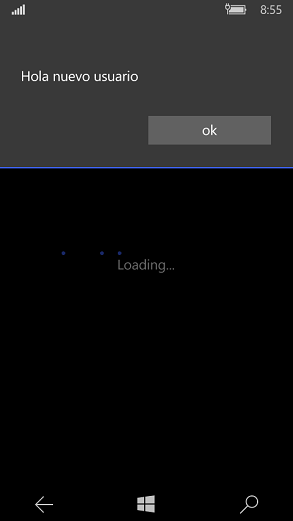
# Persistir “clave-valor”



El siguiente ejercicio nos permite reconocer si es la primera vez que un usuario entra a nuestra aplicación. Esto es muy útil para tener un buen contacto con el usuario y avisar sobre el funcionamiento de la aplicación (como guía para principiantes, avisar nuevas funcionalidades, etc..).

He creado un nuevo proyecto en blanco llamado “ISO” para realizar todas las actividades de este documento.

El código que realizaremos para esta primera actividad guardará un “true” si es nuevo usuario y le mostrará un mensaje de bienvenida, cuando sea la segunda vez ya no le mostrará el mensaje.

Para comenzar vamos a agregar en App.xaml.cs el siguiente código:

using System.IO.IsolatedStorage; //IsolatedStorage me permite persitir datos en la aplicación

namespace ISO

{

public partial class App : Application

{...

public App()

{

.. .

//Comprobamos si no hay ningún valor con la clave nuevousuario

if (!IsolatedStorageSettings.ApplicationSettings.Contains("nuevousuario"))

{ //guardo el valor true para nuevousuario

IsolatedStorageSettings.ApplicationSettings["nuevousuario"] = true;

}

}}}

En mainpage.cs

...

using System.IO.IsolatedStorage;

namespace ISO

{

public partial class MainPage : PhoneApplicationPage

{

// Constructor

public MainPage()

{

InitializeComponent();

revisarNuevoUsuario();

}

public void revisarNuevoUsuario()

{

//Buscamos si el dato persistido como "nuevousuario" es verdadero

if ((bool)IsolatedStorageSettings.ApplicationSettings["nuevousuario"])

{

//Sis es verdadero mostramos un mensaje, y de paso confirmamos que ya no es un nuevo usuario

MessageBox.Show("Hola nuevo usuario");

IsolatedStorageSettings.ApplicationSettings["nuevousuario"] = false;

}

}

# Persistir más datos

En apps siempre se recomienda almacenar pocos datos (solo los datos que son de configuración o algunos pocos datos que sabemos que no van a aumentar drásticamente), de tal manera que nuestra app pueda mantenerse siempre liviana y accesible. Para volúmenes de datos amplios se recomienda la utilización de Web Services.

Para el siguiente ejemplo guardaremos los datos de una lista de actividades:

Para ello vamos a crear una clase nueva llamada actividad.cs

using System.Data.Linq; //Esta implementación funciona en WP7/WP8.0/Windows Phone Silverlight, para versiones superiores hay alternativas de terceros (buscar en internet)

using System.Data.Linq.Mapping;

namespace ISO

{

//Agrego una clase actividad

[Table]

public class actividad

{

[Column(IsPrimaryKey = true)]

public string id { get; set; }

[Column]

public string descripcion { get; set; }

//Los [] sirven para mapear la tabla (sirven para cuando la clase se "convierta" en tabla.

}

//Voy a crear un clase que herede DataContext (permite trabajar con una base de datos )

public class actividadDC: DataContext

{

//Voy a crear la tabla objeto llamada oActividad (esto permitirá convertir la clase en una tabla)

public Table<actividad> oActividad;

//Agrego la cadena de conexión

private actividadDC(string cadenaConexion) : base(cadenaConexion) { }

//Inicialmente el DataContext debe ir vacío

static actividadDC dc = null;

//Agrego un acceso al DataContext

public static actividadDC accesoDC

{

get

{

//Si es la primera vez que inicia voy a crear la conexión, si no existe la Base de Datos voy a crear una:

if (dc == null)

{

dc = new actividadDC("isostore:/LugarBD.sdf");

if (!dc.DatabaseExists())

{

dc.CreateDatabase();

}

}

return dc;

}

}

}

}

En mainPage.xaml.cs

...

public partial class MainPage : PhoneApplicationPage

{

// Constructor

public MainPage()

{

InitializeComponent();

revisarNuevoUsuario();

cargarDatos();

}

public void revisarNuevoUsuario()

{

//Buscamos si el dato persistido como "nuevousuario" es verdadero

if ((bool)IsolatedStorageSettings.ApplicationSettings["nuevousuario"])

{

//Sis es verdadero mostramos un mensaje, y de paso confirmamos que ya no es un nuevo usuario

MessageBox.Show("Hola nuevo usuario");

IsolatedStorageSettings.ApplicationSettings["nuevousuario"] = false;

}

}

public void cargarDatos()

{

//para traer los datos realizo una consulta (usando LINQ)

var query = from actividad in actividadDC.accesoDC.oActividad

select actividad.descripcion;

//al resultado de la consulta la muestro en un listbox

LstActividades.ItemsSource = query;

}

//Desde el evento click del botón BtnAgregar

private void BtnAgregar\_Click(object sender, RoutedEventArgs e)

{

//Creo un nuevo ID

Guid guid = Guid.NewGuid();

string miGuid = guid.ToString();

//Creo una instancia de actividad

actividad nuevaActividad = new actividad();

nuevaActividad.descripcion = TextActividad.Text;

nuevaActividad.id = miGuid;

//Guardo en la BD la nueva actividad

actividadDC.accesoDC.oActividad.InsertOnSubmit(nuevaActividad);

actividadDC.accesoDC.SubmitChanges();

//Alternativa: una forma de guardar una clave/dato

IsolatedStorageSettings.ApplicationSettings["datoguardado"] = true;

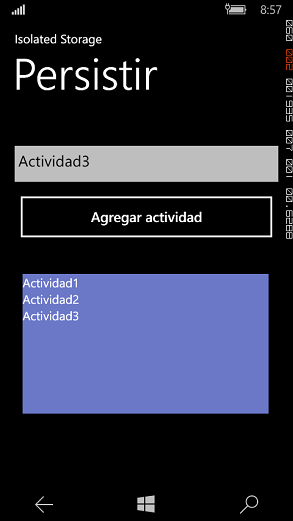
//cargo nuevamente los datos

cargarDatos();

}

}

}



Para ver todo este código desde la solución original ver en:

<https://github.com/jrodrigopuca/IsolatedStorageWP>

Para ver un ejemplo más real de Isolated Storage, recomiendo ver el código de: <https://rgeeks.wordpress.com/2013/05/06/taller-de-windows-phone/>