Mobile Computing Notes

BCA VI

UNIT – II

Jiwaji University

Year – 2019-20

An Introduction to mobile computing

Mobile Computing is a technology that allows transmission of data, voice and video via a computer or any other wireless enabled device without having to be connected to a fixed physical link. The main concept involves −

- Mobile communication
- Mobile hardware
- Mobile software

Mobile communication

The mobile communication in this case, refers to the infrastructure put in place to ensure that seamless and reliable communication goes on. These would include devices such as protocols, services, bandwidth, and portals necessary to facilitate and support the stated services. The data format is also defined at this stage. This ensures that there is no collision with other existing systems which offer the same service.

Since the media is unguided/unbounded, the overlaying infrastructure is basically radio wave-oriented. That is, the signals are carried over the air to intended devices that are capable of receiving and sending similar kinds of signals.

Mobile Hardware

Mobile hardware includes mobile devices or device components that receive or access the service of mobility. They would range from portable laptops, smartphones, tablet Pc's, Personal Digital Assistants.

These devices will have a receptor medium that is capable of sensing and receiving signals. These devices are configured to operate in full- duplex, whereby they are capable of sending and receiving signals at the same time. They don't have to wait until one device has finished communicating for the other device to initiate communications.

Above mentioned devices use an existing and established network to operate on. In most cases, it would be a wireless network.

Mobile software

Mobile software is the actual program that runs on the mobile hardware. It deals with the characteristics and requirements of mobile applications. This is the engine of the mobile device. In other terms, it is the operating system of the appliance. It's the essential component that operates the mobile device.

Mobile Application Programming

Mobile app development is the act or process by which a mobile app is developed for mobile devices, such as personal digital assistants, enterprise digital assistantsor mobile phones. These applications can be pre-installed on phones during manufacturing platforms, or delivered as web applications using server-side or client-side processing (e.g., JavaScript) to provide an "application-like" experience within a Web browser. Application software developers also must consider a long array of screen sizes, hardware specifications, and configurations because of intense competition in mobile software and changes within each of the platforms. Mobile app development has been steadily growing, in revenues and jobs created. A 2013 analyst report estimates there are 529,000 direct *app economy* jobs within the EU 28 members, 60% of which are mobile app developers.

As part of the development process, mobile user interface (UI) design is also essential in the creation of mobile apps. Mobile UI considers constraints, contexts, screen, input, and mobility as outlines for design. The user is often the focus of interaction with their device, and the interface entails components of both hardware and software. User input allows for the users to manipulate a system, and device's output allows the system to indicate the effects of the users' manipulation. Mobile UI design constraints include limited attention and form factors, such as a mobile device's screen size for a user's hand(s). Mobile UI contexts signal cues from user activity, such as location and scheduling that can be shown from user interactions within a mobile app. Overall, mobile UI design's goal is mainly for an understandable, user-friendly interface. The UI of mobile apps should: consider users' limited attention, minimize keystrokes, and be task-oriented with a minimum set of functions. This functionality is supported by mobile enterprise application platforms or integrated development environments (IDEs).

What is a Mobile Operating System (Mobile OS)?

Much like the Linux or Windows operating system controls your desktop or laptop computer, a mobile operating system is the software platform on top of which other programs can run on mobile devices. The operating system is responsible for determining the functions and features available on your device, such as thumb wheel, keyboards, WAP, synchronization with applications, email, text messaging and more. The mobile OS will also determine which third-party applications (mobile apps) can be used on your device.

Types of Mobile Operating Systems

When you purchase a mobile device the manufacturer will have chosen the operating system for that specific device. Often, you will want to learn about the mobile operating system before you purchase a device to ensure compatibility and support for the mobile applications you want to use.

9 Popular Mobile Operating Systems

1. Android OS (Google Inc.)

The Android mobile operating system is Google's open and free software stack that includes an operating system, middleware and also key applications for use on mobile devices, including smartphones. Updates for the open source Android mobile operating system have been developed under "dessert-inspired" version names (Cupcake, Donut, Eclair, Gingerbread, Honeycomb, Ice Cream Sandwich) with each new version arriving in alphabetical order with new enhancements and improvements.

2. Bada (Samsung Electronics)

Bada is a proprietary Samsung mobile OS that was first launched in 2010. The Samsung Wave was the first smartphone to use this mobile OS. Bada provides mobile features such as multipoint-touch, 3D graphics and of course, application downloads and installation.

3. BlackBerry OS (Research In Motion)

The BlackBerry OS is a proprietary mobile operating system developed by Research In Motion for use on the company's popular BlackBerry handheld devices. The BlackBerry platform is popular with corporate users as it offers synchronization with Microsoft Exchange, Lotus Domino, Novell GroupWise email and other business software, when used with the BlackBerry Enterprise Server.

4. iPhone OS / iOS (Apple)

Apple's iPhone OS was originally developed for use on its iPhone devices. Now, the mobile operating system is referred to as iOS and is supported on a number of Apple devices including the iPhone, iPad, iPad 2 and iPod Touch. The iOS mobile operating system is available only on Apple's own manufactured devices as the company does not license the OS for third-party hardware. Apple iOS is derived from Apple's Mac OS X operating system.

5. MeeGo OS (Nokia and Intel)

A joint open source mobile operating system which is the result of merging two products based on open source technologies: Maemo (Nokia) and Moblin (Intel). MeeGo is a mobile OS designed to work on a number of devices including smartphones, netbooks, tablets, in-vehicle information systems and various devices using Intel Atom and ARMv7 architectures.

6. Palm OS (Garnet OS)

The Palm OS is a proprietary mobile operating system (PDA operating system) that was originally released in 1996 on the Pilot 1000 handheld. Newer versions of the Palm OS have added support for expansion ports, new processors, external memory cards, improved security and support for ARM processors and smartphones. Palm OS 5 was extended to provide support for a broad range of screen resolutions, wireless connections and enhanced multimedia capabilities and is called Garnet OS.

7. Symbian OS (Nokia)

Symbian is a mobile operating system (OS) targeted at mobile phones that offers a high-level of integration with communication and personal information management (PIM) functionality.

Symbian OS combines middleware with wireless communications through an integrated mailbox and the integration of Java and PIM functionality (agenda and contacts). Nokia has made the Symbian platform available under an alternative, open and direct model, to work with some OEMs and the small community of platform development collaborators. Nokia does not maintain Symbian as an open source development project.

8. webOS (Palm/HP)

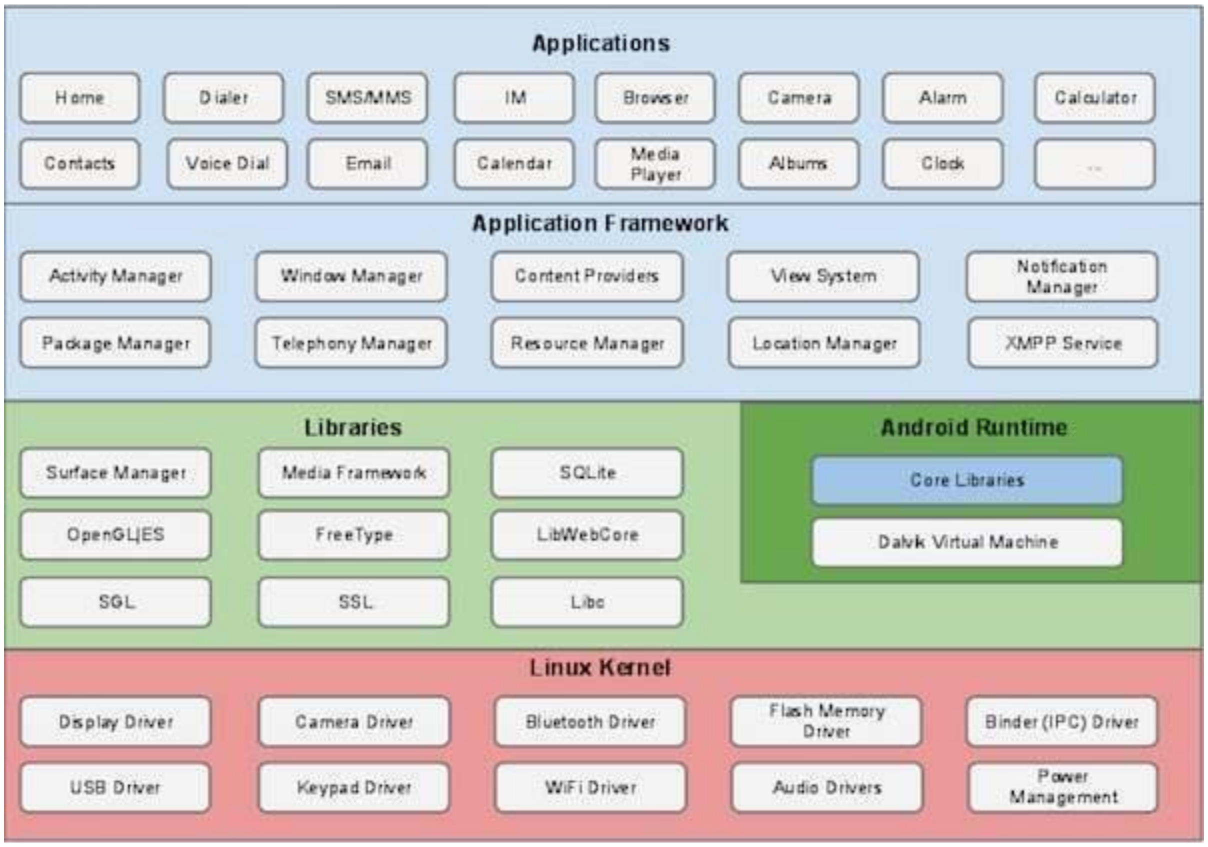
WebOS is a mobile operating system that runs on the Linux kernel. WebOS was initially developed by Palm as the successor to its Palm OS mobile operating system. It is a proprietary Mobile OS which was eventually acquired by HP and now referred to as webOS (lower-case w) in HP literature. HP uses webOS in a number of devices including several smartphones and HP TouchPads. HP has pushed its webOS into the enterprise mobile market by focusing on improving security features and management with the release of webOS 3.x. HP has also announced plans for a version of webOS to run within the Microsoft Windows operating system and to be installed on all HP desktop and notebook computers in 2012.

9. Windows Mobile (Windows Phone)

Windows Mobile is Microsoft's mobile operating system used in smartphones and mobile devices – with or without touchscreens. The Mobile OS is based on the Windows CE 5.2 kernel. In 2010 Microsoft announced a new smartphone platform called Windows Phone 7.

Android operating system

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.



Linux kernel

At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

Libraries

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

Android Libraries

This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access. A summary of some key core Android libraries available to the Android developer is as follows −

- **android.app** − Provides access to the application model and is the cornerstone of all Android applications.
- **android.content** − Facilitates content access, publishing and messaging between applications and application components.
- **android.database** − Used to access data published by content providers and includes SQLite database management classes.
- **android.opengl** − A Java interface to the OpenGL ES 3D graphics rendering API.
- **android.os** − Provides applications with access to standard operating system services including messages, system services and inter-process communication.
- **android.text** − Used to render and manipulate text on a device display.
- **android.view** − The fundamental building blocks of application user interfaces.
- **android.widget** − A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- **android.webkit** − A set of classes intended to allow web-browsing capabilities to be built into applications.

Having covered the Java-based core libraries in the Android runtime, it is now time to turn our attention to the C/C++ based libraries contained in this layer of the Android software stack.

Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called **Dalvik Virtual Machine** which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

The Android framework includes the following key services −

- **Activity Manager** − Controls all aspects of the application lifecycle and activity stack.
- **Content Providers** − Allows applications to publish and share data with other applications.
- **Resource Manager** − Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- **Notifications Manager** − Allows applications to display alerts and notifications to the user.
- **View System** − An extensible set of views used to create application user interfaces.

Applications

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games etc.

Applications

These are the basics of Android applications:

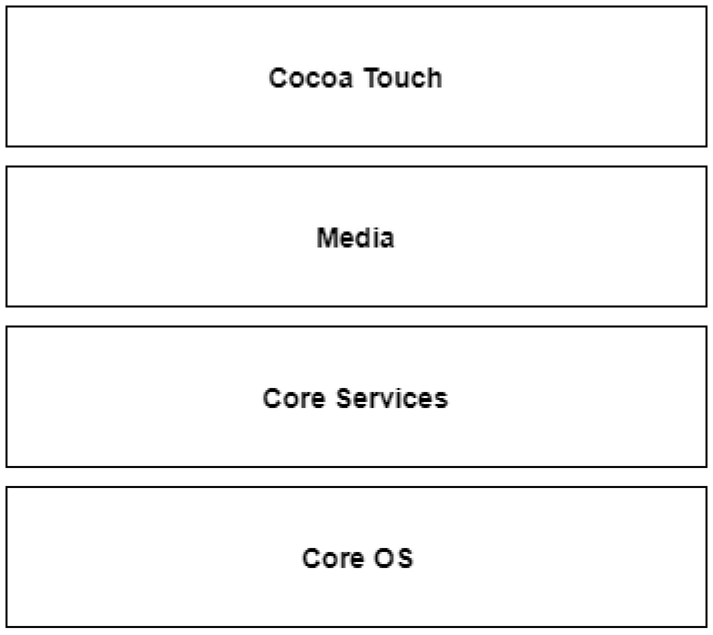
- Android applications are composed of one or more application components (activities, services, content providers, and broadcast receivers)
- Each component performs a different role in the overall application behavior, and each one can be activated individually (even by other applications)
- The manifest file must declare all components in the application and should also declare all application requirements, such as the minimum version of Android required and any hardware configurations required
- Non-code application resources (images, strings, layout files, etc.) should include alternatives for different device configurations (such as different strings for different languages)

Google, for software development and application development, had launched two competitions ADC1 and ADC2 for the most innovative applications for Android. It offered prizes of USD 10 million combined in ADC1 and 2. ADC1 was launched in January 2008 and ADC 2 was launched in May 2009. These competitions helped Google a lot in making Android better, more user friendly, advanced and interactive.

The iOS is the operating system created by Apple Inc. for mobile devices. The iOS is used in many of the mobile devices for apple such as iPhone, iPod, iPad etc. The iOS is used a lot and only lags behind Android in terms of popularity.

The iOS architecture is layered. It contains an intermediate layer between the applications and the hardware so they do not communicate directly. The lower layers in iOS provide the basic services and the higher layers provide the user interface and sophisticated graphics.

The layered architecture of iOS is given as follows:



Apple iOS Architecture

Layers in iOS Architecture

The different layers as shown in the above diagram are given as follows:

Core OS

All the iOS technologies are build on the low level features provided by the Core OS layer. These technologies include Core Bluetooth Framework, External Accessory Framework, Accelerate Framework, Security Services Framework, Local Authorisation Framework etc.

Core Services

There are many frameworks available in the cure services layer. Details about some of these are given as follows:

Cloudkit Framework

The data can be moved between the app the iCloud using the Cloudkit Framework.

Core Foundation Framework

This provides the data management and service features for the iOS apps.

Core Data Framework

The data model of the model view controller app is handled using the Core Data Framework.

Address Book Framework

The address book framework provides access to the contacts database of the user.

Core Motion Framework

All the motion based data on the device is accessed using core motion framework.

Healthkit Framework

The health related information of the user can be handled by this new framework.

Core Location Framework

This framework provides the location and heading information to the various apps.

Media

The media layer enables all the graphics, audio and video technology of the system. The different frameworks are:

UIKit Graphics

This provides support for designing images and animating the view content.

Core Graphics Framework

This provides support for 2-D vector and image based rendering and is the native drawing engine for iOS apps.

Core Animation

The Core Animation technology optimizes the animation experience of the apps.

Media Player Framework

This framework provides support for playing playlists and enables the user to use their iTunes library.

AV Kit

This provides various easy to use interfaces for video presentation.

Cocoa Touch

The cocoa touch layer provides the following frameworks:

EventKit Framework

This shows the standard system interfaces using view controllers for viewing and changing calendar related events.

GameKit Framework

This provides support for users to share their game related data online using Game center.

MapKit Framework

This provides a scrollable map which can be included into the app user interface.

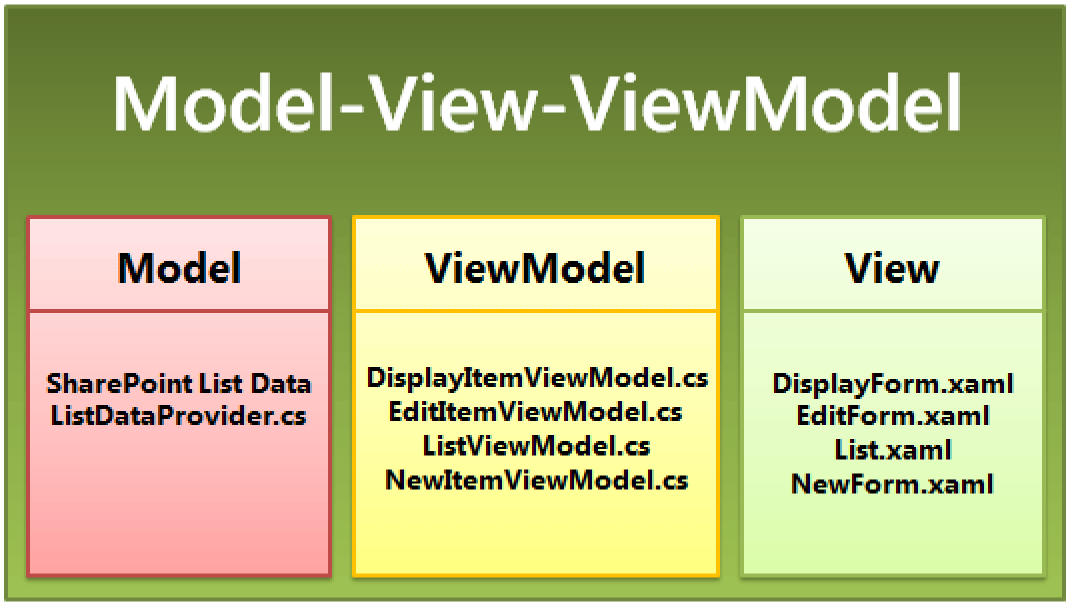
The Windows Phone SharePoint List Application template and the MVVM design pattern

The Windows Phone SharePoint List Application template generates a Visual Studio 2010 project for a Silverlight-based Windows Phone app developed according to a software design pattern known as the View-Model-ViewModel (MVVM) pattern. The MVVM pattern is a way of organizing and compartmentalizing code in a project into manageable layers, which can be independently developed, tested, and modified. It is a particularly effective development pattern for Windows Presentation Foundation (WPF) and Silverlight projects because, among other benefits, the pattern allows for the presentation layer of a given application to have a less rigid dependency on the structure of underlying data, freeing developers to adapt the presentation layer for different contexts (as, for instance, Web browsers, mobile device interfaces, or desktop applications) while retaining the same underlying data structures.

As opposed to a simpler approach of, say, writing all of your data-management code in the code-behind files associated with particular XAML files in a Silverlight application, organizing a project according to the MVVM pattern involves an additional initial investment of effort to plan and develop the necessary classes, the inheritance model, and the methods of communication between the components of the pattern. The Windows Phone SharePoint List Application template takes care of this initial configuration and development work to set up the pattern for you, allowing you to customize and extend the project to develop a functional MVVM application quickly.

The three main components or layers of the MVVM pattern are the View, the Model, and the ViewModel. In projects based on the Windows Phone SharePoint List Application template, these components are implemented by various project files, as shown in Figure 1.

Figure 1. Windows Phone SharePoint List Application files in the MVVM pattern



The following sections explain some of the details related to the implementation of these components in the Windows Phone SharePoint List Application template.

The Model component

The Model component in the MVVM pattern refers to the classes and structures used to represent the data for an application. For an app based on a SharePoint list, the list and its items serve as the underlying data. In the Windows Phone SharePoint List Application, the **ListDataProvider** class handles the standard SharePoint client object model operations for connecting to a SharePoint list; for example, creating an instance of the **ClientContext** class and setting its properties. The exact implementation details of the **ListDataProvider** class in the template depend on the options specified in the steps of the SharePoint Phone Application Wizard when you create a project based on the template.

The base class, **ListDataProviderBase** (in the Microsoft.SharePoint.Phone.Application.dll), from which the **ListDataProvider** class is derived implements a caching mechanism for SharePoint list data. When list items are retrieved from the server, they are cached by the **ListDataProvider** class in the local memory allocated to the phone app and when those items are needed in the app, the cache is checked first in order to conserve resources and reduce trips to the server.

If you want to filter the data retrieved from the SharePoint list or specify exactly what data to retrieve, you can modify the code in the **ListDataProvider** class (in the ListDataProvider.cs file). The parts of the file you would most likely modify for these purposes are the **LoadDataFromServer** method and the implementation of the static **CamlQueryBuilder** class. You can also derive your own class from the **ListDataProviderBase** class. If you do so, be sure to implement the abstract methods from the base class, **LoadData** and **LoadItem**, and also implement the **Context** property member of the base class, providing a suitable **get** accessor method.

The View component

The View component in the MVVM pattern refers to the user interface (UI) of an app. In Silverlight-based Windows Phone app, the View component is constituted by XAML files for declaring and qualifying UI elements and code-behind files associated with those XAML files that implement event handlers and other code to determine how users interact with the UI elements.

It is important to distinguish between two senses of the word "view" in the context of developing SharePoint list apps for Windows Phone. A SharePoint list is associated with one or more views, as, for example, the default All Items view for a list, or the Current Events view for a list based on the Calendar list template. These views represent ways of organizing and displaying list items in a SharePoint list. Depending on the kind of SharePoint list (and on whether custom views have been added to the list) you target for your app, the views associated with the list, such as All Tasks or Current Events, are made available for you to choose to include in your app in the SharePoint Phone Application Wizard when you create a project from the template. If you include a given view, the template generates a **PivotItem** control (contained within a **Pivot** control) to render the view of the list.

This sense of the word "view" is to be distinguished from the sense of the word as it applies to the Views in the template. In a project based on the Windows Phone SharePoint List Application template, the Views (implemented as XAML files in the Views folder of the project) refer conceptually to the View component of the MVVM pattern. That is, the Views in the project represent the presentation layer for the data (or, Model) of a given entity. In this case, the entity is either a SharePoint list or a SharePoint list item.

Although the List form (List.xaml) in the project can be said to correspond to the default view associated with a SharePoint list, the conceptual distinction between the default view of a SharePoint list and the View as represented by the List form should still be maintained, because the List form in the project doesn't necessarily map to the default view of the list on the server. If, for example, you modify the default list view on the server (by, say, specifying a given sort order or displaying certain fields and not others), the modifications will not be represented in the XAML that constitutes the List form in the project. You set the order of the items as they are shown in the List form in your app based on your choices in the SharePoint Phone Application Wizard (or based on your subsequent customizations of the List form), regardless of the order configured for the default view associated with the SharePoint list on the server.

The List form represents the View (or presentation layer) for the SharePoint list. The other three View files pertain to individual list items, and they can be said to correspond to the forms available (generally) from the list item menu for a list item in SharePoint.

- The Display form (DisplayForm.xaml) corresponds to the View Item form (DispForm.aspx) for a SharePoint list. This form presents a View for an individual item in a SharePoint list.
- The Edit form (EditForm.xaml) corresponds to the Edit Item form (EditForm.aspx) for a SharePoint list. This form presents a View for a given item as it is exposed for editing.

- The New form (NewForm.xaml) file corresponds to the New Item form (NewForm.aspx) for a SharePoint list. This form presents a View for a given item that is to be created and added to the list.

The List form is always included by default in a project based on the Windows Phone SharePoint List Application template. The XAML files for the other forms in the Views folder of the project are generated based on the list operations (New, Display, or Edit) selected in the SharePoint Phone Application Wizard.

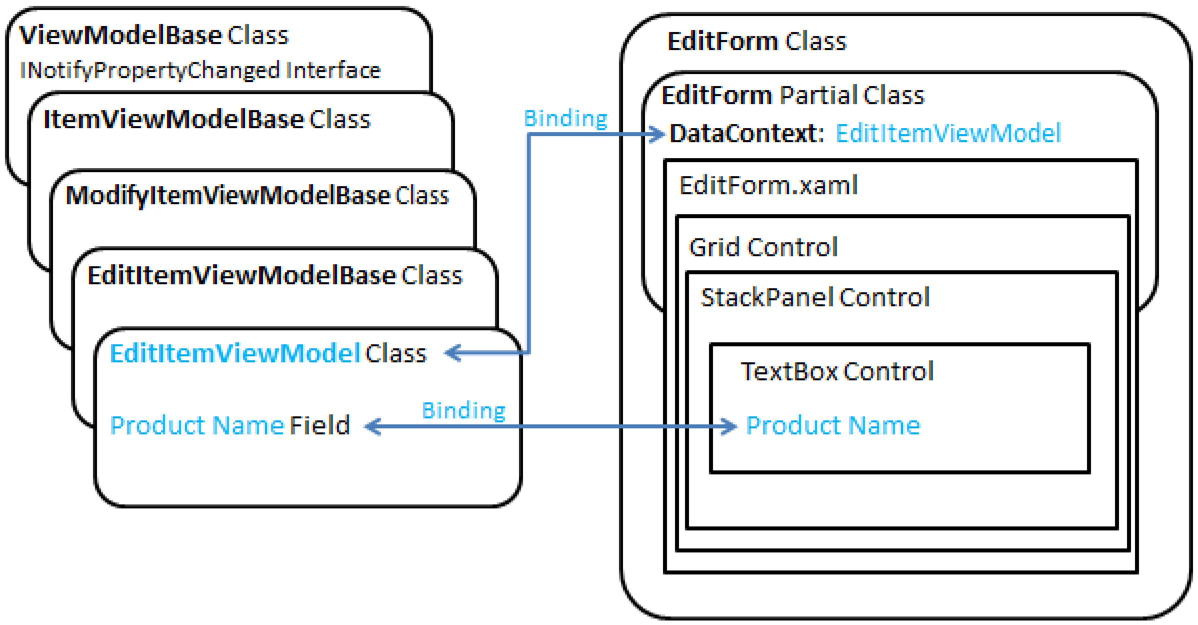
The ViewModel component

The ViewModel component in the MVVM pattern is intended to serve as a kind of broker to facilitate the interactions between the View component and the Model component, while decoupling the View component from the Model component so it is easier to change one or the other without adversely affecting the other. Strictly speaking, the ViewModel component could be considered part of the presentation layer, because it often includes logic for "shaping" underlying data for presentation in the View component. In projects based on the Windows Phone SharePoint List Application template, the ViewModels implement the code for binding SharePoint list data retrieved from the Model component (that is, from an object of the **ListDataProvider** class) to UI controls in a part of the View component (for example, the Edit form). Depending on the kind of control used to display the data from the list and the type of data (that is, whether the field type for the list item is text or numerical data or something like a SharePoint Choice field), the ViewModel first processes or converts the data such that it can be bound to a given UI control.

In particular, the ViewModel classes in the project (as, for example, the **EditItemViewModel** class) are derived from a base class, **ViewModelBase** (in the Microsoft.SharePoint.Phone.Application.dll), which implements the **INotifyPropertyChanged** interface so that the Silverlight controls constituting the user interface of the app can be updated when values in the underlying data change, and (going in the other direction) changes to the values stored in UI controls can be applied to the underlying data (if bi-directional, or "two-way", binding is configured for a control).

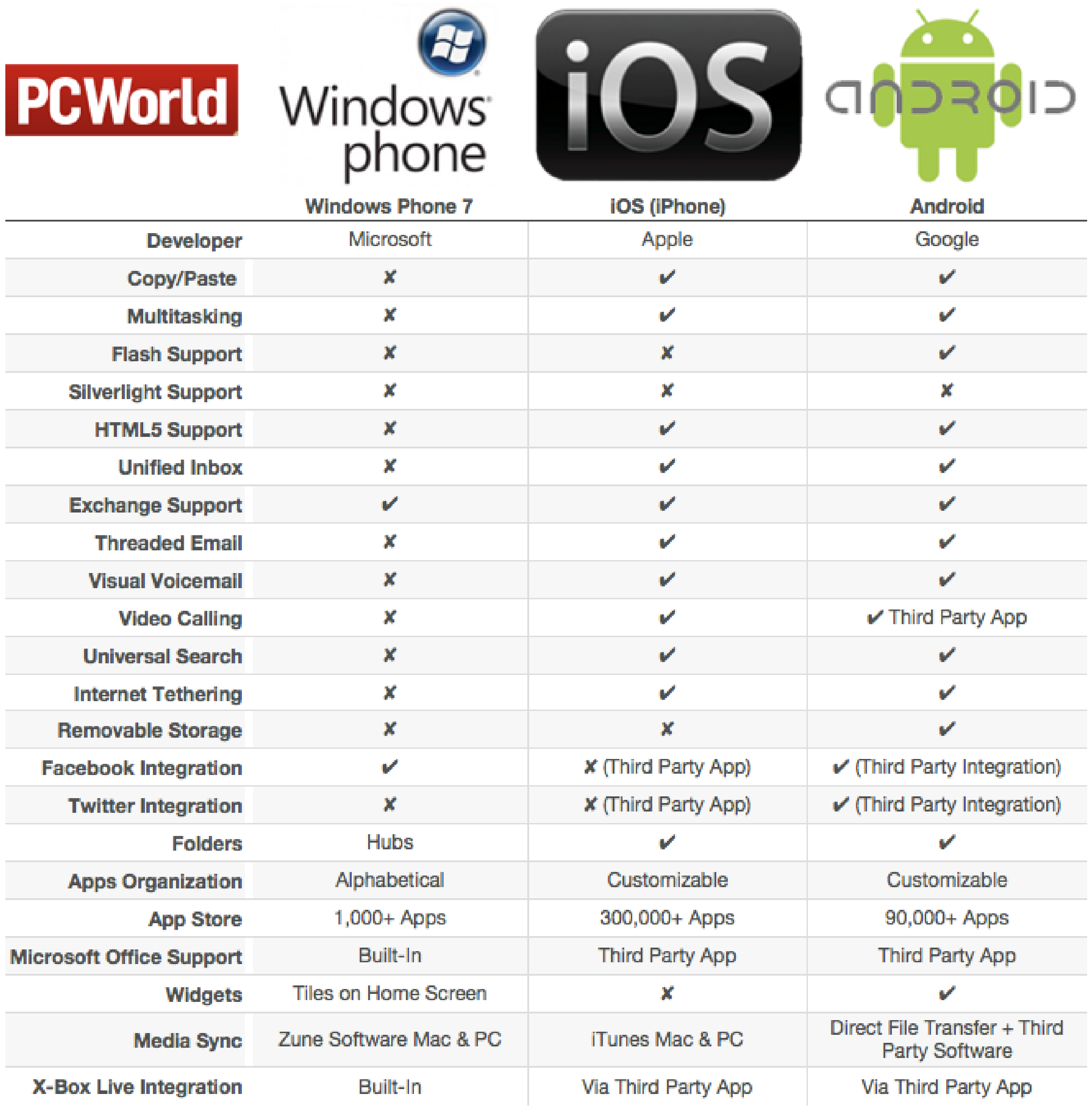
Figure 2 shows a simplified representation of the class inheritance hierarchy for the **EditItemViewModel** class and the binding for a given UI control in the Edit form with the corresponding field in the ViewModel.

Figure 2. The EditItemViewModel and EditForm classes



The **EditForm** class (which represents the View component from the MVVM pattern) is defined and implemented by two files, the EditForm.xaml file and its associated code-behind file, EditForm.xaml.cs. In the EditForm.xaml.cs file, the **EditItemViewModel** class (representing the ViewModel component of the MVVM pattern) is bound to the View in the EditForm.xaml.cs file by setting the **DataContext** property of the **EditForm** class to an object of the **EditItemViewModel** class.

Software designs based on the MVVM pattern often confine business logic and validation routines to the Model component of the pattern. In projects based on the Windows Phone SharePoint List Application template, however, some operations that are typically considered part of the Model component have been implemented in the ViewModel component to make it more convenient for developers to extend the projects, at the cost of slightly blurring the conceptual distinction between the data layer (Model) and the presentation layer (ViewModel). For example, the ViewModel classes for editing and creating list items (that is, the **EditItemViewModel** and **NewItemViewModel** classes) expose a **Validate** method that developers can override to implement validation of data entered by users. (For information on implementing data validation with these ViewModels, see How to: Implement business logic and data validation in a Windows Phone app for SharePoint..md)



	Windows Phone 7	iOS (iPhone)	Android
Developer	Microsoft	Apple	Google
Copy/Paste	✘	✔	✔
Multitasking	✘	✔	✔
Flash Support	✘	✘	✔
Silverlight Support	✘	✘	✘
HTML5 Support	✘	✔	✔
Unified Inbox	✘	✔	✔
Exchange Support	✔	✔	✔
Threaded Email	✘	✔	✔
Visual Voicemail	✘	✔	✔
Video Calling	✘	✔	✔ Third Party App
Universal Search	✘	✔	✔
Internet Tethering	✘	✔	✔
Removable Storage	✘	✘	✔
Facebook Integration	✔	✘ (Third Party App)	✔ (Third Party Integration)
Twitter Integration	✘	✘ (Third Party App)	✔ (Third Party Integration)
Folders	Hubs	✔	✔
Apps Organization	Alphabetical	Customizable	Customizable
App Store	1,000+ Apps	300,000+ Apps	90,000+ Apps
Microsoft Office Support	Built-In	Third Party App	Third Party App
Widgets	Tiles on Home Screen	✘	✔
Media Sync	Zune Software Mac & PC	iTunes Mac & PC	Direct File Transfer + Third Party Software
X-Box Live Integration	Built-In	Via Third Party App	Via Third Party App

You will be glad to know that you can start your Android application development on either of the following operating systems –

- Microsoft Windows XP or later version.

- Mac OS X 10.5.8 or later version with Intel chip.
- Linux including GNU C Library 2.7 or later.

Second point is that all the required tools to develop Android applications are freely available and can be downloaded from the Web. Following is the list of software's you will need before you start your Android application programming.

- Java JDK5 or later version
- Android Studio

Here last two components are optional and if you are working on Windows machine then these components make your life easy while doing Java based application development. So let us have a look how to proceed to set required environment.

Set-up Java Development Kit (JDK)

You can download the latest version of Java JDK from Oracle's Java site − Java SE Downloads. You will find instructions for installing JDK in downloaded files, follow the given instructions to install and configure the setup. Finally set PATH and JAVA_HOME environment variables to refer to the directory that contains **java** and **javac**, typically java_install_dir/bin and java_install_dir respectively.

If you are running Windows and installed the JDK in C:\jdk1.8.0_102, you would have to put the following line in your C:\autoexec.bat file.

```
set PATH=C:\jdk1.8.0_102\bin;%PATH%
set JAVA_HOME=C:\jdk1.8.0_102
```

Alternatively, you could also right-click on *My Computer*, select *Properties*, then *Advanced*, then *Environment Variables*. Then, you would update the PATH value and press the OK button.

On Linux, if the SDK is installed in /usr/local/jdk1.8.0_102 and you use the C shell, you would put the following code into your **.cshrc** file.

```
setenv PATH /usr/local/jdk1.8.0_102/bin:$PATH
setenv JAVA_HOME /usr/local/jdk1.8.0_102
```

Alternatively, if you use Android studio, then it will know automatically where you have installed your Java.

Android IDEs

There are so many sophisticated Technologies are available to develop android applications, the familiar technologies, which are predominantly using tools as follows

- Android Studio
- Eclipse IDE(Deprecated)

1. Universal Chargers

Apple’s Lightning connector works on all mobile Apple devices purchased after 2012. If your Apple device is older than that, its charger won’t work on your newer device. The new Lightning connector won’t work on devices other than Apple without a Lightning to Micro USB adapter. Android, meanwhile, already uses the standardised and ubiquitous Micro USB connection for its chargers.

2. More Phone Choices

Apple enthusiasts have a limited selection of phones available for upgrades, whereas Android users have several brands of phones to choose from at different price levels.

3. Removable Storage and Battery

Increasing memory capacity in an iPhone or iPad requires a costly upgrade, but many Android devices have expandable microSD card slots. Android batteries are also removable, allowing for replacement of just the battery and not the entire phone.

4. Widgets

Widgets, or self contained programs, add functionality and flexibility to Android devices. Android has far more to offer than Apple, such as Battery Widget Reborn and Circle Launcher.

5. Better Hardware

Some Android flagship phones compete well against iPhone, with better hardware. The Samsung Galaxy S7 Edge, for example, out muscles the iPhone 6S Plus with a faster processor, more RAM, increased battery capacity and better screen resolution.

6. Better Charging Options

Android’s fast charging is enticing enough without another added Android charging perk: wireless.

7. Infrared

Why would you be happy that your Android has infrared? Think wireless cross-device interaction, and more importantly, it can act as a TV remote.

8. More App Choices

With Android devices, you can shop for apps outside the Play Store at places such as Amazon.

9. Custom Keyboards

If you don't like your Apple keyboard, your options are limited, whereas Android offers alternative choices such as Swiftkey.

10. Google Play Is More User-Friendly

Both the Apple App Store and Android's Google Play offer over one million downloadable apps. However, the App Store has restrictions, such as a mandatory iTunes interface and a requirement that downloaded movies be played on Apple devices.

Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to give you a powerful, integrated environment in which to build Android applications.

ADT extends the capabilities of Eclipse to let you quickly set up new Android projects, create an application UI, add packages based on the Android Framework API, debug your applications using the Android SDK tools, and even export signed (or unsigned) .apk files in order to distribute your application.

Developing in Eclipse with ADT is highly recommended and is the fastest way to get started. With the guided project setup it provides, as well as tools integration, custom XML editors, and debug output pane, ADT gives you an incredible boost in developing Android applications.

Categories:

- Mobile and Device Development

Tags:

- android,
- Mobile,

- smartphone,
- Mobile apps,
- tablet,
- google

HideAdditional Details

Eclipse Versions:
Mars (4.5), Luna (4.4), Kepler (4.3), Juno (4.2, 3.8), Previous to Juno (<=4.1), Neon (4.6), Oxygen (4.7), Photon (4.8), 2018-09 (4.9)

Platform Support:
Windows, Mac, Linux/GTK

Organization Name:
Google, Inc.

Date Created:
Wed, 2012-05-09 10:58

Development Status:
Production/Stable

License:
Apache 2.0

Date Updated:
Mon, 2019-02-04 13:50

Submitted by:
Eric Cloninger