Saving application preferences in Windows Phone

This article explains how to save and load application preferences using IsolatedStorageSettings on Windows Phone 7.5.

Introduction

Saving user preferences is a common task requirement for applications. Common scenarios include saving the user's country, top scores in games, game sounds, login ids etc.

This articles discusses the procedure to save and load such user preferences using IsolatedStorageSettings class. It provides a dictionary-like, key-value store which is preserved across device reboots. The settings are securely stored inside the Isolated Storage space of an application, and are inaccessible to other applications.

In order to work best with the application life-cycle we load and save the settings when the settings page is loaded.

Sample application

Below, we create a simple application that stores the user's preferences in a game. Three elements such as Game Music, Timed Mode and Difficulty level have been identified as preferences for a game. Since Game Music and Timed Mode essentially hold boolean values, in the UI of application we use ToggleSwitch. ToggleSwitch does not come along with the Windows Phone SDK and has to be downloaded separately from codeplex and installed. This has been described here. This is rich resource for Windows Phone controls. Lets create the user interface for the settings page in XAML.

```
<phone:PhoneApplicationPage
    x:Class="Pref.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"
    xmlns:toolbox="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    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 GAME" Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="Settings" 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">

            <toolbox:ToggleSwitch Name="gameMusic" Header="Game Music"></toolbox:ToggleSwitch>
            <toolbox:ToggleSwitch Name="timed" Header="Timed Mode" Margin="6,119,6,371"></toolbox:ToggleSwitch>

            <Slider Height="84" HorizontalAlignment="Left" Margin="12,310,0,0" Name="slider1" VerticalAlignment="Top" Width="418" Va
            <TextBlock Height="30" HorizontalAlignment="Left" Margin="12,274,0,0" Name="Difficulty" Text="Difficulty Level" Vertical
        </Grid>
    </Grid>
</phone:PhoneApplicationPage>
```

Above code gives us an UI interface like this.

(C) Copyright Nokia 2013. All rights reserved.

Code Behind

Instead of having a save and load button on the UI, it is more intuitive to simply load and save preferences when the user navigates TO and FROM the page. Hence in the MainPage.xaml.cs Class, we override the `OnNavigatedTo()` and `OnNavigatedFrom()` methods.

```
namespace Pref
{
    public partial class MainPage : PhoneApplicationPage
    {
        private IsolatedStorageSettings settings;
        // Constructor
        public MainPage()
        {
            InitializeComponent();
            settings = IsolatedStorageSettings.ApplicationSettings;
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            System.Diagnostics.Debug.WriteLine("into the app");
            try
            {
                System.Diagnostics.Debug.WriteLine("Retrieving values");
                gameMusic.IsChecked = (bool)settings["gamemusic"];
                timed.IsChecked = (bool)settings["timed"];
                slider1.Value = ( Int16)settings["diff"];

            }
            catch(KeyNotFoundException ex)
            {
                System.Diagnostics.Debug.WriteLine("First Time using the app");
                settings.Add("timed", false);
                settings.Add("gamemusic", false);
                settings.Add("diff", 1);
                settings.Save();

            }

        }
        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            System.Diagnostics.Debug.WriteLine("Exiting, so save now");
            settings["timed"] = timed.IsChecked;
            settings["diff"] = (Int16) slider1.Value;
            settings["gamemusic"] = gameMusic.IsChecked;
            settings.Save();
        }
    }
}
```

So as you can observe, I am accessing the values in the `IsolatedStorageSettings` in the form of Key Value pairs. Remember to cast the values when retrieving or saving to this file. It is also reminded that `Save()` method needs to be called to store or flush these values. If a key is non-existent in the `IsolatedStorageSettings`, then it may raise `KeyNotFoundException`.

Summary

This article describes one of the fundamental tasks in application development - saving and loading user preferences. This is can be extended to many minimal storage use-cases such as top scores in games.

Further reading

(C) Copyright Nokia 2013. All rights reserved.

1. How to: Store and Retrieve Application Settings Using Isolated Storage
2. IsolatedStorageSettings class

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.