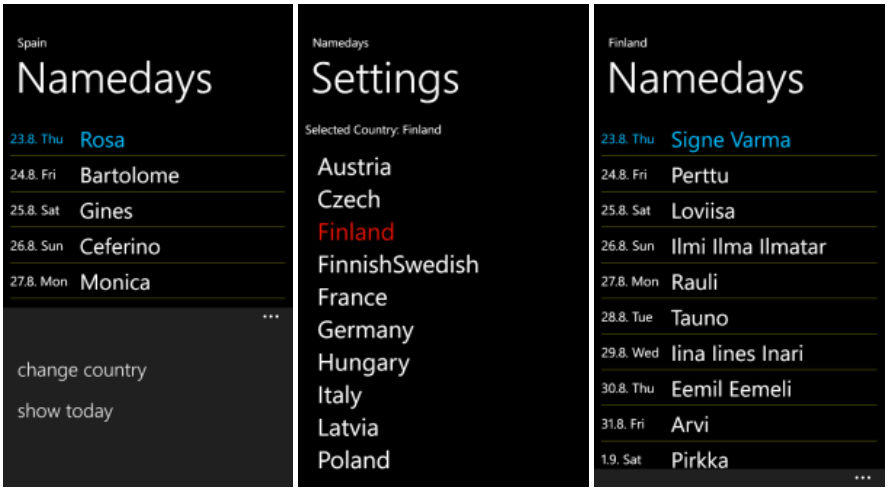
User can scroll the Main Page UI (List) freely and it is easy to get back todays position via Application Bar.

```
private void ShowToday_Click(object sender, EventArgs e)
{
 if (isNet) showToday();
}
```


(C) Copyright Nokia 2013. All rights reserved.

Changing the Country

User can change the country of the application here in the Settings Page. All the available countries are loaded from the server and countries are displayed in UI with List.



Design (SettingsPage.xaml)

Design Page's `ContentPanel` contains `TextBlock` to display informations and `ListBox` to display all the country names. Add a new `Page` to your application and name it *Settings*.

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
 <StackPanel Orientation="Vertical">
  <TextBlock Text="Selected Country : "
       Foreground="White"
       x:Name="SelectedCountry"/>
  <ListBox x:Name="CountriesList"
     VerticalAlignment="Top"
     FontSize="40"
     Foreground="White"
     ItemsSource="{Binding countries}"
     Margin="20,20,20,20"
     Height="540"
     SelectionChanged="CountriesList_SelectionChanged"/>
 </StackPanel>
</Grid>
```

Coding (SettingsPage.xaml)

The main idea of the Settings Page is store selected country name to Isolated Storage, so the Main Page can load selected country namedays data.

Use following variables to store needed data in this page.

```
private IsolatedStorageSettings appSettings;
const string CountrySettingsKey = "CountryKey";
private string country;
string[] countries;
```

Constuctor initialized UI and connect's to Isolated Storage.

```
public Settings()
{
 InitializeComponent();
 appSettings = IsolatedStorageSettings.ApplicationSettings;
}
```

`OnNavigatedTo` method read's the previously stored country data from Isolated Storage and shows the localized text in the UI. `LoadCountries()` method is used to start loading data from server side.

```
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
 // load previous country name from Isolated Storage
 if (appSettings.Contains(CountrySettingsKey))
 {
  country = (string)appSettings[CountrySettingsKey];
 }
 else
 {
  country = "Finland";
 }
 // show localized text in the UI
 ApplicationTitle.Text = AppResources.MainPageTitle;
 PageTitle.Text = AppResources.LoadingApplicationTitle;
 SelectedCountry.Text = AppResources.SelectedCountryText + " " + country;

 // load available countries from the server
 LoadCountries();
}
```


(C) Copyright Nokia 2013. All rights reserved.

`LoadCountries` method uses `WebClient` class to start loading data from the server side. Remember to change your own server address here.

```
// load available country names from server
private void LoadCountries()
{
 WebClient downloader = new WebClient();
 Uri uri = new Uri("http://your server here/namedaysWP7.php", UriKind.Absolute);
 downloader.DownloadStringCompleted += new DownloadStringCompletedEventHandler(CountriesDownloaded);
 downloader.DownloadStringAsync(uri);
}
```

`CountriesDownloaded` method will be called when loading is done. Method shows country names in the `CountriesList` list.

```
// country data is loaded from server
private void CountriesDownloaded(object sender, DownloadStringCompletedEventArgs e)
{
 if (e.Result == null || e.Error != null)
 {
  // show localized text in UI. (error)
  MessageBox.Show(AppResources.ErrorLoadingCountries);
 }
 else
 {
  // show countries names in List
  countries = e.Result.Split(',');
  foreach (var c in countries)
  {
   CountriesList.Items.Add(c);
  }
 }
 // show localized "settings" string in UI
 PageTitle.Text = AppResources.SettingsPageTitle;
}
```

Selected country will be saved to Isolated Storage, when the user navigates from this page.

```
// called when user navigates from this page (to the main page)
protected override void OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs e)
{
 appSettings.Remove(CountrySettingsKey);
 appSettings.Add(CountrySettingsKey, country);
}
```

The final method here in Settings Page is `CountriesList_SelectionChanged` method which is called when the user is selecting a new item from the `CountriesList`. This method only gives a new value to country variable (selected country name) which is stored to Isolated Storage when this page is ended.

```
// a new country is selected in UI
private void CountriesList_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
 int selectedIndex = (sender as ListBox).SelectedIndex;
 country = countries[selectedIndex];
 SelectedCountry.Text = AppResources.SelectedCountryText + " " + country;
}
```

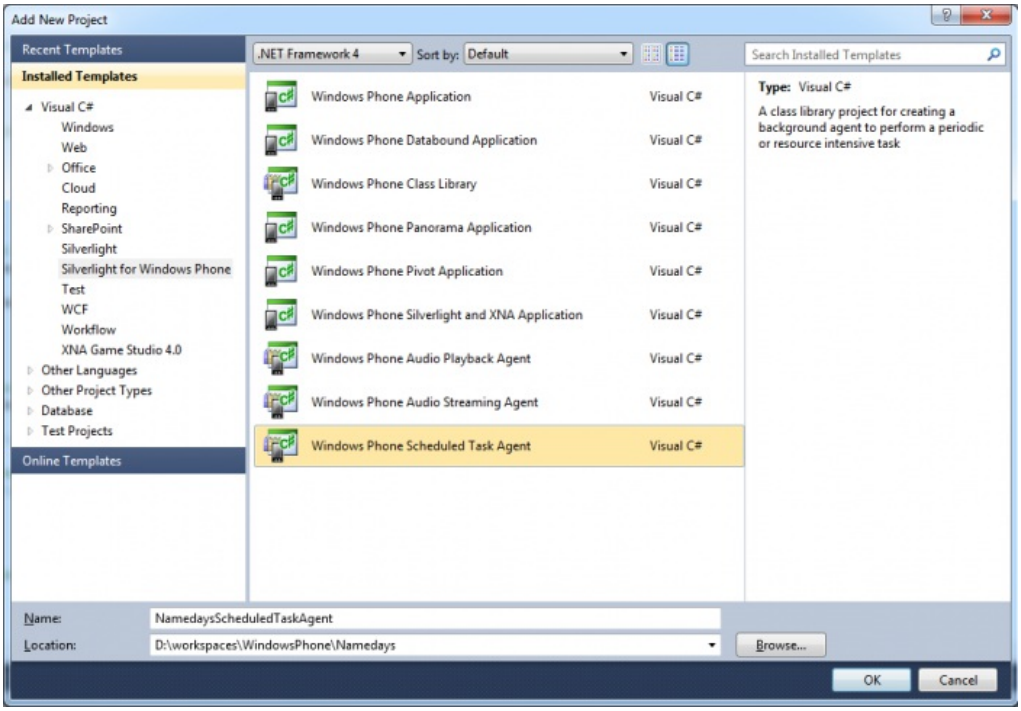
LiveTile - add Scheduled Task Agent to Namedays

Now you can test this application and it should work nicely. But if you keep your phone running over the night, the Live Tile is showing previous days data - not good! Scheduled Task Agent can be used to run tasks in Windows Phone.

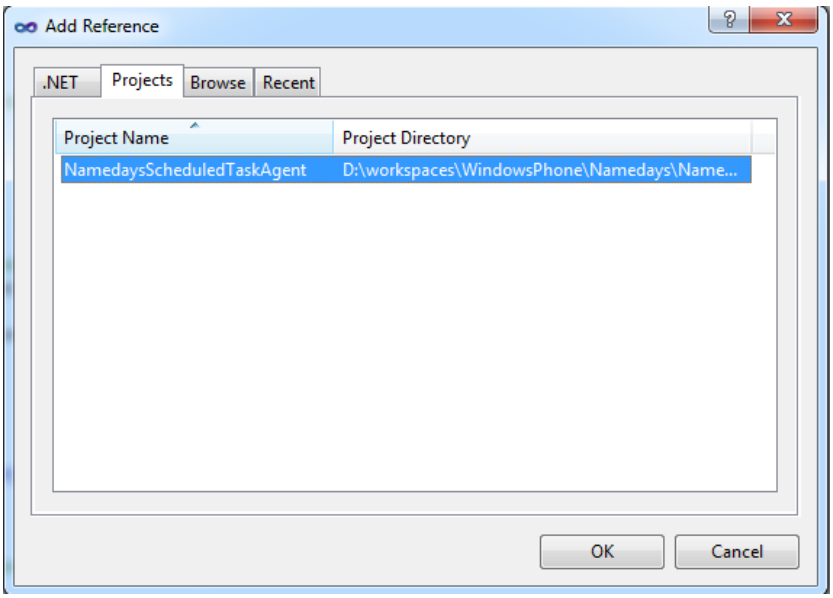
Add a new Scheduled Task Agent to your solution:

1. select your solution from Solution Explorer
2. add a new project
3. select Installed Templates, Silverlight for Windows Phone, Windows Phone Scheduled Task Agent
4. name your project for example NamedaysScheduledTaskAgent

(C) Copyright Nokia 2013. All rights reserved.



Scheduled Task Agent is not working right a way in your project, you should add reference to it from Namedays project.



Start Agent from MainPage.cs

Start using Scheduled Task Agent from your main project's MainPage.cs class with PeriodicTask Class. This class represents a scheduled task that runs regularly for a small amount of time.

Add a few new variables to Main.xaml.cs class.

```
PeriodicTask periodicTask;
string periodicTaskName = "NamedaysPeriodicAgent";
```

Scheduled Task Agent will be launched when user navigates from this application (ends it).

Modify OnNavigatedFrom method to include following method call.

```
// start agent
StartPeriodicAgent();
```

`StartPeriodicAgent` method will first check if there is a previous agent running and kill it. After that a new task is created and `NamedaysScheduledTaskAgent`'s **ScheduledAgent.cs** class is called. This call happens about once in 30 min in real device but you can test it with using debug settings for example in every 30 seconds.

Note! Add `#define DEBUG_AGENT` line in your code and remove those three commends below.

```
private void StartPeriodicAgent()
```


(C) Copyright Nokia 2013. All rights reserved.

```
{
 // is old task running, remove it
 periodicTask = ScheduledActionService.Find(periodicTaskName) as PeriodicTask;
 if (periodicTask != null)
 {
  RemoveAgent(periodicTaskName);
 }
 // create a new task
 periodicTask = new PeriodicTask(periodicTaskName);
 // load description from localized strings
 periodicTask.Description = AppResources.periodicTaskDescription;
 // set expiration days
 periodicTask.ExpirationTime = DateTime.Now.AddDays(14);
 try
 {
  // add thas to scheduled action service
  ScheduledActionService.Add(periodicTask);
  //#if(DEBUG_AGENT)
  //ScheduledActionService.LaunchForTest(periodicTaskName, TimeSpan.FromSeconds(30));
  //#endif
 }
 catch (InvalidOperationException exception)
 {
  if (exception.Message.Contains("BNS Error: The action is disabled"))
  {
   // load error text from localized strings
   MessageBox.Show(AppResources.BackgroundAgentDisabledText);
  }
  if (exception.Message.Contains("BNS Error: The maximum number of ScheduledActions of this type have already been added."))
  {
   // No user action required. The system prompts the user when the hard limit of periodic tasks has been reached.
  }
 }
 catch (SchedulerServiceException)
 {
  // No user action required.
 }
}
```

`RemoveAgent` method will be used to remove previously running agent.

```
// remove previous running agent
private void RemoveAgent(string name)
{
 try
 {
  ScheduledActionService.Remove(name);
 }
 catch (Exception)
 {
 }
}
```

Programming NamedaysScheduledTaskAgent Project

Background agent will call `ScheduleAgent.cs` class and more specially it's OnInvoke method to display new data in Live Tile. All the names are stored in the Isolated Storage, so data has to be load and then set names visible in application Live Tile.

Add a few variables to ScheduleAgent.cs class to handle names, Isolated Storage and data loading.

```
private IsolatedStorageSettings appSettings;
private List<List<string>> names = new List<List<string>>();
private string country = "Finland";

const string CountrySettingsKey = "CountryKey";
const string NamesSettingsKey = "NamesKey";
const string RunningSettingsKey = "RunningKey";
```

Connect to Isolated Storage in ScheduledAgent Constructor.

```
// connect to application isolated storage
appSettings = IsolatedStorageSettings.ApplicationSettings;
```

If Namedays application is running (visible in device UI) there is no need to update Live Tile, otherwise following steps are used in the code:

1. get country and names from Isolated Storage
2. get today data from device
3. display new names in Live Tile (if needed)

Live Tile is updated using the same algorithm which is used in the `MainPage` class.

Add following code to `OnInvoke` method to perform your task in background.

```
protected override void OnInvoke(ScheduledTask task)
{
 // is real application running (visible, no need to update tile)
 string running = "false";

 // is application running
 if (appSettings.Contains(RunningSettingsKey))
 {
  running = (string)appSettings[RunningSettingsKey];
 }

 // application is not running (visible in UI), update Live Tile
 if (running != "true")
 {
  // get country
  if (appSettings.Contains(CountrySettingsKey))
  {
   country = (string)appSettings[CountrySettingsKey];
```


(C) Copyright Nokia 2013. All rights reserved.

```
    }
    // get names
    if (appSettings.Contains(NamesSettingsKey))
    {
     names = (List<List<string>>)appSettings[NamesSettingsKey];
    }
    // today
    DateTime dNow = DateTime.Now;
    int month = dNow.Month;
    int day = dNow.Day;

    // is there data available in isolated storage (day and month => dayNames)
    bool dataAvailable = true;
    int months = names.Count();
    if (month > months) dataAvailable = false;
    if (dataAvailable)
    {
     int days = names[month - 1].Count();
     if (day > days) dataAvailable = false;
    }

    // no need to update Live Tile, all done here!
    if (!dataAvailable)
    {
     NotifyComplete();
     return;
    }

    // get names
    string[] todayNames = names[month - 1][day - 1].Split(',');
    string todaydate = day + "." + month + ". " + dNow.ToString("ddd");
    string tileNames = "";
    string tileBackground = "Images/Namedays173Eng.png";
    string tileCountry = country;

    // if country is Finland, show "Suomi" and different tile graphics
    if (country == "Finland")
    {
     tileCountry = "Suomi";
     tileBackground = "Images/Namedays173Fi.png";
    }

    // generate tileNames string
    if (todayNames.Count() <= 4)
    {
     foreach (string name in todayNames)
     {
      tileNames += name + "\n";
     }
    }
    else
    {
     for (int i = 0; i < 3; i++)
     {
      tileNames += todayNames[i] + "\n";
     }
     tileNames += "...";
    }

    // update Live Tile
    ShellTile currentTiles = ShellTile.ActiveTiles.First();
    var tilesUpdatedData = new StandardTileData
    {
     Title = todaydate,
     BackgroundImage = new Uri(tileBackground, UriKind.Relative),
     Count = todayNames.Count(),
     BackTitle = tileCountry,
     BackContent = tileNames,
    };
    currentTiles.Update(tilesUpdatedData);
   }

   //#if DEBUG_AGENT
   //ScheduledActionService.LaunchForTest(task.Name, TimeSpan.FromSeconds(30));
   //System.Diagnostics.Debug.WriteLine("Period Agent: "+task.Name);
   //#endif

   // complete
   NotifyComplete();
  }
```

Uncomment those `DEBUG_AGENT` comments and test. Live Tile should updated in every 30 seconds.

Summary

This code example shows how to load XML based data from the server and display data with localized texts. Application contains also Scheduled Task Agent to update application Live Tile. Hope you find this article useful and it helps you work with Windows Phone 7.

You can download source codes here: File:PTM Namedays.zip

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.