MODULE – 5: IOS AND WINDOWS 7

Syllabus: iOS – Obtaining the tools and SDK – Components of XCODE – Architecture of iOS – Building Derby App in iOS – Other useful iOS things – Windows Phone: Getting the tools you need – Windows Phone 7 Project – Building Derby App in Windows Phone 7 – Distribution – Other useful Windows Phone Thing

5.1 IOS: OBTAINING THE TOOLS

To develop an app for iOS, you may need to do a bit of planning, with the first and most expensive being hardware. Depending on your development intentions you may need to pay Apple for the honor of being an iOS developer as well. Important tools to be acquired for IOS App development are listed in to following categories:

- Hardware
- xCode and IOS SDK
- The IOS Human Interface Guideline

Details of these are discussed hereunder.

5.1.1 Hardware

To develop iPhone, iPod, and iPad applications you must have a Mac. The iPhone SDK runs only on Mac OS X. The only sanctioned hardware for iPhone, iPod, and iPad development is an Intel-based Macintosh.

- **Program Levels:** If you do not have an Apple Developer account, you can create a free account at https://developer.apple.com/. Having the Apple Developer account allows you to create iOS applications and run them locally on your machine using the iOS Simulator. To deploy applications you have created to a physical device (iPhone, iPad, iPod Touch) you must belong to the iOS Developer program.
    - **IOS Developer Program:** This program level allows developers to distribute apps in the App Store as an individual, a sole proprietor, a company, an organization, a government entity, or an educational institution. The cost for this program is $99 a year, and you are allowed to name 100 devices within your iOS Developer account.
    - **IOS Developer Enterprise Program:** This program level allows developers to develop proprietary apps for internal distribution within your company, organization, government entity, or educational institution. The cost for this program is $299 a year. This level of the program will not allow you to distribute apps through the App store, but allows ad hoc distributions (distribute directly to a device without using the App Store) to devices in your organization. A valid Dun & Bradstreet (DUNS) number is required, and this program level will take a little bit longer to get enrolled in. This process takes well over a month before acceptance into the program.
    - **IOS Developer University Program:** This program level allows higher-education institutions to create teams of up to 200 developers that can develop iOS applications. This program level is free, and allows for programs to be tested on physical devices, but does not allow for ad hoc or App Store deployment.

- **The IOS Provisioning Portal:** iOS Developer Center that allows you to create the files necessary to deploy development and distribution (production) builds onto physical devices.
    - **Certificates:** During the development process of your iOS app, you will more than likely create both a development and a distribution certificate. These certificates are used to digitally sign the app, and verify you are who you say you are.

- **App IDs:** Each iOS application that you create (that you intend to deploy to a device) needs to be identified on the App IDs section of the iOS Provisioning Portal. The app ID that is created is a unique ID that contains a number from Apple and then a bundle identifier that you specify. The bundle identifier is usually in the format `com.companyname.appname`. As you start to develop more applications, they tend to become messy in this interface.
- **Devices:** The Devices section in the iOS Provisioning Portal section allows developers to maintain a list of devices in which their iOS applications will be developed. These are the devices that are either used for testing your app or for ad-hoc deployments. The number of devices that you can register on this screen relates to the type of Apple Developer account level you selected. For example, if you registered at the iOS Developer level, you will be able to add 100 devices. This number is 100 per year, meaning if you add 100 devices and then delete 10, you are still out of spaces for accounts until you re-enroll in the program the following year, which will still only have a maximum of 100 devices.
- **Provisioning Files:** After the certificate, the app ID, and devices have been created/added, you can then create a provisioning profile. The provisioning profile combines the information about which apps/certificates can be installed on which devices. As with certificates there will be a Development and Distribution version.

5.1.2 xCode and IOS SDK

To create native iOS applications, you will need to install both the xCode IDE as well as the iOS SDK. Although you can obtain xCode by using the App Store within Mac OS X. Downloading xCode and the SDK from the downloads section in the iOS Dev Center.

- **Installation:** After you follow the steps to install xCode, you should have the xCode IDE as well as a great deal of other useful development tools installed to `/Developer/Applications`. You can start xCode from this directory or by using spotlight.
- **Components of iPhone SDK:** The iPhone SDK includes a great number of tools that help create iOS for apps. These tools range from debugging and profiling to developing. Following is the list of most common tools that we use that are included in the iOS SDK:
  - **xCode:** xCode is Apple's Integrated Development Environment (IDE) for creating Objective-C applications. xCode enables you to manage, author, and debug your Objective-C projects.
  - **Dashcode:** Dashcode is an IDE that enables you to develop web-based iPhone/iPad applications and Dashboard widgets.
  - **iPhone Simulator:** This tool provides a method to simulate an iPhone or iPad device on your Mac, for use with testing your iOS applications.
  - **Interface Builder:** The Interface Builder, or IB, is a visual editor that is used for designing the user interface for your iOS application.
  - **Instruments:** Instruments is a suite of tools that helps you analyze your iOS application and monitor for performance bottlenecks as well as memory leaks in real time while attached to an iOS device or iOS Simulator.

5.1.3 The IOS Human Interface Guideline

The iOS **Human Interface Guideline** (HIG) document is one of the most valuable tools to the iOS developer. The iOS HIG describes guidelines and principles that help the iOS developer design a superlative user interface and user experience. It is very important for new iOS developers to read through this document; if you do not develop using the UI principles found in the HIG, your application could be rejected when submitted to the Apple App Store.

5.2 IOS PROJECT

There are many steps to be followed before you start creating an IOS application. Here we will discuss few of them.

5.2.1 Anatomy of an iOS App

The files that are actually deployed to the iOS device are known as .app files and these are just a set of directories. Although there is an actual binary for the iOS application, you can open the .app file and find the images, meta data, and any other resources that are included.

- **Views:** iPhone apps are made up of one or more views. Views usually have GUI elements such as text fields, labels, buttons, and so on. You can build a view built using the Interface Builder tool, which enables you to drag and drop controls on the view, or you can create a view entirely with code.
- **Code that makes the Views work:** Because iOS applications follow the MVC design pattern, there is a clean break between the UI and code that provides the application code.
- **Resources:** Every iOS application contains an icon file, an `info.plist` file that holds information about the application itself and the binary executable. Other resources such as images, sounds, and video are also classified as resources.
- **Project Structure in Depth:** When an iOS project is created within xCode, the IDE creates a set of files that are ready to run. These files provide the basics of what is needed to get going with a new project.
    - **Main.m:** As with any C program, the execution of Objective-C applications start from the main() function, which is the `main.m` file.
    - **AppDelegate.m:** The `AppDelegate` receives messages from the application object during the lifetime of your application. The `AppDelegate` is called from the operating system, and contains events such as the `didFinishLaunchingWithOptions`, which is an event that iOS would be interested in knowing about.
    - **MainStoryBoard.storyboard:** This is where the user interface is created. In past versions of xCode/iOS the user interface was stored within `.xib` (pronounced NIB) files. Although this method is still supported, Storyboards are a great improvement over `.xib` files for applications with complex navigation and many views.
    - **Supporting Files:** The supporting files directory contains files such as the `plist` setting files (which contain customizable application settings), as well as string resource files that are used within your app.

5.2.2 Getting to Know the xCode IDE

It is important to use the correct tool for the job, regardless of whether you are constructing a house or constructing an application. If you are new to xCode, there will be a bit of a learning curve to becoming proficient with the IDE, but xCode is a top-notch IDE with many features for you to discover.

- **Navigators:** The left side of the xCode window is known as the navigator area. A variety of navigators enable you to list the contents of your project, find errors, search for code, and more. The remainder of this section introduces the Project Navigator, the Search Navigator, and the Issue Navigator. Going from left to right, the project navigator is the first of the xCode navigators; the icon looks like a file folder. The Project Navigator simply shows the contents of your project or workspace, as shown in Figure 5.1. Double-clicking a file in the Project Navigator opens the file in a new window, and single-clicking opens the file within the xCode workspace.

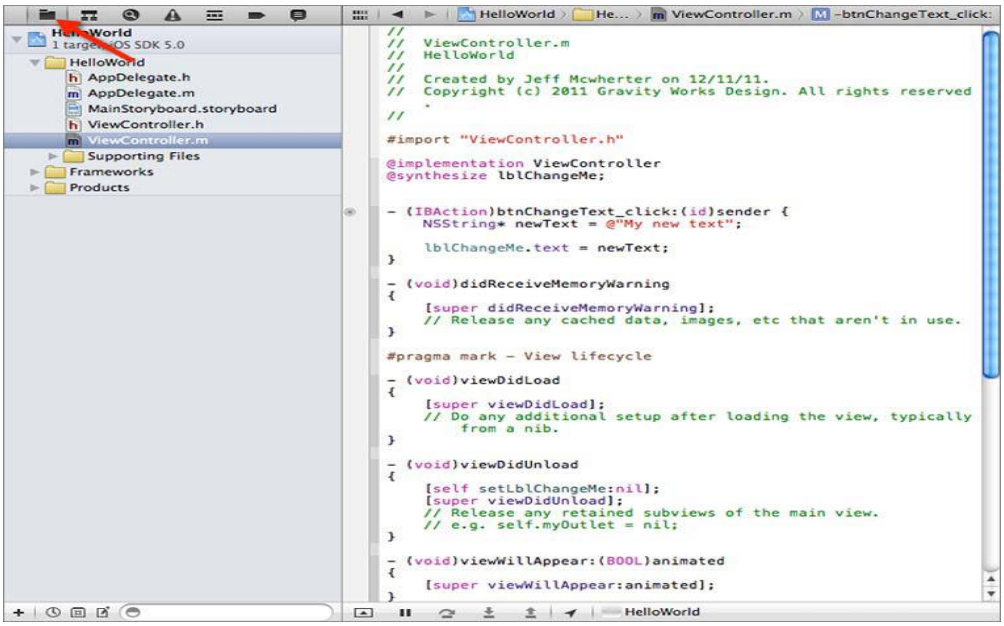

Figure 5.1 Project Navigator window

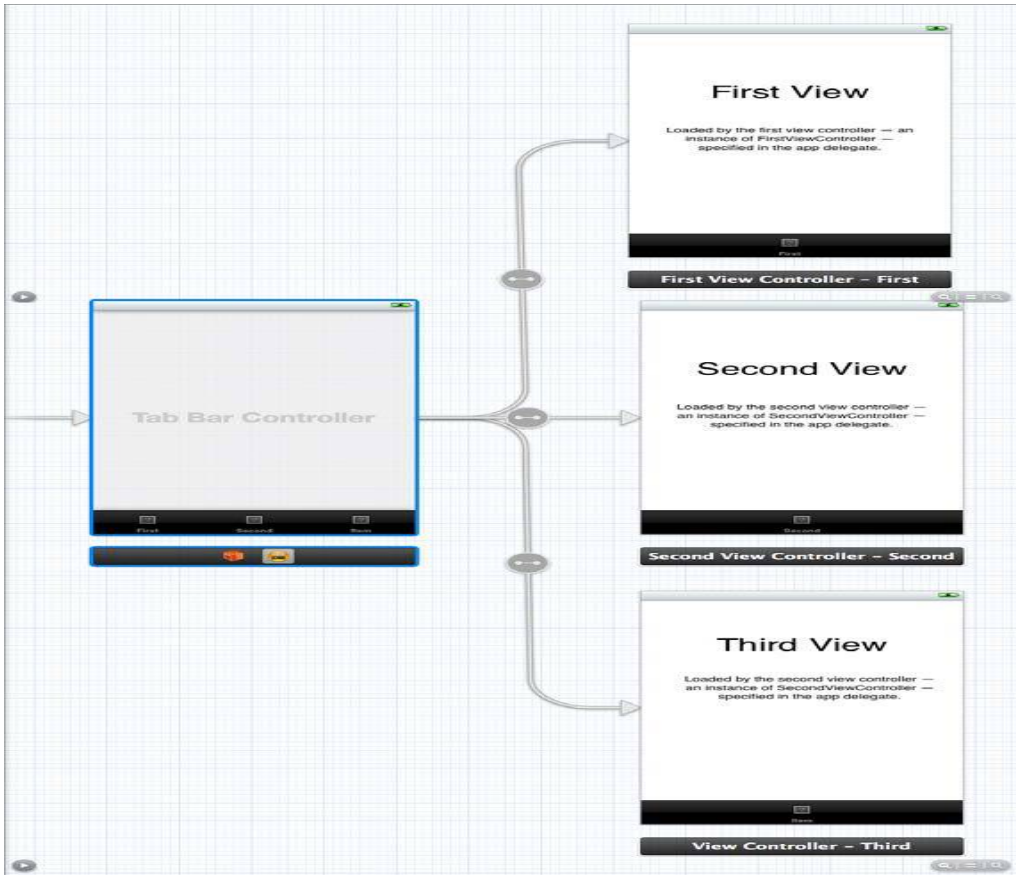

Figure 5.2 Storyboard

- **Storyboards:** In iOS versions prior to iOS 5, developers needed to create a separate XIB file for each view of their application. A XIB file is an XML representation of your controls

and instance variables that get compiled into the application. Managing an application that contains more than a few views could get cumbersome. iOS 5 contained a new feature called storyboards that enables developers to lay out their workflow using design tools built within xCode. Apps that use navigation and tab bars to transition between views are now much easier to manage, with a visual representation of how the app will flow. With Storyboards, you will have a better conceptual overview of all the views in your app and the connections between them. Figure 5.2 shows an example of a storyboard for an application containing a tab bar for navigation to three other views.

5.3 BUILDING THE DERBY APP IN IOS

The idea of the Derby App is to build the same app over all of the mobile platforms. The iOS version is very similar to the program built in Android. Following are the steps to be followed:

1. **User Interface**
2. **Team Roster**
3. **Details**
4. **Leagues and Team Names**

We will discuss each of the steps now.

User Interface: The derby app contains a list of data, so you use table views throughout the application to show the information to the user. Start by creating a new tabbed application within xCode as shown in Figure 5.3:

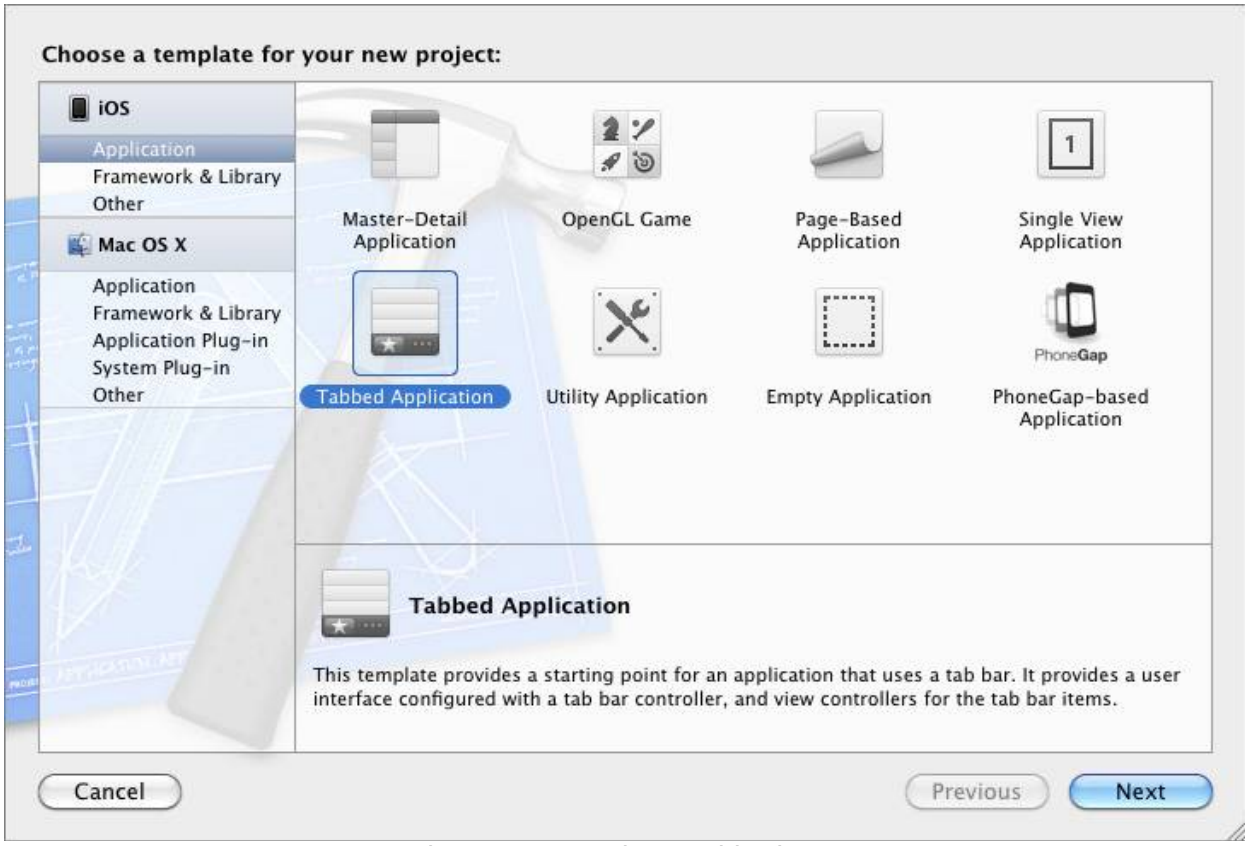

Figure 5.3 Creating a tabbed app

By default xCode creates a new iOS application that contains a Storyboard with tabbed navigation. Your project needs to contain three table views and one Navigation controller. The table views are used to the list the data about the derby teams and rosters, and the navigation controller enables you to create a "back stack" that will allow users to navigate back to the team names table view, when finished viewing the team's roster data. Start by dragging three Table View Controllers from the Object Library onto the storyboard, and then remove the views that were added by default. Once the previous views are removed and the table views are added, you need to connect them to the navigation controller. To do this, Control-click the Tab Bar Controller icon on the Tab Bar View, within the Storyboard, and drag it to the new view as shown in Figure 5.4.

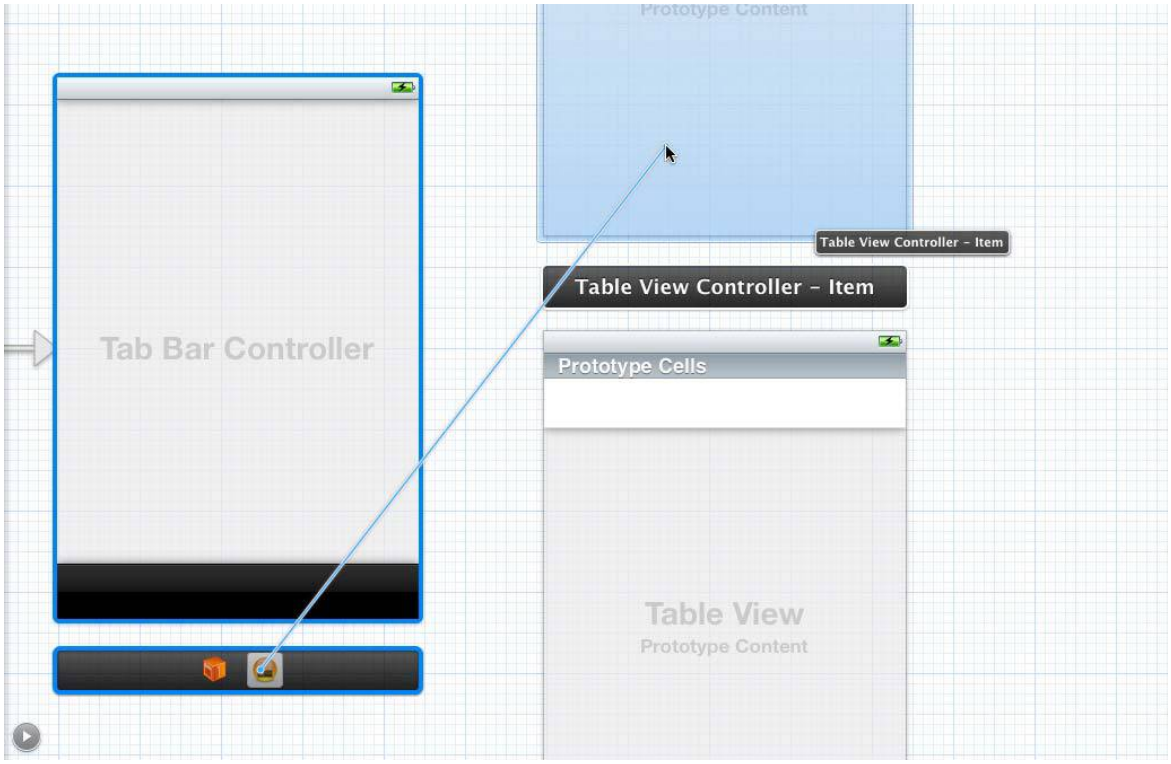

Figure 5.4 Connecting view to tab bar

Team Roaster: With the storyboard in place, you can now write the code to populate the table views. Start with the Team Roster tab, which will go out to a web service, and obtain a list of all the Lansing Derby Vixens. First create a new class named `VixensViewController`:

```
@interface VixensViewController : UITableViewController
@property(nonatomic, retain) NSArray *listData;
@end
```

In the viewDidLoad method within the implementation file, add your logic to go out the web service and get the roster for the Lansing Derby Vixens. Notice the filter criteria contained within the URL. iOS 5 is the first version of the iOS SDK that contains built-in support for parsing JSON strings. This is great because you do not have to depend on a third-party tool.

Details: The next class you need to create is DetailViewController. This is the code that drives the details that are displayed when a team name is selected.

```
#import <UIKit/UIKit.h>
```

```
@interface DetailViewController : UITableViewController
@property (nonatomic, retain) NSString *data;
@property(nonatomic, retain) NSArray *listData;
@end
```

Leagues and Team Names: Listing all of the team names is very similar to the code you just created for displaying the roster and the details. Start by creating a new class named LeagueTableViewController:

```
#import <UIKit/UIKit.h>
@interface LeagueTableViewController : UITableViewController
@property(nonatomic, retain) NSArray *listData;
@end
```

After all of the code has been added to the new View Controllers you created, you need to go back to the Storyboard and attach them as shown in Figure 5.5. You do this using the Identity Inspector found near the upper right of the screen when you are viewing the storyboard. Map each view to the correct class that was created. Final application looks as given in Figure 5.6.

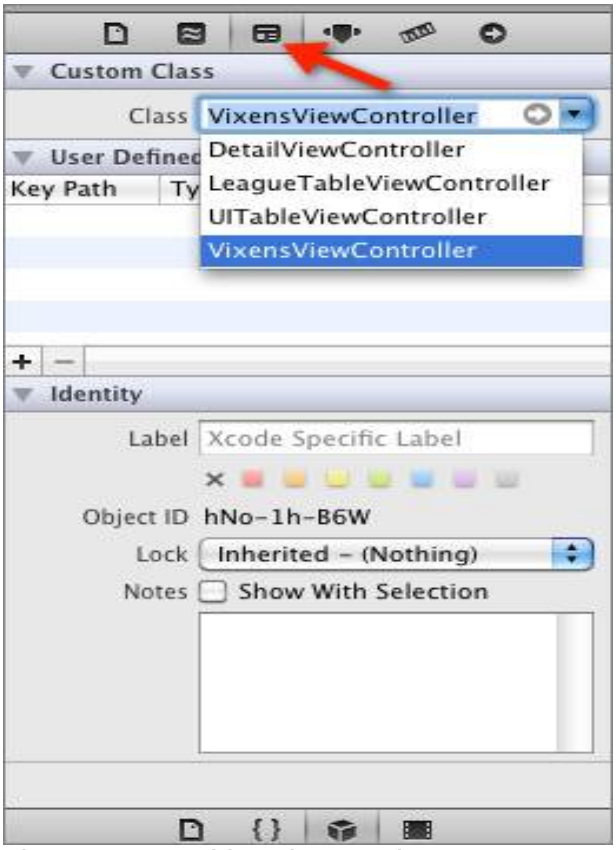

Figure 5.5 Attaching class to view

Figure 5.6 Completed Derby App

5.4 Other Useful IOS Things

IOS applications can be used do many more tasks as explained in following sections.

5.4.1 Offline Storage

Even if your application is using a web service for retrieving information, at some point you may need to save information on the device. Depending on the size and type of data, you have a few different options.

Plist : Property lists are the simplest way to store information on the device. In the Mac world, many applications use the plist format to store application settings, information about the application, and even serialized objects. It's best to keep the data contained in these files simple and small, though. The following example finds the path to a plist stored in the supporting files directory, with a name of example. It then loads the plist into a dictionary object, and loops through each time writing the contents of each item in the plist to the debug console.

```
- (void)getValuesFromPlist
{
// build the path to your plist
NSString *path = [[NSBundle mainBundle] pathForResource:
@"example" ofType:@"plist"];
// load the plist into a dictionary
NSDictionary       *pListData       =       [[NSDictionary       alloc]
initWithContentsOfFile:path];
// loop through each of the Items in the property list and log
for (NSString *item in pListData)
NSLog(@"Value=%@", item);
}
```

Core Data: If the data that you need to persist on the device is nontrivial, meaning there is a great deal of it or its complex, Core Data is the way to go. Core Data is described by Apple as a "schema-driven object graph management and persistence framework." Core Data is not an ORM (Object Relational Mapper). Core Data is an API that abstracts the actual data store of the objects. Core Data can be configured to store these objects as a SQLite database, a plist, custom data, or a binary file. Core Data has a steep learning curve, but is well worth learning more about if your app will have a great deal of data held within.

5.4.2 GPS

One of the great benefits to mobile devices is the GPS functionality. Once you are able to get over the hurdles of learning the basic functions within the iOS platform, starting to work with the GPS functions can be a great deal of fun. The GPS functions are located in the CoreLocation framework, which is not added to a new project by default. To do this, you will need to click the Build Phases tab on the project settings page.

Once on the Build Phases tab, expand the Link Binary With Libraries section, and click the + button. You are then prompted with a list of frameworks to add. Select the CoreLocation.framework. Use the code given below –

```
// GPS Example
locationManager = [[CLLocationManager alloc] init];
locationManager.delegate = self;
locationManager.distanceFilter = kCLDistanceFilterNone;
// get GPS DatalocationManager.desiredAccuracy =
    kCLLocationAccuracyHundredMeters;
[locationManager startUpdatingLocation];
```

5.5 Windows Phone

Here, we will discuss the basics of developing for Windows Phone 7 — how to acquire the tooling and basic design patterns, and preparing to distribute your application to the marketplace.

5.5.1 Getting the Tools you need

To develop software for Windows Phone 7 you need a machine running Windows (Vista or Windows 7), Visual Studio 2010, and the Phone SDK, as well as a Windows Phone 7 device to test with.

Hardware: HTC, Nokia, and Samsung are currently manufacturing Windows Phone 7 devices. Device resolutions are 800X480 pixels, and most are outfitted with both front- and rear-facing cameras, and screens from 4.3 to 4.7 inches. They are primarily found on GSM carriers.

Visual Studio and Windows Phone SDK:

- Code for the Windows Phone is written in the .NET Framework, either with XNA (Microsoft's run time for game development), or a custom version of Silverlight (Microsoft's Rich Internet Application framework).
- User interfaces are created with XAML (eXtensible Application Markup Language).
- The Windows Phone SDK works with Visual Studio Express, and will install it if you don't already have it installed.
  - If you have another version of Visual Studio 2010 on your machine it will add the functionality to that install.
  - The SDK also installs a specialized version of Expression Blend set up to work specifically for Windows Phone 7 development.
- Expression Blend is a tool developed by Microsoft for working with XAML.
  - It provides similar functionality to Visual Studio, though its layout is designed to be more user friendly to designers.

Installation:

- Microsoft's App Hub (http://create.msdn.com/) is the download site for the Windows Phone SDK.
- The Windows Phone SDK installer includes:
  - Visual Studio 2010 Express for Windows Phone (if you do not have another version of Visual Studio 2010 installed)
  - Windows Phone Emulator
  - Windows Phone SDK Assemblies
  - Silverlight 4 SDK
  - Phone SDK 7.1 Extensions for XNA Game Studio
  - Expression Blend for Windows Phone 7
  - WCF Data Services Client for Windows Phone
  - Microsoft Advertising SDK

To install the Windows Phone SDK you need:

- Vista (x86 or x64) or Windows 7 (x86 or x64)
- 4 GB of free disk space
- 3 GB of RAM
- DirectX 10 or above capable graphics card with a WDDM 1.1 driver

Getting to Know Visual Studio: Visual Studio is the integrated development environment from Microsoft for the .NET Framework. Visual Studio Express for Windows phones is a trimmed-down

version of Microsoft's full retail products. It provides developers with everything they need to develop Windows Phone 7 apps. Though the interface may seem daunting to the uninitiated, it has a relatively simple learning curve. You are afforded both a WYSIWYG (What You See Is What You Get) and text-based editor.

Getting to Know Expression Blend: Expression Blend is a user interface design tool developed by Microsoft, with emphasis on a WYSIWYG design for XAML-based projects (Silverlight and WPF). Figure 5.7 shows the Blend UI with the standard tools displayed in a Windows Phone 7 Pivot application.

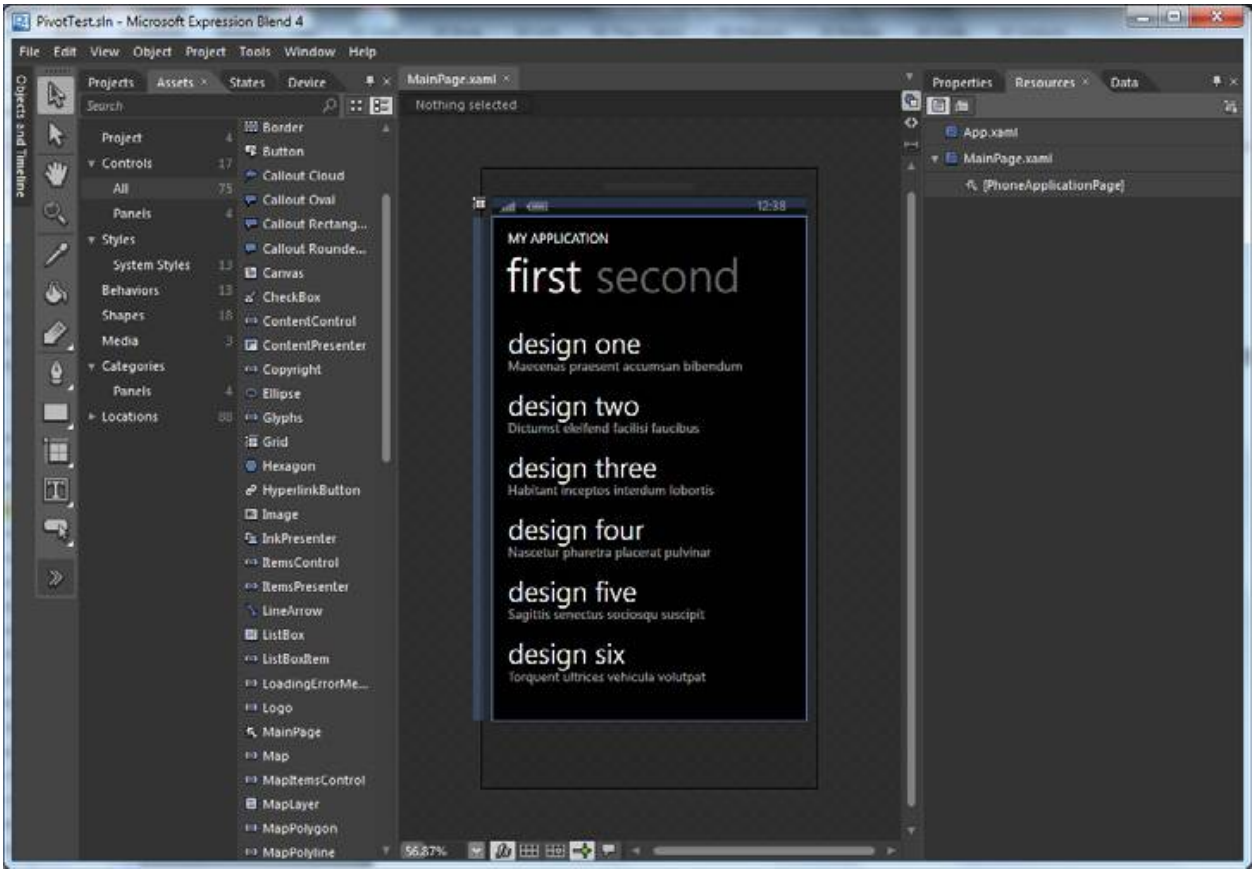

Figure 5.7 Expression Blend for visual studio

5.6 Windows Phone 7 Project

A Windows Phone 7 project is similar to a Silverlight project, which is technically a subset of WPF.

5.6.1 Silverlight vs. Windows Phone 7

Silverlight contains a subset of APIs from the .NET Framework, all optimized to run in a browser host so that it can be run cross-platform. The Windows Phone 7 SDK is a subset of that. When porting third-party Silverlight libraries you have to make sure they build against the WP7 version of Silverlight because some of the APIs don't transfer. Windows Phone 7 contains a fair amount of controls for your application, but they are not all inclusive. Multiple control toolkits have been released on CodePlex. Most of the companies that produce control packages for WPF and Silverlight have created Windows Phone 7 control packages as well. Telerik, ComponentOne, and

Infragistics all have prebuilt and skinnable control packs for Windows Phone 7 ranging in price from roughly $100 to $1,500.

5.6.2 Anatomy of a Windows Phone App

Here we discuss the basic design elements used in Windows Phone 7 application development, and how you can leverage the tools you have at hand to implement them.

1. **Storyboards:** Storyboards are Silverlight's control type for managing animations in code. They are defined in a given page's XAML and leveraged using code behind. Uses for these animations are limited only by the transform operations you are allowed to perform on objects. Anytime you want to provide the user with a custom transition between your pages or element updates, you should consider creating an animation to smooth the user experience.

   Because storyboards are held in XAML you can either edit them manually or use Expression Blend's WYSIWYG editor. In Blend, in the Objects and Timelines Pane at the left, click the (+) icon to create a storyboard. Once you have a storyboard, you can add key frames on your time line for each individual element you would like to perform a transformation on. This can include moving objects and changing properties (like color or opacity). After setting up your time line, you can start the storyboard in code. The name you created for your storyboard will be accessible in code behind.

2. **Pivot vs Panorama:** Both the Pivot and Panorama controls are used to delineate categories and subsets of data. With the Pivot control you get strict isolation of these groupings, with the menu providing discoverable UI to show the other categories. 2. With the Panorama control you get transitions between the groupings with discoverable content on the window boundaries.

5.6.3 The Windows Phone Emulator

The Windows Phone 7 emulator is a very powerful tool. Not just a simulator, the emulator runs a completely sandboxed virtual machine in order to better mimic the actual device. It also comes with some customization and runtime tools to manipulate sensors that are being emulated on the device, including GPS and accelerometer, as well as provide a way to capture screenshots while testing and developing applications. The debugging experience inside of Visual Studio is superior to the ones in Eclipse and the third-party frameworks. The load time of the Emulator is quite fast. It acts responsively, and the step-through just works.

5.7 Building Derby App in Windows Phone 7

Here, you implement the features of the Derby Names project using Microsoft Visual Studio, while also taking time to learn Windows Phone 7–specific technologies.

- **Creating the Project:** Open Visual Studio and create a new Windows Phone project. For this application, choose Panorama because it offers a UI in which you can share your data. In a Panorama application the application is created with the default Panorama background. Visual Studio will create `SampleData` and `ViewModels` for your application. Ultimately, you will be able to remove these from your application when you implement your service communications. `App.xaml` is the entry point for your application and `MainPage.xaml` is the page that loads by default.

- **User Interface:** The default Panorama application defines its `DataContext` in XAML. The `DataContext` has first item's binding associated by default. The Panorama control can be likened to any collection-based UI element (`UITableView` in iOS or the `ListView` in Android), and the `PanoramaItems` are the respective rows in that collection element. When you feel familiar enough to start working with the data you will need to create a service reference to the Odata feed.

  To reference an OData feed you need only to right-click your project, choose Add Service Reference, enter in the URL of your service, and click Go. After it has found the service it should enumerate the models. You are then allowed to update the namespace and create this reference. Once you create the service you can start working with the Panorama control to bind the data available from these entities. After you have made this service, be sure to reference this entity context when your page needs to make calls to the service:

```
readonly DerbyNamesEntities context = new DerbyNamesEntities(new
                            Uri("http://localhost:1132/DerbyNames.svc/"));
```

- **Derby Names:** To bind data to your Panorama item you need to set the `ItemsSource` and `TextBlock` bindings. Each individual entry in the DerbyNames entity in OData contains properties for Name and League, which you will bind to the `TextBlocks` in your Panorama item.

- **Leagues:** Each derby team belongs to a league. The entity for League is similar to the DerbyNames entity, and will make it easy to bind from.

5.8 Distribution

To distribute applications in the App Hub you must create a developer account at `https://users.create.msdn.com/Register`. Registration costs $99 per year and allows you to:

- Make free, paid, or ad-funded apps and games.
- Submit unlimited paid apps to Windows Phone Marketplace.
- Submit up to 100 free apps to Windows Phone Marketplace; additional submissions are $19.99 USD per submission.
- Expand your reach with worldwide distribution and trial options.

Additionally, all apps are content and code-certified.

Submitting your application to the App Hub is a five-step process:

1. You upload your compiled application XAP file. The XAP file is the binary for your application that will be pushed to the phone. To do this you must have a unique name for your application, you must select whether this application is being released to the public or simply being distributed to the App Hub for a private beta test, and you need to specify a version for your app.
2. Next, you must provide an application description (this includes category of app, keywords, detailed description, languages supported, and art assets)
3. Then, you set up your price and select what markets you want to distribute your application in.
4. Next, you provide test information so that the developer in charge of approving your app understands the use cases.

5. Finally, you choose your publishing options (as soon as approved, as soon as approved but hidden, manual publish). You then submit your application for certification.

5.9 Other Useful Windows Phone Thing

5.9.1 Offline Storage

Windows Phone 7 has the `System.IO.IsolatedStorage` namespace to handle persisting data between application runs. Isolated storage is application-specific storage on the device filesystem. The simplest means of implementing an isolated storage solution in Windows Phone is to leverage your `PhoneApplicationService`'s state-based events. `Launching` and `Activated` handle application load and resume from tombstone, respectively; `Closing` and `Deactivated` handle application exit and tombstoning, respectively. Making sure that your application loads your isolated storage instance on `Launch` and `Activate`, and saves on `Close` and `Deactivate`, gives you tremendous capability with little effort.

The following code persists a unique identifier to be passed to a web service as part of an authentication token. You first need to declare the property for the unique identifier in your `App.xaml.cs` (your application's code behind file):

```
public partial class App : Application
{
    public Guid UserAuthToken { set; get; }
}
```

5.9.2 Notifications

Setting up notifications for Windows Phone 7 is a multistage process. First you must build up a push channel to receive communications within your app. Creating that push channel provides you with a Service URI to post data to. Posting data in specific formats determines what type of message will be displayed to the client app. There are **three types** of notifications:

1. **Toast notification:** The first and simplest is the *toast notification.* With a toast notification you can pass a title, a string of content, and a parameter.
    - The title will be boldfaced then displayed
    - the content will follow non-boldfaced, and
    - the parameter will not be shown, but it is what is sent to your application when the user taps on the toast message. This can contain parameters to load on the default page, or a relative link to the page you want loaded when the app loads as a result of the tap. Then the user taps on the *toast message.*

2. **Tile notification:** With the tile notification you can update the application tile content. The XML data that you post contains fields for the title on
    - the front of the tile,
    - front of the tile background image,
    - the count for the badge,
    - the title for the back of the tile,
    - the back of the tile background image, and
    - string of content for the back of the tile.

3. **Raw Notifications:** The third and most developer-centric notification type is raw. With the raw notification type you can pass data directly to the app. It will not be delivered if the application is not running.

5.9.3 GPS

Windows Phone 7 has built-in functionality for leveraging the geolocation sensors in your device. Using the `System.Device.Location` namespace and tracking the `PositionChanged` event of a `GeocoordinateWatcher` adds a button to your application bar that will tell you the device's distance from our local derby team, the Lansing Derby Vixens. The Windows Phone emulator has a great interface for mocking GPS location changes while developing and debugging your app.

5.9.4 Accelerometer

In addition to GPS, Windows Phone 7 devices are outfitted with an accelerometer. The emulator provides a 3-D interface for simulating accelerometer change events. You can track the movement of the device by capturing the `ReadingChanged` event on the accelerometer. However, you need to have a delegate to call back to the UI thread if you want to display anything special based on the event. If the application can access the UI thread, the `ReadingChanged` event handler will call the delegate function; otherwise, it will dispatch the event on the UI thread. You must also make sure that when you are done capturing this data, you stop the accelerometer to preserve battery life.

5.9.5 Web Services

The Derby application is an example of leveraging data over the web to add value to your application. If you don't want to be the central repository for all data exposed to your users, you can leverage web services that exist from other vendors.

Questions:

1. What are the tools required for developing app for iOS? Explain.
2. What is program level? Discuss the various programs levels applicable for an iOS developer.
3. Explain the components of iPhone SDK.
4. Discuss anatomy of iOS app.
5. Explain various steps in developing derby app in ios.
6. Explain silver-light and windows phone 7.
7. Discuss the tools required to develop Microsoft Windows 7 applications.
8. Explain anatomy of Windows Phone app.
9. Explain various steps in developing derby app in Windows Phone 7.
10. Write a short note on
    a. ios story board.
    b. accelerometer in windows phone 7
    c. Distribution of Windows Phone 7 App
    d. Notifications in Windows Phone 7 App.