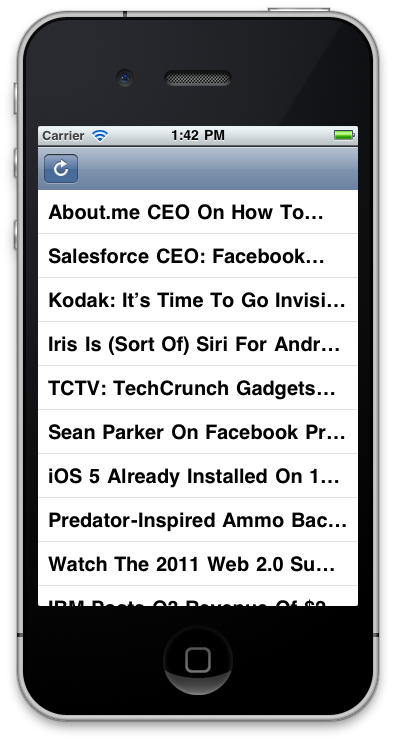
**ASSIGNMENT 6 - Model-View-Controller Pattern, App Lifecycle**

Due Week 10

**Task 1 - Separate Your Concerns, Part II**

* 1. **Application Demo**

****

**Figure 1: RSS from TechCrunch**

**1.2 Model-View-Controller design pattern**

The Model-View-Controller (MVC) pattern separates the modeling of the domain, the presentation, and the actions based on user input into three separate classes. In my application, I used MVC (Model-View-Controller). It is very good design patterns to develop the data source to users. Recently, user interface always changed to improve user experience. With MVC, each component is separated. Each component is easy to test and modified. For example, if user interface engineer design a completed new interface. Developer only needs to developed new view classes without effective the other classes. The View is just user interface. It shows the data source to users. However, there are many data source format. View does not know what data source is, like html, xml, JSON, CSV. View need controller to understand the contents of external or internal data source. Controller should convert this data source that view can understand. Also the all user inputs are handle by controller not view. For example, if user clicks on the screen, controller should respond users action. If in the game design, controller should monitor view’s location and user click location. If they are matched, click happened. The model is a data source from outside. In sometimes, Model may change. For example, a database has a new column in entity. Controller should change API to retrieve data.

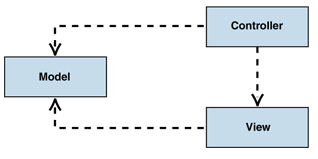


Figure 2: depicts the structural relationship between the three objects.

(Source: <http://msdn.microsoft.com/en-us/library/ff649643.aspx>)

MVC Structure in my app:

* view: RSSViewController.xib file.
  + UITableVIew
  + ToolBar
* Controller:
  + RSSViewController.h
  + RSSViewController.m
* Model:
* Rss.xlm in bundle
* Rss from internet

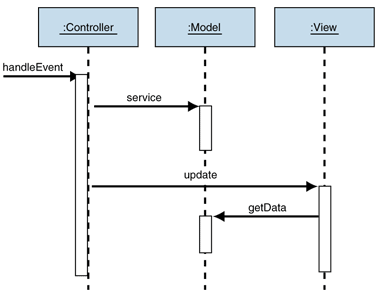


Figure 3: Behavior of the passive model

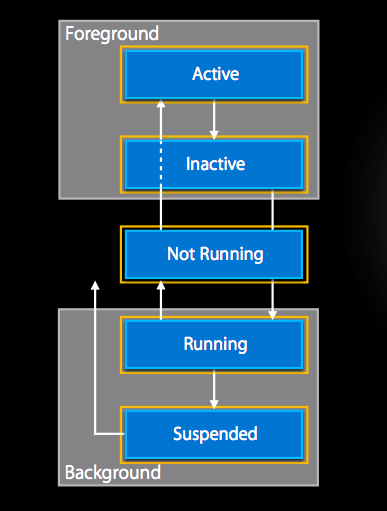
Source: <http://msdn.microsoft.com/en-us/library/ff649643.aspx>

Figure 2 show how does the message pass in the Model View Controller design patterns. Controller handles user’s action. User’s action is one of Create/Read/Update/Delete. Controller will change the change the data in the model. After update model, controller should also update the view display the updated model.

In my application, user can only create and update data from model. When the view is load, each cell will load data from xml to memory. This data is converted from XML. When each time the xml load to memory, the reloadData will be called to update UITableView. We can use interface builder to connect button in the view with controller.

**Task 2 – The Cycle of Life**

The application in mobile device has some special event can reduce user experience, like phone call, SMS message, slowly 3G networking, limited battery and limited memory. Application needs delegations to handle events in iOS. When a project is created in Xcode, an application delegation file will generate automatically.



When the application running first time, the following method will called

* application:didFinishLaunchingWithOptions: is used to Load all resource to memory and initialize the game environment.
* applicationDidBecomeActive:

When the application is interrupted by a phone call or SMS message, only one method will called.

* applicationDidEnterBackground:

When user press home button, OpenGL ES will be terminal by OS, following method will be called. Or user decides to answer the phone call from last session.

* applicationDidEnterBackground:
* beginBackgroundTaskWithExpirationHandler

beginBackgroundTaskWithExpirationHandler is used to run for a period of time after it transitions to the background, if the task need more than 5 seconds to process the data, like networking or permit storage.

When user presses twice home button to launch application again or click an application’s icon to bring a suspend application to foreground, follow methods will be called.

* applicationWillEnterForeground
* applicationWillResignActive:

When user presses the lock button while on the Main Menu of the application, outside of a game. The application did not transition to background. It is still running in foreground. The follow method will be called.

* applicationWillResignActive: