The Use of Microsoft Technologies for the Benefit of the Community

Eyal Carmel  (eyal.carmel@microsoft.com)
Prof. Yehuda Afek  (afek@tau.ac.il)
Mr. Hillel Avni  (hillel.avni@gmail.com)
Technologies - Windows Phone 8

1. Power/sleep
2. Volume up and volume down
3. Camera
4. Back
5. Start
6. Search
Technologies - Windows Phone 8

- 4.5" PureMotion HD+ IPS LCD
- 768x1280 resolution
- 332 ppi

- 32 GB storage
- 1024 MB RAM

- 8.7 MP
- 1080p HD video
- Front-facing camera
- LED flash

- 1.5 GHz MSM8960
- Dual core

- Autofocus
- Carl Zeiss lens
- Optical Image Stabilization
Windows Phone 8 Programming APIs

- **WP8.0 Games** DirectX/Direct 3D & C++
- **WP8.0 XAML & C#/VB with Direct3D Graphics** + C++
- **WP8.0 XAML & C#/VB** + C++
- **WP7.1 XNA & C#/VB**
- **WP7.1 XAML & C#/VB**

**Windows Phone 8 supports**

- Managed app dev using the WP7.1, WP8.0 .NET and Windows Phone Runtime APIs
- Native app dev using Windows Runtime and Win32
- Games dev using the WP7.1 XNA framework
- Games dev using Direct3D or DirectX

**.NET API for Windows Phone**

- Managed

**Windows Runtime**

- Managed & Native

**Win32 & COM**

- Native
MVVM (Model – View – ViewModel)

Architectural pattern used in software engineering

Facilitates a clear separation of the development of the graphical user interface from the development of the business logic or back end logic known as the model (also known as the data model to distinguish it from the view model)

- **View**: The UI controls
- **ViewModel**: Adapter between View and Model
- **Model**: The data. Database / XML / etc.
MVVM (Model – View – ViewModel)

Architectural pattern used in software engineering

Facilitates a clear separation of the development of the graphical user interface from the development of the business logic or back end logic known as the model (also known as the data model to distinguish it from the view model).
MVVM (Model – View – ViewModel)

- **View**: The UI controls
- **ViewModel**: Adapter between View and Model
- **Model**: The data. Database / XML / etc.
- **XAML**: Properties, Commands, View Logic
- **Data, Service Proxies**: Connections for data retrieval
MVVM (Model – View – ViewModel)

View
The UI controls

ViewModel
Adapter between View and Model

Properties, Commands, View Logic

Model
The data. Database / XML / etc.

Data, Service Proxies
MVVM (Model – View – ViewModel)
“Hello World”

- Visual Studio
- XAML
- C#
**One Time binding** - sends data from the source to the target; however, it does this only when the application is started, as a result, does not listen for change notifications in the source.

**One Way binding** - the data flows from the source to the target each time a change is made on the source.

**Two Way binding** - sends the source data to the target, and if there are changes in the target property's value, those will be sent back to the source.
Demo

Data Binding
NotifyPropertyChanged

For changes to the source object to propagate to the target, the source must implement the INotifyPropertyChanged interface. INotifyPropertyChanged has the PropertyChanged event. This event tells the binding engine that the source has changed so that the binding engine can update the target value.
// Create a class that implements INotifyPropertyChanged.
public class MyColors : INotifyPropertyChanged
{
    private SolidColorBrush _Brush1;
    // Declare the PropertyChanged event.
    public event PropertyChangedEventHandler PropertyChanged;
    // Create the property that will be the source of the binding.
    public SolidColorBrush Brush1
    {
        get { return _Brush1; }
        set
        {
            _Brush1 = value;
            // Call NotifyPropertyChanged when the source property
            // is updated.
            NotifyPropertyChanged("Brush1");
        }
    }
    // NotifyPropertyChanged will raise the PropertyChanged event,
    // passing the source property that is being updated.
    public void NotifyPropertyChanged(string propertyName)
    {
        if (PropertyChanged != null)
        {
            PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
Binding

**One Time binding** - sends data from the source to the target; however, it does this only when the application is started, as a result, does not listen for change notifications in the source

**OneWay binding** - the data flows from the source to the target each time a change is made on the source

**TwoWay binding** - sends the source data to the target, and if there are changes in the target property's value, those will be sent back to the source
Demo

Panorama

Pivot
Building the UI

Grid layout: including RowDefinitions and ColumnDefinitions

StackPanel layout

- Margins: to move elements away from the left-side, top-side, right-side or bottom-side
- HorizontalAlignment and VerticalAlignment attributes
Grid

Sizing

- **Fixed** - Fixed size
- **Auto** - Takes as much space as needed by the contained control
- **Star(*)** Takes as much space as available (after filling all auto and fixed sized columns), proportionally divided over all star-sized columns

Example:

```
400 – 40 – 20 = 340
340/3 = 113
340*2/3 = 226
```
Grid

<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
    <Grid.RowDefinitions>
        <RowDefinition Height="1*"/>
        <RowDefinition Height="2*"/>
        <RowDefinition Height="3*"/>
    </Grid.RowDefinitions>

    <Rectangle Fill="Red" Grid.Row="0" />
    <Rectangle Fill="Blue" Grid.Row="1" />
    <Rectangle Fill="Green" Grid.Row="2" />
</Grid>
Demo

Grid & StackPanel
Application Deactivation and Reactivation

Not a desktop application!!!

- Save battery life
- Ensure a good end-user experience

- At any given time, only one application is active in the foreground !!!
- Users many deactivate their applications and reactivate them later

- Applications have to deal with being activated and deactivated !!!
Windows Phone Application Lifecycle

- Windows Phone apps transition between different application states
  - Apps are launched from Start Screen icon, apps menu or from deep link
  - User may close apps
  - The OS will suspend your app if it loses focus. Suspended apps may be tombstoned
  - Apps may be reactivated from a suspended state
- When the user starts a new instance of your app, any suspended instance is discarded
  - In Windows Phone 8.0, you can enable Fast Application Resume to relaunch the suspended instance
From Dormant to Tombstoned

- An application will be held dormant in memory alongside other applications
  - If the operating system becomes short of memory it will discard the cached state of the oldest dormant application
  - This process is called “Tombstoning”

- The page navigation history and a special cache called the state dictionaries are maintained for a tombstoned application

- When a dormant application is resumed the application resumes running just where it left off

- When a tombstoned application is resumed, it restarts at the correct page but all application state has been lost – you must reload it

- An application can determine which state it is being activated from
Launchers and Choosers

• A Chooser is used when an application wants to perform a task and get a response from the user
  • Select a contact or image

• On completion of the choice the application is resumed

• A Launcher called when application wants to perform a task
  • Visit a web page or place a phone call
  • The user can return to the application via the navigation stack

• In either case the application is made dormant/tombstoned while the task completes
## Chooser Tasks

<table>
<thead>
<tr>
<th>Action</th>
<th>Chooser Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the Wallet application and add an item <strong>NEW</strong></td>
<td>AddWalletItemTask</td>
</tr>
<tr>
<td>Search contacts to return addresses, phone numbers and email addresses</td>
<td>AddressChooserTask, PhoneNumberChooserTask, EmailAddressChooserTask</td>
</tr>
<tr>
<td>Select a picture from the media store</td>
<td>PhotoChooserTask</td>
</tr>
<tr>
<td>Capture a picture using the camera</td>
<td>CameraCaptureTask</td>
</tr>
<tr>
<td>Invite players to a multi-game session</td>
<td>GameInviteTask</td>
</tr>
<tr>
<td>Save a new contact</td>
<td>SaveContactTask</td>
</tr>
<tr>
<td>Save email addresses or phone numbers</td>
<td>SaveEmailAddressTask, SavePhoneNumberTask</td>
</tr>
<tr>
<td>Save ringtones</td>
<td>SaveRingtoneTask</td>
</tr>
</tbody>
</table>
## Launcher Tasks

<table>
<thead>
<tr>
<th>Action</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Bing maps and directions</td>
<td>BingMapsTask, BingMapsDirectionsTask</td>
</tr>
<tr>
<td>Launch the network connections settings dialog</td>
<td>ConnectionSettingsTask</td>
</tr>
<tr>
<td>Send an email</td>
<td>EmailComposeTask</td>
</tr>
<tr>
<td>Show maps and directions</td>
<td>MapsTask, MapsDirectionsTask</td>
</tr>
<tr>
<td>Download and update offline maps</td>
<td>MapDownloaderTask, MapUpdaterTask</td>
</tr>
<tr>
<td>Search the Marketplace and find applications</td>
<td>MarketplaceSearchTask, MarketPlaceHubTask</td>
</tr>
<tr>
<td>Show Marketplace App Details and Review Apps</td>
<td>MarketplaceDetailTask, MarketPlaceReviewTask</td>
</tr>
<tr>
<td>Play media</td>
<td>MediaPlayerLauncher</td>
</tr>
<tr>
<td>Place a phone call</td>
<td>PhoneCallTask</td>
</tr>
<tr>
<td>Create an Appointment</td>
<td>SaveAppointmentTask</td>
</tr>
<tr>
<td>Share a link, status or photo/video on a social network</td>
<td>ShareLinkTask, ShareStatusTask, ShareMediaTask</td>
</tr>
<tr>
<td>Send an SMS message</td>
<td>SMSComposeTask</td>
</tr>
<tr>
<td>Start a search using Bing</td>
<td>SearchTask</td>
</tr>
<tr>
<td>Open a web page</td>
<td>WebBrowserTask</td>
</tr>
</tbody>
</table>
## Launching Built-in Apps

Use `LaunchUriAsync` to launch many of the built-in apps.

<table>
<thead>
<tr>
<th>URI scheme</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>http:[URL]</td>
<td>Launches the web browser and navigates to the specified URL.</td>
</tr>
<tr>
<td>mailto:[email address]</td>
<td>Launches the email app and creates a new message with the specified email address on the To line. Note that the email is not sent until the user taps send.</td>
</tr>
<tr>
<td>ms-settings-accounts:</td>
<td>Launches the Account Settings app.</td>
</tr>
<tr>
<td>ms-settings-airplanemode:</td>
<td>Launches the Airplane Mode Settings app.</td>
</tr>
<tr>
<td>ms-settings-bluetooth:</td>
<td>Launches the Bluetooth Settings app.</td>
</tr>
<tr>
<td>ms-settings-cellular:</td>
<td>Launches the Cellular Settings app.</td>
</tr>
<tr>
<td>ms-settings-emailandaccounts:</td>
<td>Launches the email and accounts settings app.</td>
</tr>
<tr>
<td>ms-settings-location:</td>
<td>Launches the Location Settings app.</td>
</tr>
<tr>
<td>ms-settings-lock:</td>
<td>Launches the Lock Screen settings app.</td>
</tr>
<tr>
<td>ms-settings-wifi:</td>
<td>Launches the Wi-Fi Settings app.</td>
</tr>
</tbody>
</table>
PeriodicTask Agents

- A PeriodicTask Agent runs every now and then
  - Typically every 30 minutes or so, depending on loading on the phone
- It is intended to perform a task that should be performed regularly and complete quickly
  - The agent is allowed to run for 25 seconds or so
  - Memory usage allowed $\leq 6$ MB
  - Unscheduled after two consecutive crashes
<table>
<thead>
<tr>
<th>Sensor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td>Detects acceleration along three axes (x, y, and z).</td>
</tr>
<tr>
<td>Inclinometer</td>
<td>Detects angle of incline along three axes (pitch, roll, and yaw).</td>
</tr>
<tr>
<td>Gyrometer</td>
<td>Detects angular velocity along three axes.</td>
</tr>
<tr>
<td>Compass</td>
<td>Detects heading in degrees relative to magnetic north (and due north when integrated with onboard GPS).</td>
</tr>
<tr>
<td>Light</td>
<td>Detects ambient lighting level in lumens.</td>
</tr>
<tr>
<td><strong>Orientation</strong></td>
<td>Combines the data from the accelerometer, compass, and gyrometer sensors to provide smoother and more sensitive rotation data than can be obtained from any of the sensors alone.</td>
</tr>
<tr>
<td>Simple Orientation</td>
<td>Uses the accelerometer to obtain device orientation as a rotation into one of four quadrants, or face-up, or face-down.</td>
</tr>
</tbody>
</table>

Local Data Storage: Overview

- Package Manager installs all app files into the Installation Folder
  - Read-only access from app
  - Read-only reference database

- Apps store data in Local Folder
  - Settings and properties in the app dictionary
  - Unstructured data in Isolated Storage files
  - Structured data in database files
## Different Methods For Addressing Storage Locations

<table>
<thead>
<tr>
<th>File Type/ API</th>
<th>Installation Folder</th>
<th>Local Folder</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Database data context</td>
<td>appdata:/</td>
<td>isostore:/</td>
<td><code>MyDataContext db = new MyDataContext(&quot;isostore:/mydb.sdf&quot;)</code></td>
</tr>
<tr>
<td>Files access using WP7.1 Isolated Storage API</td>
<td>not supported</td>
<td>StorageFile and StorageFolder APIs</td>
<td><code>var isf = IsolatedStorageFile.GetUserStoreForApplication()</code></td>
</tr>
<tr>
<td>File access using Windows.Storage API via URIs</td>
<td>ms-appx:///</td>
<td>ms-appdata:///local/</td>
<td><code>var file = await Windows.Storage.File.GetFileFromApplicationUriAsync(new Uri(&quot;ms-appdata:///local/AppConfigSettings.xml&quot;));</code></td>
</tr>
</tbody>
</table>
Saving Data – Using StorageFolder

```csharp
using System.IO;
using Windows.Storage;
...

private async void saveToLocalFolderAsync(string message)
{
    // Get a reference to the Local Folder
    StorageFolder localFolder = ApplicationData.Current.LocalFolder;

    // Create the file in the local folder, or if it already exists, just open it
    StorageFile storageFile = await localFolder.CreateFileAsync("Myfile.store",
        CreationCollisionOption.OpenIfExists);

    Stream writeStream = await storageFile.OpenStreamForWriteAsync();
    using (StreamWriter writer = new StreamWriter(writeStream))
    {
        await writer.WriteAsync(logData);
    }
}
```
using System.IO;
using Windows.Storage;
...

private async string loadStringAsync()
{
    string theData = string.Empty;

    // Get a reference to the file in the Local Folder
    StorageFolder localFolder = ApplicationData.Current.LocalFolder;
    StorageFile storageFile = await localFolder.GetFileAsync("Myfile.store");

    // Open it and read the contents
    Stream readStream = await storageFile.OpenStreamForReadAsync();
    using (StreamReader reader = new StreamReader(readStream))
    {
        theData = await reader.ReadToEndAsync();
    }

    return theData;
}
Three Kinds of Notifications

- **Raw**
  - Notification message content is application-specific
  - Delivered directly to app only if it is running

- **Toast**
  - Specific XML schema
  - Content delivered to app if it is running
  - If app is not running, system displays Toast popup using notification message content

- **Tile**
  - Specific XML schema
  - Never delivered to app
Push Notification Data Flow

1. Push endpoint is established. URI is created for the endpoint.
2. URI to the service:
   "https://notify.live.com/throttled/thirdparty/BLGDA/AFRQHgiiMIVNYyFIDYkHEZc_AgnNgzQwAwAAMaQMDAmMDAwMDAwMDAwMDAwMDA"
3. HTTP POST the message
4. Send PN Message

Push enabled applications
Notifications service
Third-party service
MPNS: Microsoft hosted server
Setup the Push Notification Channel

// Try to find an existing channel
HttpNotificationChannel httpChannel = HttpNotificationChannel.Find("MyChannel");
if (null == httpChannel)
{
    httpChannel = new HttpNotificationChannel("MyChannel");
    // handle Uri notification events
    httpChannel.ChannelUriUpdated +=
        new EventHandler<NotificationChannelUriEventArgs>(httpChannel_ChannelUriUpdated);
    httpChannel.Open();
}
else
{
    // the channel already exists. httpChannel.ChannelUri contains the device’s
    // unique locator
}
// handle error events
httpChannel.ErrorOccurred +=
    new EventHandler<NotificationChannelErrorEventArgs>(httpChannel_ErrorOccurred);
Send a Push Notification to a Client Phone

// create a payload for a toast notification
string msg =
   "<?xml version="1.0" encoding="utf-8"?>" +
   "<wp:Notification xmlns:wp="WPNotification">" +
   "<wp:Toast>" +
   "<wp:Text1><string/></wp:Text1>" +
   "<wp:Text2><string/></wp:Text2>" +
   "</wp:Toast>" +
   "</wp:Notification>";
byte[] msgBytes = new UTF8Encoding().GetBytes(msg);
// create a web request that identifies the payload as a toast notification
WebRequest request = (HttpWebRequest)WebRequest.Create(deviceUri);
request.ContentType = "text/xml";
request.ContentLength = msg.Length;
request.Headers["X-MessageID"] = Guid.NewGuid().ToString();
request.Headers["X-WindowsPhone-Target"] = "toast";
request.Headers["X-NotificationClass"] = "2";
// post the payload
Stream requestStream = request.GetRequestStream();
requestStream.Write(msgBytes, 0, msgBytes.Length);
requestStream.Close();
Sources

GPS: +Accuracy, -Power, -Speed, -Indoors

Cell Towers: -Accuracy, +Power, +Speed, -Wilderness
Controlling the Sources the Geolocation Service Uses

• You can’t!

• You can set the `DesiredAccuracy` property of the `Geolocator` object:
  • `PositionAccuracy.High` – if you want the most accurate data available, but at the cost of increased battery usage, network bandwidth and possibly monetary charges from wireless network operators. Often this causes the GPS to be activated
  • `PositionAccuracy.Default` – to optimize for power

• You can also set the `DesiredAccuracyInMeters` property to indicate to the Geolocation service the desired accuracy of any results

• However, the Geolocation service determines the best location data to provide to the application
private async void OneShotLocation_Click(object sender, RoutedEventArgs e)
{
    Geolocator geolocator = new Geolocator();
geolocator.DesiredAccuracyInMeters = 50;
    try
    {
        Geoposition geoposition = await geolocator.GetGeopositionAsync(
            maximumAge: TimeSpan.FromMinutes(5),
            timeout: TimeSpan.FromSeconds(10));
        LatitudeTextBlock.Text = geoposition.Coordinate.Latitude.ToString("0.00");
        LongitudeTextBlock.Text = geoposition.Coordinate.Longitude.ToString("0.00");
    }
    catch (UnauthorizedAccessException)
    {
        // the app does not have the right capability or the location master switch is off
        StatusTextBlock.Text = "location is disabled in phone settings.";
    }
}
Where to Start?

• SW Downloads
  • www.dreamspark.com/Student/Default.aspx
  • http://e5.onthehub.com/WebStore/ProductsByMajorVersionList.aspx?ws=23e93290-629b-e011-969d-0030487d8897&vsro=8&JSEnabled=1

• Resources
  • http://www.microsoft.com/israel/4steps/
  • http://channel9.msdn.com/Series/Windows-Phone-8-Development-for-Absolute-Beginners