

NYC Apps Proof of Concept Mobile App Platform Setup and Application Guide

Table of Contents

Chapter 1 Getting Started	3
1 Overview	3
1.1 Cross Platform Mobile Framework	3
1.1.1 Environment Setup - Rhodes	3
1.1.2 Environment Setup – Mobile SDKs	3
1.1.3 Sample Application Source Code	4
1.1.4 Application Configuration Files.....	4
1.1.4.1 “build.yml”	4
1.1.4.2 “rhoconfig.txt”	4
1.1.5 Running Sample application	5
2 Application Overview.....	5
2.1 Screen Flows	5
2.2 Views	6
2.2.1 Recent	6
2.3 Other Common Functionality:.....	11
2.3.1 Tabs:.....	11
2.3.2 CSS Stylesheets:.....	11
2.3.3 Navbar	12
2.3.4 Menu items.....	12
2.4 Data Persistence.....	12
2.5 Calling the Web Service.....	13
2.6 Data from Web Service	13
2.7 Geolocation and Mapview	13
Revision History.....	Error! Bookmark not defined.

Chapter 1 Getting Started

1 Overview

1.1 Cross Platform Mobile Framework

The solution uses Rhomobile, which is an “open source Ruby-based framework to rapidly build native apps for all major smartphone operating systems (iPhone, Windows Mobile, RIM, Symbian and Android).”.

From <http://rhomobile.com/products/rhodes/>

Rhomobile supports creating mobile apps using support for Model View Controller framework with:

- Model – data persisted with SQLite on device
- View – display views based on HTML / CSS / Javascript
- Controller – business logic developed in Ruby

This document outlines the steps required to set up the environment to work with the sample code submitted. This application integrates with services hosted on Google App Engine (GAE) and is subject to Google App Engine’s “Free Default Quota” limits outlined for application resources. For more information on free and billable quota limits, refer to <http://code.google.com/appengine/docs/quotas.html>

1.1.1 Environment Setup - Rhodes

The following section details the setup required to for the sample Proof of Concept application to work with Rhomobile.

Follow the steps in http://wiki.rhomobile.com/index.php/Tutorial#Install_Rhodes for your operating system.

1.1.2 Environment Setup – Mobile SDKs

Follow the steps here for the mobile platform you wish to develop for:

http://wiki.rhomobile.com/index.php/Building_Rhodes_Apps_on_Supported_Platforms

iPhone development can only be done on OSX operating system. Android development can be done on OSX or Windows.

After setting up the mobile SDK and downloading Rhodes, run “rhodes-setup” and enter the location of the mobile platform SDKs.

1.1.3 Sample Application Source Code

The source code for the sample application can be downloaded from:

`git@git.github.com:nycapps-poc/myneighborhood.git`

The command to clone the code is

`“git clone git://github.com/nycapps-poc/myneighborhood.git”`

The POC application is called “NYCAppsPOC”. The original project was called “myneighborhood”. If you are not already using Git, see <http://help.github.com/> for more information.

1.1.4 Application Configuration Files

1.1.4.1 “build.yml”

After downloading the code, you need to set the build.yml file (which outlines the build parameters) to your environment by running “set-rhodes-sdk” from the base folder of the application.

“Name: <replace with your own application name>”

The SDKs that your application will be packaged to run for are also specified here.

Apikey:

If you are developing for Android, you will need to replace the “API key” value with your own key. See <http://code.google.com/apis/maps/signup.html> for details of obtaining your own key. This is only required for the Mapview to load on Android.

From the “build.yml” file:

```
android:
  version: 2.2
  apikey: <replace with your own key value>
  mapping: true
```

1.1.4.2 “rhoconfig.txt”

These are the following custom parameters that the application uses:

App Parameters

Modified: 1/26/2011 12:34 PM
© 2011 Accenture LLP. All rights reserved. Any third-party names or trademarks contained in this document are the property of their respective owners.

appname = 'NYCAppsPOC' (name of the application that is displayed on Titlebar)

version = '1.0' (version number of the application –displayed in “Settings, About” view in the application)

The “data_url_base” key sets the URL of the web service the application will call to retrieve the data.

1.1.5 Running Sample application

To list the commands to build and deploy to emulators or devices, type “rake -T”.

To compile and deploy to the iPhone emulator, type “rake run:iphone”

To compile and deploy to the Android emulator, type “rake run:android”

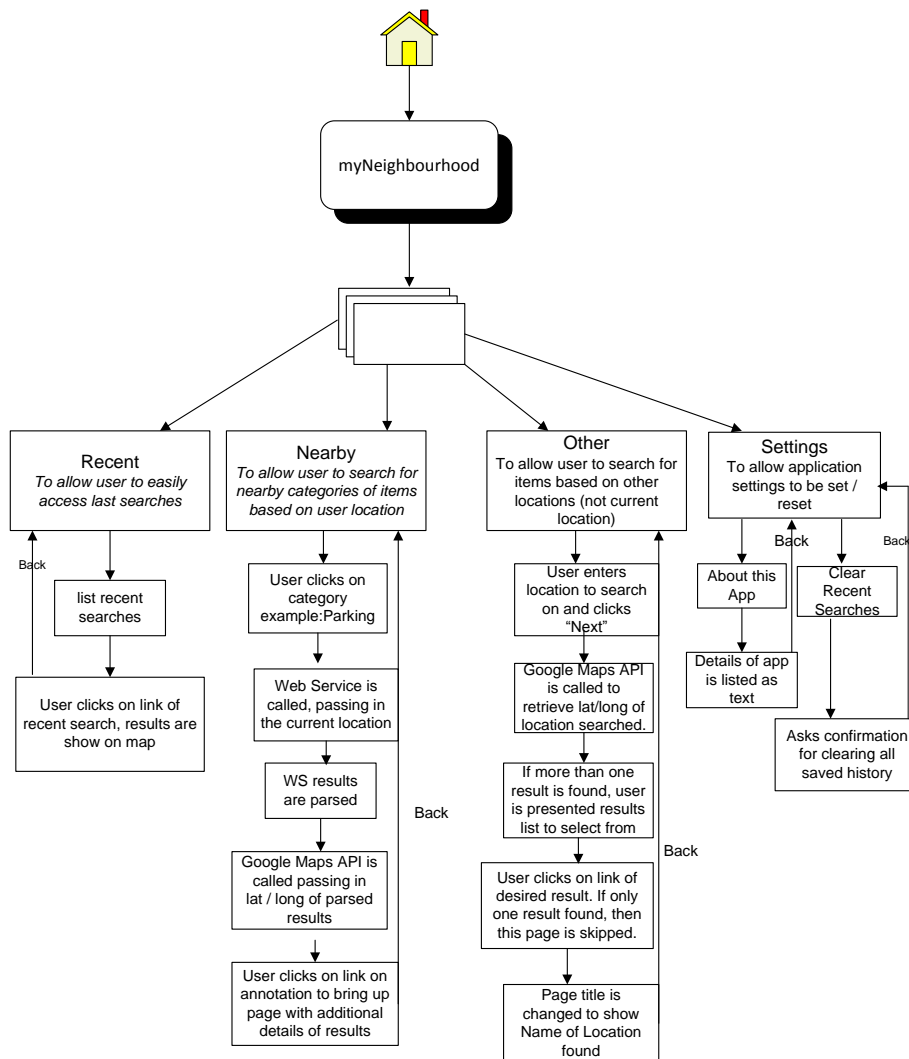
2 Application Overview

The following are the main tabs in the sample application:

Tab	Functionality
Recent	To display recently performed searches.
Nearby	To display results of search performed with user's current location.
Other Location	To display results of search performed with other location.
Settings	To display information about the application and allow user to clear data from device.

2.1 Screen Flows

The following is the high level screen navigation flow for the application:



2.2 Views

2.2.1 Recent

This is the startup path in Rhoconfig.txt

```
# Startup page for your application
```





```
start_path = '/app/Search?type=recent'
```


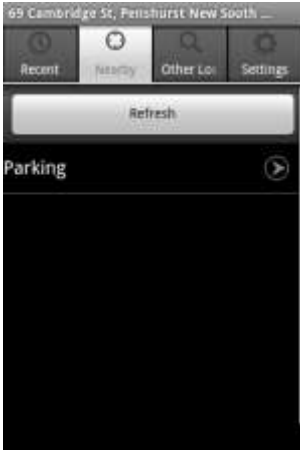

Description	iPhone View	Android View	Source Files
-------------	-------------	--------------	--------------

Modified: 1/26/2011 12:34 PM


6




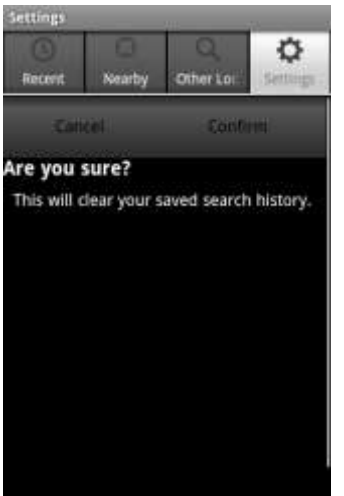
© 2011 Accenture LLP. All rights reserved. Any third-party names or trademarks contained in this document are the property of their respective owners.

<p>Recent: No previous search</p> <p>Upon Initial Load, controller.rb in “Search” folder launched.</p> <p>Index function is called to determine the functionality to perform.</p>			<p>recent.erb, controller.rb</p>
<p>Recent: previous searches</p> <p>Previously performed searches are displayed in this view.</p>			<p>listing.erb, controller.rb</p>

<p>Nearby:</p> <p>The current location is obtained via:</p> <p><i>GeoLocation.latitude</i></p> <p><i>GeoLocation.longitude</i></p> <p>The function “nearby_geo_callback1” attempts to get location and tries again by calling “nearby_geo_callback2” if the first call fails.</p> <p>When location is known, “geocode_other_location” is called to geocode the location (to display user friendly name instead of latitude / longitude)</p>			<p>listing.erb, controller.rb</p>
<p>Other Location:</p> <p>The “input_other_location.erb” view is used to capture user input.</p> <p>Search term entered is passed to Google Maps Geocoding API via the “input_other_location” function.</p> <p>If multiple potential results are found, they are listed for the user to select from with the “select_geocode_results.erb” view. When selected, the lat / long is used.</p> <p>When results found, listing.erb is used to</p>			<p>put_other_lo ation.erb, select_geocod _results.erb, controller.rb,</p>

			
<p>Showing results on Map:</p> <p>When the latitude and longitude is selected (either for current or other location), the “<i>map</i>” function is called.</p> <p>The latitude and longitude to search for is appended to the URL set in the rhoconfig.txt is referenced as “<i>Rho::RhoConfig.data_url_base</i>” and called via <i>Rho::AsyncHttp.get</i> function.</p> <p>The wait screen (<i>wait.erb</i>) is called until the application receives a callback.</p> <p>When the call results are returned, the “<i>display_mapped_data_callback</i>” function is triggered.</p> <p>The return data is in JSON format and automatically parsed into a Ruby hash object.</p>	 <p>The data within the “<i>parkingFacilityList</i>” object and the items for display in Title, subtitle and URL parameters are set to the <i>annotations</i> object.</p> <p>The “<i>MapView.create</i>” function is called to create the embedded map with the required parameters.</p>	 <p>Note: Title and subtitles are not displaying on Android. This is a known issue with Rhodes framework as of time of writing.</p> <p>“<i>map_params</i>” determines the settings for the map to be displayed.</p> <p>Map_type can be “standard”, “roadmap” but only “standard” seems to work for Android as of time of writing.</p>	Controller.rb , wait.erb,

<p>Details</p> <p>This view displays the details of the location which were parsed from the results of the web service call detailed previously.</p> <p>The details to be shown are appended to the URL parameter in the annotations object.</p> <p>CustomURIs are added to demonstrate additional functionality such as launching the phone dialer (for phone number) and alternative map view using browser (for address)</p>			<p>details.erb</p>
<p>Settings</p> <p>This functionality is in files the "Settings" folder.</p> <p>This view displays links to display information about the app and to reset the database.</p>			<p>Settings/index.erb</p>

<p>About</p> <p>Displays details of the application.</p>			<p>Settings/about.out.erb</p>
<p>Clear Recent Searches</p> <p>The <code>do_reset</code> function calls the</p> <p><code>Rhom::Rhom.database_full_reset</code> function to reset the database.</p>			<p>Settings/controller.rb</p>

2.3 Other Common Functionality:

2.3.1 Tabs:

The four main tabs are set in “application.rb”, which set out the action to perform with parameters to match the required action, ie show “Recent”, “Nearby”, “Other Location” or “Settings” views.

The icons for the tabs are in the `/public/images/tabs` folder.

2.3.2 CSS Stylesheets:

“Layout.erb” sets out the relevant CSS stylesheets to call for the detected device platform.

Modified: 1/26/2011 12:34 PM

11

© 2011 Accenture LLP. All rights reserved. Any third-party names or trademarks contained in this document are the property of their respective owners.

The CSS stylesheets for Android, Blackberry, iPhone and WindowsMobile are in the *public/css* folder.

2.3.3 Navbar

The “navbar” function detects if the device is iOS and uses the standard iOS navigation buttons for the application. For other device types, navigation buttons are created using HTML/CSS

This function is defined in both the Controller.rb in the “Search” and “Settings” folder.

Function usage:

```
navbar :title => <Title to show>, :left => { :action => url_for(:action => :<function to call>), :label
=> <Label for left button> }, (optional):right => { :action => url_for(:action => :do_reset), :label =>
<Label for right button> }
```

example:

```
navbar :title => "Clear Search History", :left => { :action => url_for(:action => :index), :label =>
"Cancel" }, :right => { :action => url_for(:action => :do_reset), :label => "Confirm" }
```

2.3.4 Menu items

The options that are brought up when the “menu” key is pressed (on devices that have it, such as Android) are set in *application.rb*.

```
@default_menu = {
  "Home" => :home,
  "Settings" => :options,
  # "Log" => :log,
  "Close" => :close,
}
```

These are generated by Rhomobile and can be customized for your application. This application does not use any synchronization with Rhosync, which is a data synchronization server from Rhomobile, hence the default “Sync” menuitem has been removed.

2.4 Data Persistence

In *Search/controller.rb*, in the *display_mapped_data_callback* function, after a search has been performed and the callback received, the following parameters are saved to the “Search” model in the database on the device:

- *latitude*
- *longitude*
- *location*
- *timestamp*

When starting the application, the *recent* function is called which uses the *find* command to retrieve the data persisted and display them if found.

2.5 Calling the Web Service

The “*data_url_base*” key in the “*rhoconfig.txt*” sets the URL of the web service the application calls to retrieve the data.

For this application, the following URL is called: *http://nyc-apps-poc.appspot.com/rest/json/*

That web service expects an input parameter of latitude and longitude.

The code adds “*parking_facility*” as a category and appends the latitude and longitude.

http://nyc-apps-poc.appspot.com/rest/json/parking_facility/<latitude>/<longitude>

Other search categories could be added later if the web service is updated to support it.

2.6 Data from Web Service

The data exposed from that web service has been loaded from the following dataset:

http://www.nyc.gov/html/datamine/downloads/data/parking%20facilities.csv

The following is a sample of the data retrieved in JSON format:

```
{ "parkingFacilityList": [ { "entityName": "CHELSEA ENCLAVE GARAGE  
LLC", "licenseNumber": "1345812", "addressBuilding": "177", "addressStreetName": "9  
AVENUE", "addressCity": "NEW  
YORK", "addressState": "NY", "addressZipCode": "10011", "facilityType": "Parking  
Garage", "camisTradeName": "", "addressLocation": "", "telephoneNumber": "2149562280", "  
numberOfSpaces": "36", "status": 1, "entityId": 146004, "latitude": 40.745045, "longitude": -  
74.00241, "distance": 0.12643275 }
```

The JSON data is automatically parsed in *Search/controller.rb* in the *display_mapped_data_callback* function for use in mapping and displaying the results.

2.7 Geolocation and Mapview

The geolocation and map is developed following the mapview sample on Rhomobile Wiki - <http://wiki.rhomobile.com/>