

ECE 480 Group 1 Application Notes

Ziye Xing

11/12/2015

Abstract

We will introduce a novel method taking the advantage of RFID technology to manage the baggage tracking system implemented in airport. This system will assign each baggage with a RFID tag which will store the unique ID and information it associated with. Both a fixed and portable reader could scan the tag and read the stored information. The users of this system will be able to use an iPhone APP to check the current location in real time and use the portable reader to check the authentic one on the conveyor. Our purpose is to provide a more efficient and more reliable baggage tracking system for all users, and ensure their baggage arriving destination as planed.

Keywords: RFID, baggage, iPhone application, Swift, object-oriented programming, Model-View-Controller

Introduction

This note will cover the software design along with database server and their implementation method and concept involved in this project. This note will also introduce all the tools and methods used during the developing procedure. One who wish to understand how the software was developed or how the database server was implemented should read this note. After reading this note one should be able to understand the code structure of the application as well as the

implementation details or even reproduce a similar project. Please notice that this note will not introduce any hardware reference or parameter used, if you are interested in related material, please read other notes for more information.

Application Front Development

The application was developed on the iOS platform by Swift language. The development environment is the latest Xcode 7.0 platform. The main purpose of this application is to allow the user to check the current location of their baggage and keep tracking of them by using their mobile device. The rationale behind this is that user will search their baggage from the database and add this item to a list to track it. In Figure 1, it displays the main storyboard of this project.

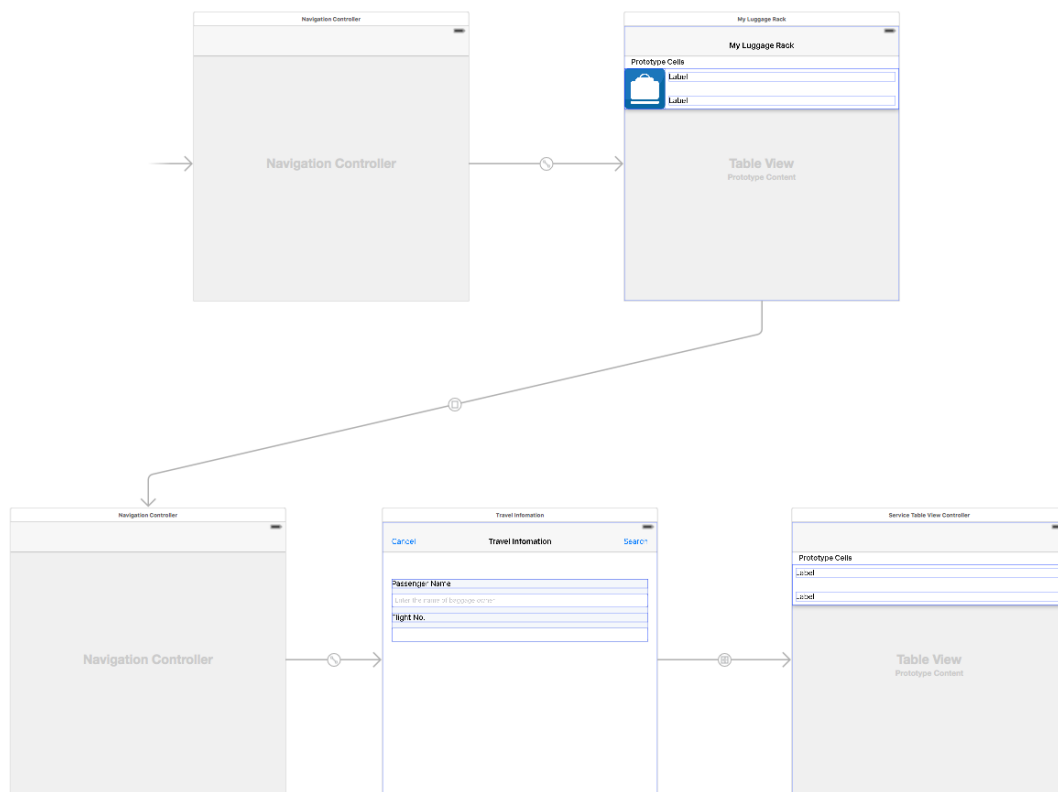


Figure 1: tentative main storyboard

The storyboard shows the flow path of the view in this application. Here the screen displayed by phone is consist of these views. Figure 1 is only a tentative storyboard because the application is still under developing. There are more views to be added to completed the final project.

This is the View part in the Model-Controller-View(MCV) pattern. Each view is controlled by a controller. The controller will process the logic behind the scene and decide what to display on the screen. You can see 5 views from Figure 1 for now. The initial view is indicated by the small arrow on the left of the first row. It will be the view when user first start the application. For 2

navigation controller view, they will provide a navigation bar on the top of the application to allow user to control the flow of the process. For example, as shown in Figure 2, it has two buttons in the navigator bar, one is “Edit” and the other one is “Add”. The reason there are 2 of the navigation controller is because the second one will produce a visually consistent view between the transition of views. The rest are

BaggageTableViewController,
SearchViewController, and
ServiceTableViewController respectively in the order as show in Figure 1 as above. The baggage table as also shown in Figure 2 provide a place to allow user to save their searched result and repeat to

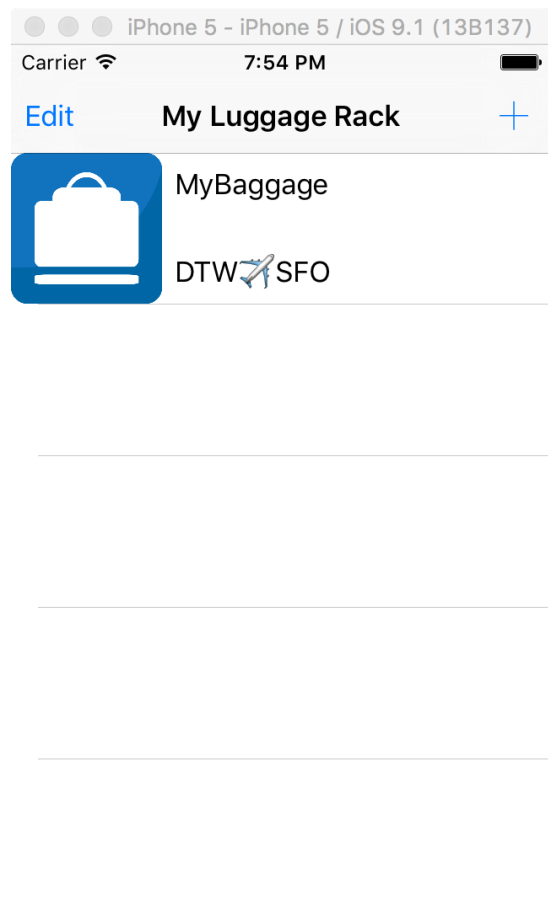


Figure 2: Navigator Bar and Table View

check status after once. This is also a good perform for saving history information. The search view provides the interface for user to enter the query information to be used in searching process. The service table will display the searching result for a sent query and list all the possible results.

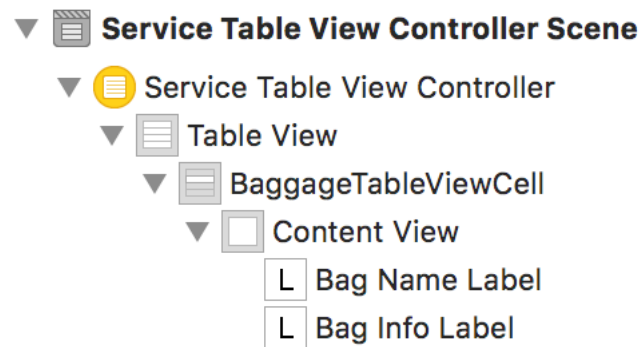


Figure 3: Table View Controller Structure

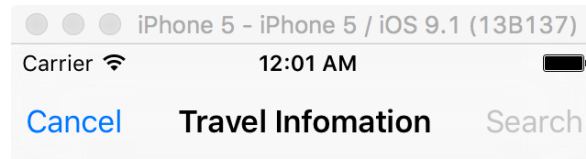
As shown in Figure 3, a table view controller has a hierarchy structure which will organize the contents in this format. The table view will have a table list structure to perform the behavior for all cells listed in this table, while the view cell will control the behavior of individual content view in this cell. The content view is the one structure defined the actual information displayed in each cell. For this particular example, it will have two attributes show name and info label in that cell.

In order to make the table view work properly, we need to write override functions to manipulate the behavior of the table view. First we need to indicate the number of rows in a section of table view. As seen in Figure 4, the first function with return value 1 indicate this table view contains one section. The next function with signature `numberOfRowsInSection` indicates the rows will have the same number as the size of baggage list. The next override function will mainly control the behavior of each cell. To be specific, the label attributes of the cell will be filled with

In the search view controller, users are supposed to insert some information to conduct the searching action. This will involve the concept of delegate. As stated above, the MVC pattern will separate the display with the controller behind it. The controller will decide what is shown on the screen. However, if we want the view send some information back to controller, we have to use the delegate to implement since the view should not directly talk to controller. For this particular example, we need the user input the data for searching, the user need use keyboard to enter string to the text field and the text field need to send the string to search view controller. By doing this we need to inherit the `UITextFieldDelegate` protocol, and then use this

`lastName.delegate = self` statement to declare itself as the delegate of the text field.

When the user tap the text field, the keyboard will be called and allow user to type words in the text field. One thing to should be noticed is that when the text field is empty or when the user is typing the search button should be disabled since the input is invalid for sure. In order to do this, we need be familiar with the concept of First Responder. When the keyboard is called, it will become the First Responder, meaning it is on the top of the view and the action interacted with view should be disabled. When user finishes typing, the keyboard will resign the First Responder, and that is when we should perform the input checking. If the input is invalid or empty, the search button will be disabled as shown in Figure 8.



Passenger Name

Ziye Xing

Flight No.

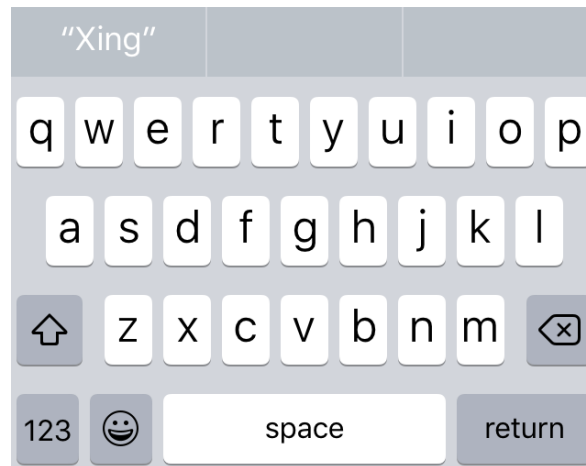


Figure 8: Keyboard as First Responder

Until now, there is one more thing we need to do before we conduct the search request to the database server. Xcode disabled the transport connection setting by default. In order to sent the http request to the server, we need to add a transport security setting entry into the Info.plist file under the project directory, as shown in Figure 9. And now we are ready to

Key	Type	Value
Go Back		
▼ Info...	Dictionary	(15 items)
Localization native development r...	String	en
Bundle display name	String	
Executable file	String	\$(EXECUTABLE_NAME)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	XPC!
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1
► NSExtension	Dictionary	(2 items)
RemoteInterfacePrincipalClass	String	InterfaceController
WKExtensionDelegateClassName	String	ExtensionDelegate
Application Category	String	
▼ NSAppTransportSecurity	Dictionary	(1 item)
NSAllowsArbitraryLoads	Boolean	YES

Figure 9: Info.plist file

Database Server Deployment & Connection

Our project used Amazon Web Services to support the remote database deployment. AWS has the virtual servers in the cloud and is free to use for basic recourse, which is suitable for the demonstration purpose at the preliminary stage of this project. For more information about the amazon Elastic Compute Cloud (EC2) service, please visit the introduction page:

<https://aws.amazon.com/ec2/>.

We deployed the Apache web server with PHP and MySQL on the Linux instance of Amazon EC2. This LAMP stack allows our mobile phone application to actively search the information in the database and to get the real time update. We have configured the security group to allow MySQL (port 3306), SSH (port 22), HTTP (port 80), and HTTPS (port 443) connection. We also have installed phpMyAdmin for user interface of database management. For more information about LAMP server configuration, please follow the tutorial at:

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/install-LAMP.html>.

The web server is hosted at: <http://ec2-54-148-117-15.us-west-2.compute.amazonaws.com/>.