Android Developer's Guide

Version 1.0

Tuesday May 7, 2013

Revision Sign-Off

By signing the following, the team member asserts that he has read the entire document and has, to the best of his knowledge, found the information contained herein to be accurate, relevant, and free of typographical error.

Name	Signature	Date
Alex Anduss		
Baer Bradford		
Greg Kolesar		

Revision History

The following is a history of document revisions.

Version	Date Revised	Comments
1.0	May 7, 2013	Version 1.

Table of Contents

Revision Sign-Off i
Revision History ii
Table of Contents iii
1. Introduction 1
  1.1 Purpose 1
  1.2 Overview 1
2. Android Device Requirements 2
  2.1 Android Version 2
  2.2 Phones 2
  2.3 Android Tablets 2
  2.4 Network 2
  2.5 Wikipedia's List of Android devices and specifications 2
3. Data Model 3
  3.1 Account Singleton 4
  3.2 Survey 5
  3.3 Question 6
  3.4 Response 7
  3.5 Fort Worth Resource 7
  3.6 Health Education Resource 8
4. Android Web Service 9
  4.1 Web Service Singleton 10
  4.2 Create new GET call 11
  4.3 Create new POST call 11
5. Adding/Removing from the Main Menu 13
  5.1 The Main Menu Arrays 13
6. Submitting to the App Store 14
7. Testing on a device 15

1. Introduction

1.1 Purpose

The purpose of this document is to aid any developers with this system through modifying the app, deploying test versions to devices, and deploying updates to the App Store.

1.2 Overview

GoodNEWS Fort Worth is a program initiated by the University of North Texas Health Science Center in conjunction with Fort Worth City Leaders to promote healthy living in the city. In the past, the program had been a series of seminars that were used to assist citizens with their lifestyle. The app will be a great way for the GoodNEWS team to reach out and help the citizens of Fort Worth better their lifestyles. The app will also allow the citizens to respond to surveys and the GoodNEWS team can use the results to make Fort Worth a healthier place.

2. Android Device Requirements

This section lists the requirements needed to run the GoodNEWS Fort Worth App.

2.1 Android Version

Android Version 4.1.1 ("Jelly Bean") and higher are required to use this app.

2.2 Phones

Any Android phone that supports Android version 4.1.1 ("Jelly Bean") will be able to run this app.

2.3 Android Tablets

This app will run on any Android tablets that support Android version 4.1.1.

2.4 Network

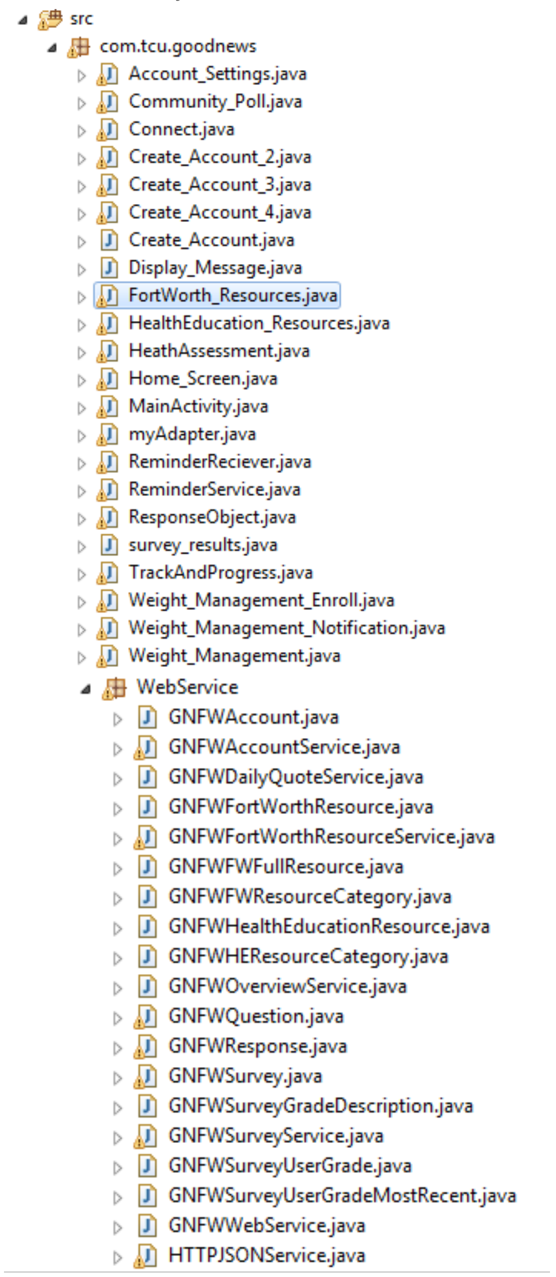
The App does require constant connection to a 3G, 4G, or LTE cellular, or a Wi-Fi network. If for some reason you lose network capability, you will be logged out of your account to avoid security and data integrity issues.

2.5 Wikipedia's List of Android devices and specifications

http://en.wikipedia.org/wiki/Android_devices#Smartphones

3. Data Model

This section lays out the Data Model as seen in **Figure 3.1**.

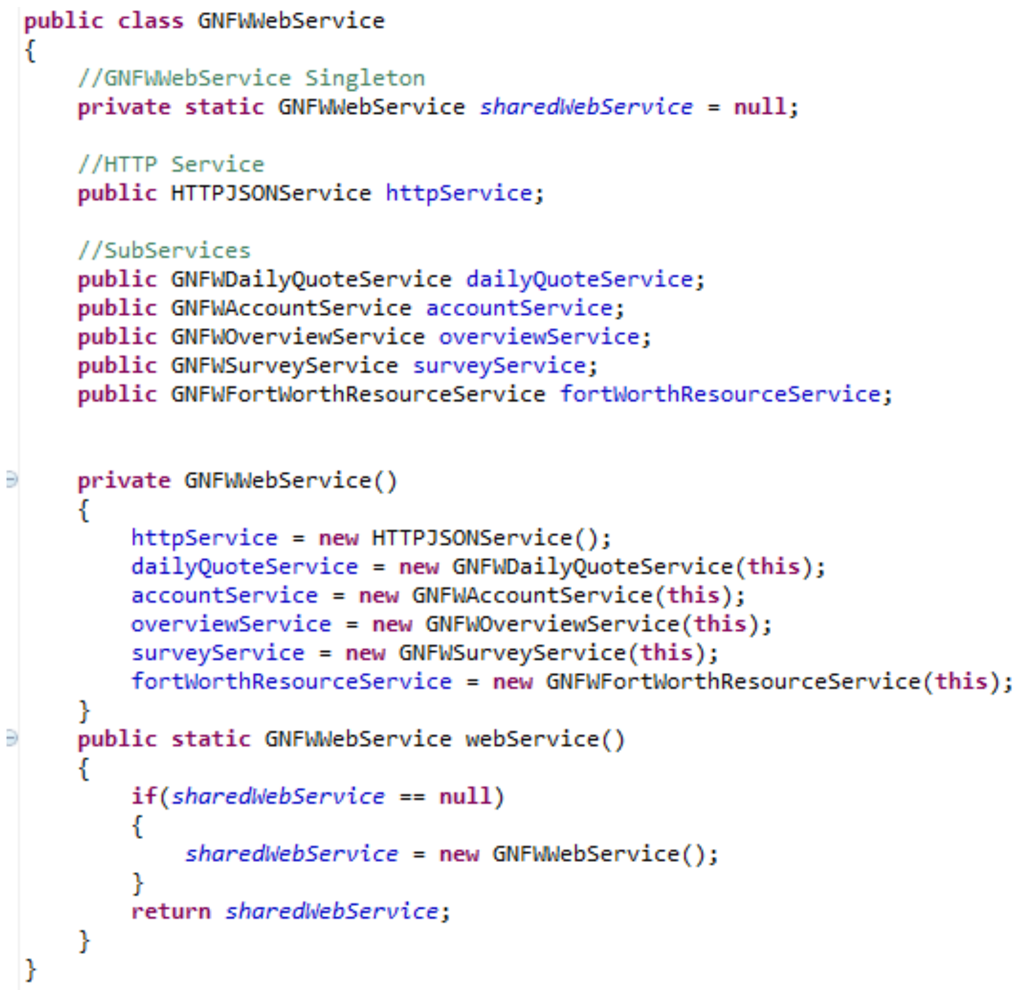


```
src
  com.tcu.goodnews
    Account_Settings.java
    Community_Poll.java
    Connect.java
    Create_Account_2.java
    Create_Account_3.java
    Create_Account_4.java
    Create_Account.java
    Display_Message.java
    FortWorth_Resources.java
    HealthEducation_Resources.java
    HeathAssessment.java
    Home_Screen.java
    MainActivity.java
    myAdapter.java
    ReminderReciever.java
    ReminderService.java
    ResponseObject.java
    survey_results.java
    TrackAndProgress.java
    Weight_Management_Enroll.java
    Weight_Management_Notification.java
    Weight_Management.java
    WebService
      GNFWAccount.java
      GNFWAccountService.java
      GNFWDailyQuoteService.java
      GNFWFortWorthResource.java
      GNFWFortWorthResourceService.java
      GNFWFWFullResource.java
      GNFWFWResourceCategory.java
      GNFWHealthEducationResource.java
      GNFWHEResourceCategory.java
      GNFWOverviewService.java
      GNFWQuestion.java
      GNFWResponse.java
      GNFWSurvey.java
      GNFWSurveyGradeDescription.java
      GNFWSurveyService.java
      GNFWSurveyUserGrade.java
      GNFWSurveyUserGradeMostRecent.java
      GNFWWebService.java
      HTTPJSONService.java
```

Figure 3.1

3.1 Account Singleton

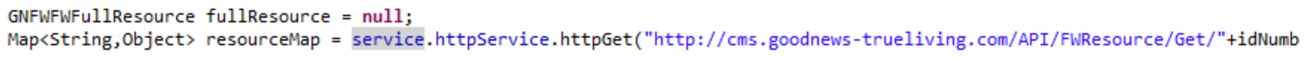
When the user creates their account or logs in, all of the important user data is stored in a single account object, or the account singleton. When a user creates their account and all of the information is submitted to the CMS, there is a call made to grab all of the account information. The JSON Parser automatically creates a dictionary of this object which is then stored in the accountDictionary variable (instance variable). **Figure 3.2** will show the header files and the getter methods that can be used to get information from the singleton.



```
public class GNFWWebService
{
    //GNFWWebService Singleton
    private static GNFWWebService sharedWebService = null;

    //HTTP Service
    public HTTPJSONService httpService;

    //SubServices
    public GNFWDailyQuoteService dailyQuoteService;
    public GNFWAccountService accountService;
    public GNFWOverviewService overviewService;
    public GNFWSurveyService surveyService;
    public GNFWFortWorthResourceService fortWorthResourceService;


    private GNFWWebService()
    {
        httpService = new HTTPJSONService();
        dailyQuoteService = new GNFWDailyQuoteService(this);
        accountService = new GNFWAccountService(this);
        overviewService = new GNFWOverviewService(this);
        surveyService = new GNFWSurveyService(this);
        fortWorthResourceService = new GNFWFortWorthResourceService(this);
    }
    public static GNFWWebService webService()
    {
        if(sharedWebService == null)
        {
            sharedWebService = new GNFWWebService();
        }
        return sharedWebService;
    }
}
```

Figure 3.2 GNFWAccount.h

Figure 3.3 demonstrates a few method calls to get information from the singleton.



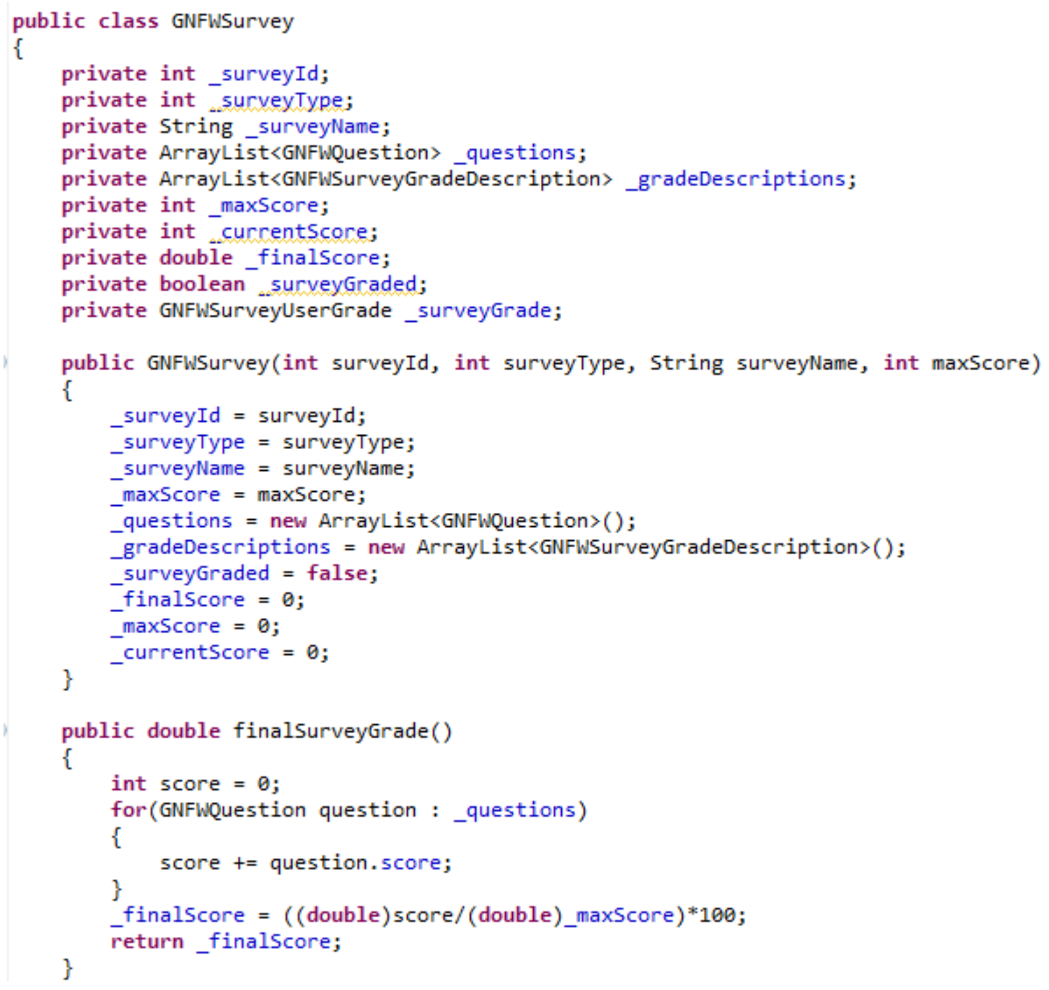
```
GNFWFWFullResource fullResource = null;
Map<String,Object> resourceMap = service.httpService.httpGet("http://cms.goodnews-trueliving.com/API/FWResource/Get/"+idNumb
```

Figure 3.3 Sample Calls to the Singleton

The call will always start with [GNFWAccount account] which gets the instance of the singleton, and then any of the methods in GNFWAccount.

3.2 Survey

There are three parts to the data structure of surveying. The main survey class is GNFWSurvey. This is seen in **Figure 3.4.**



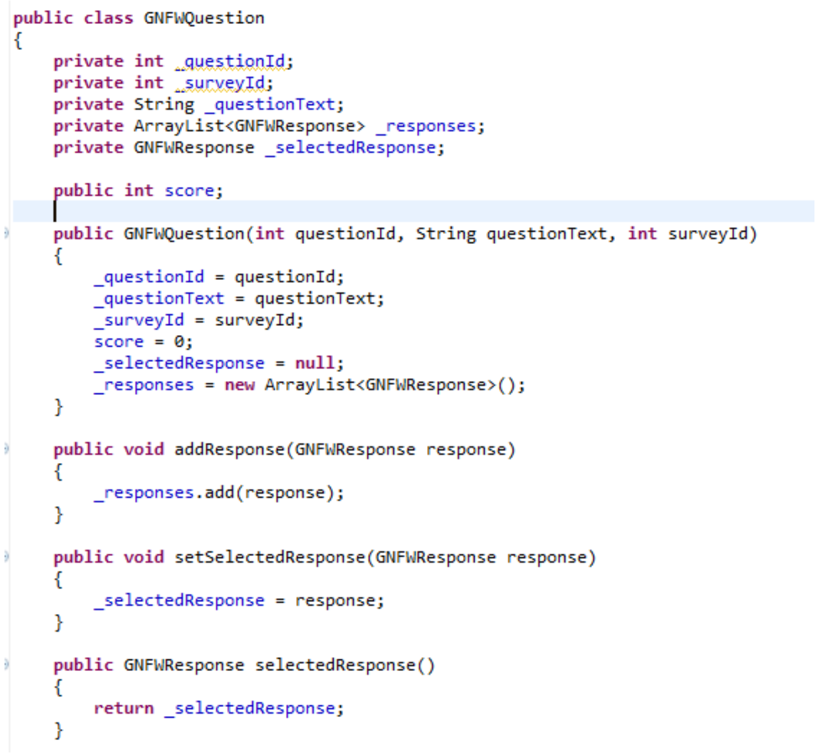
```
public class GNFWSurvey
{
    private int _surveyId;
    private int _surveyType;
    private String _surveyName;
    private ArrayList<GNFWQuestion> _questions;
    private ArrayList<GNFWSurveyGradeDescription> _gradeDescriptions;
    private int _maxScore;
    private int _currentScore;
    private double _finalScore;
    private boolean _surveyGraded;
    private GNFWSurveyUserGrade _surveyGrade;

    public GNFWSurvey(int surveyId, int surveyType, String surveyName, int maxScore)
    {
        _surveyId = surveyId;
        _surveyType = surveyType;
        _surveyName = surveyName;
        _maxScore = maxScore;
        _questions = new ArrayList<GNFWQuestion>();
        _gradeDescriptions = new ArrayList<GNFWSurveyGradeDescription>();
        _surveyGraded = false;
        _finalScore = 0;
        _maxScore = 0;
        _currentScore = 0;
    }

    public double finalSurveyGrade()
    {
        int score = 0;
        for(GNFWQuestion question : _questions)
        {
            score += question.score;
        }
        _finalScore = ((double)score/(double)_maxScore)*100;
        return _finalScore;
    }
```

Figure 3.4 GNFWSurvey

GNFWSurvey is the basic survey object. It contains the survey data, and the questions. While the user is taking a survey, the object also keeps track of how the survey is progressing, like if it has started, finished, or what was the last question the user was on. This object is used for the Community Vote and Poll section since there is no need to grade them. For the Track and Progress and Health Assessment surveys, there is the GNFWAssessmentSurvey object. This is a subclass of GNFWSurvey which means it inherits everything GNFWSurvey does, but allows for more. It leads to simpler code. GNFWAssessmentSurvey handles all of the grading of a survey that was just taken. It also stores the average grade, most recent grade, and the most recent time that a survey was taken. The final surveying object is the GNFWSurveyGradeDescriptions object. This object is used to store the upper and lower bound of a survey grade range and it also hold the description that goes along with these grades. The grade descriptions are stored in the GNFWSurvey object’s gradeDescriptions variable.

3.3 Question

The GNFWQuestion object is a simple object as seen in **Figure 3.5**.



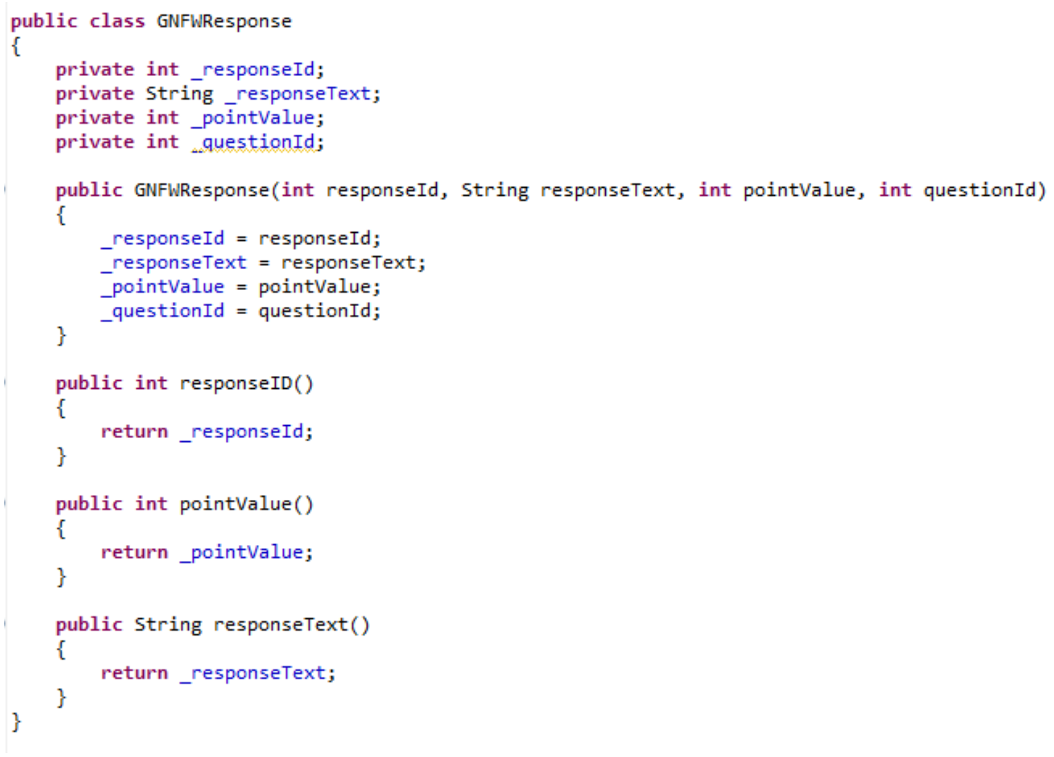
```
public class GNFWQuestion
{
    private int _questionId;
    private int _surveyId;
    private String _questionText;
    private ArrayList<GNFWResponse> _responses;
    private GNFWResponse _selectedResponse;

    public int score;

    public GNFWQuestion(int questionId, String questionText, int surveyId)
    {
        _questionId = questionId;
        _questionText = questionText;
        _surveyId = surveyId;
        score = 0;
        _selectedResponse = null;
        _responses = new ArrayList<GNFWResponse>();
    }

    public void addResponse(GNFWResponse response)
    {
        _responses.add(response);
    }

    public void setSelectedResponse(GNFWResponse response)
    {
        _selectedResponse = response;
    }

    public GNFWResponse selectedResponse()
    {
        return _selectedResponse;
    }
```

Figure 3.5 GNFWQuestion

This object holds an array of responses, the question text, and it also has the parent survey which makes it easy to traverse the web of survey-question-responses. It also holds a pointer to the selected response which makes it easy for the developer to get the response id for the CMS.

3.4 Response

The response object is also simple. See **Figure 3.6**.



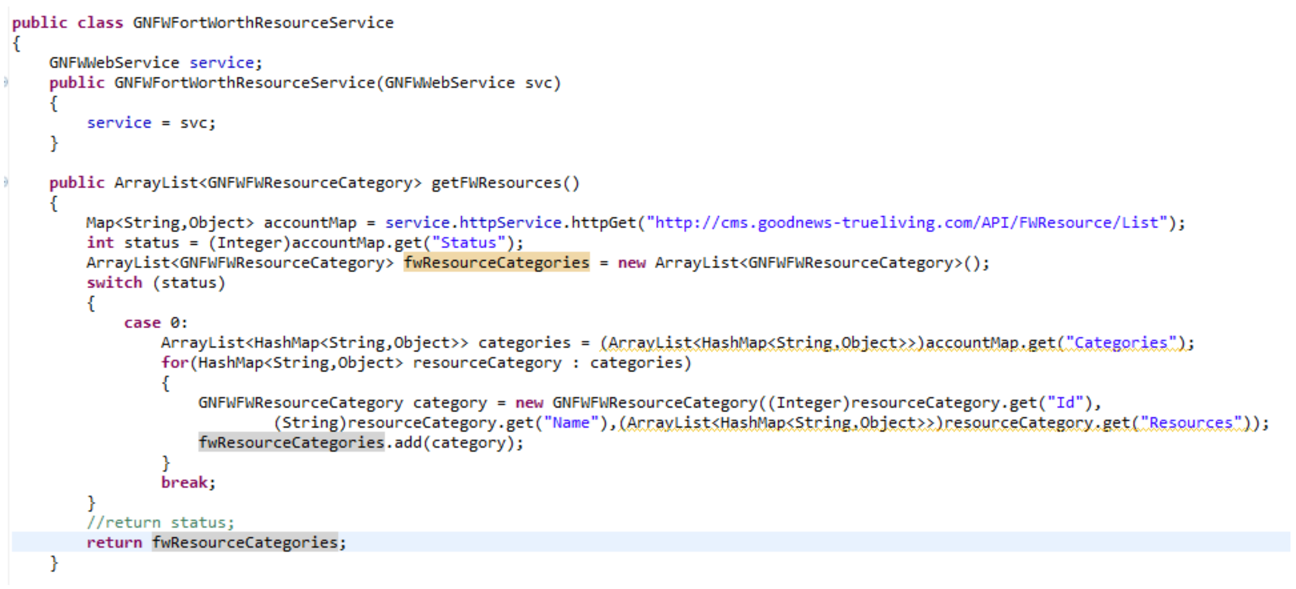
```
public class GNFWResponse
{
    private int _responseId;
    private String _responseText;
    private int _pointValue;
    private int _questionId;

    public GNFWResponse(int responseId, String responseText, int pointValue, int questionId)
    {
        _responseId = responseId;
        _responseText = responseText;
        _pointValue = pointValue;
        _questionId = questionId;
    }

    public int responseID()
    {
        return _responseId;
    }

    public int pointValue()
    {
        return _pointValue;
    }

    public String responseText()
    {
        return _responseText;
    }
}
```

Figure 3.6 GNFWResponse

This object holds the response id, response text, point value of the response for grading, and also the parent question, which also can be used to get the parent survey.

3.5 Fort Worth Resource

Fort Worth Resources are slightly complicated due to the need for Geoencoding. There is a handy wrapper class for it to keep the ugly code from being exposed to the developer.



```
public class GNFWFortWorthResourceService
{
    GNFWWebService service;
    public GNFWFortWorthResourceService(GNFWWebService svc)
    {
        service = svc;
    }

    public ArrayList<GNFWFWResourceCategory> getFWResources()
    {
        Map<String,Object> accountMap = service.httpService.httpGet("http://cms.goodnews-trueliving.com/API/FWResource/List");
        int status = (Integer)accountMap.get("Status");
        ArrayList<GNFWFWResourceCategory> fwResourceCategories = new ArrayList<GNFWFWResourceCategory>();
        switch (status)
        {
            case 0:
                ArrayList<HashMap<String,Object>> categories = (ArrayList<HashMap<String,Object>>)accountMap.get("Categories");
                for(HashMap<String,Object> resourceCategory : categories)
                {
                    GNFWFWResourceCategory category = new GNFWFWResourceCategory((Integer)resourceCategory.get("Id"),
                            (String)resourceCategory.get("Name"),(ArrayList<HashMap<String,Object>>)resourceCategory.get("Resources"));
                    fwResourceCategories.add(category);
                }
                break;
        }
        //return status;
        return fwResourceCategories;
    }
```

Figure 3.7 GNFWFortWorthResourceArray

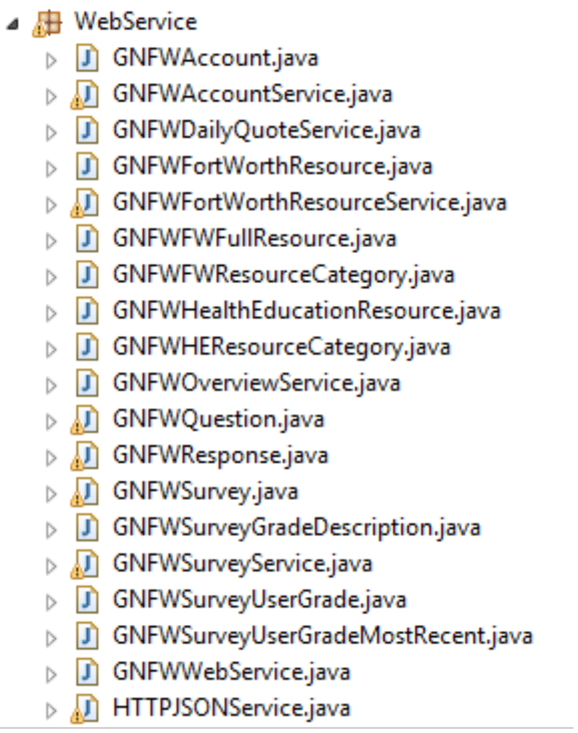
The GNFWFortWorthResourceArray class holds the array of all the Fort Worth Resources for a certain section. The object is first initialized in the web service call when the user clicks on Fort Worth Resources. To make it simple for the developer to add objects to the array, they do not have to create an AddressObject themselves, the developer just has to call to add a resource and then it will create the AddressObject and add it to the array. Due to some last minute changes, the updateResourceWithName method is no longer used. It was removed to improve efficiency with the CMS call and memory management on the devices.

3.6 Health Education Resource

A Health Education Resource object consists of a Name and a URL.

4. Android Web Service

This section will lay out how to use the web service. **Figure 4.1** shows all of the classes contained in the web service.

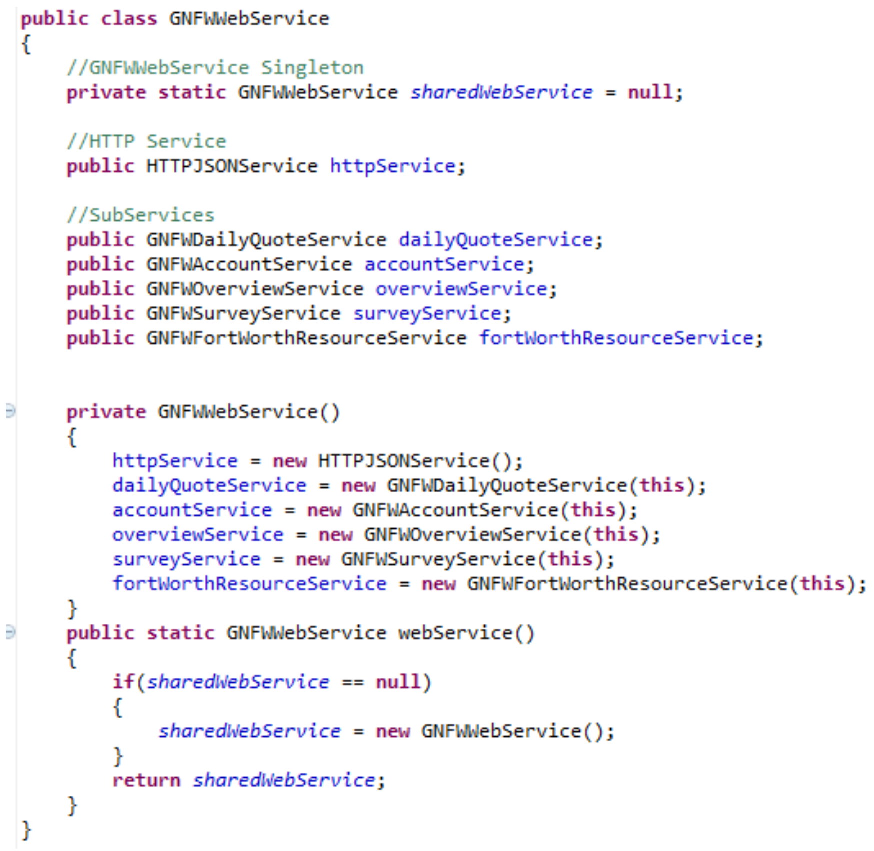


WebService
- GNFWAccount.java
- GNFWAccountService.java
- GNFWDailyQuoteService.java
- GNFWFortWorthResource.java
- GNFWFortWorthResourceService.java
- GNFWFWFullResource.java
- GNFWFWResourceCategory.java
- GNFWHealthEducationResource.java
- GNFWHEResourceCategory.java
- GNFWOverviewService.java
- GNFWQuestion.java
- GNFWResponse.java
- GNFWSurvey.java
- GNFWSurveyGradeDescription.java
- GNFWSurveyService.java
- GNFWSurveyUserGrade.java
- GNFWSurveyUserGradeMostRecent.java
- GNFWWebService.java
- HTTPJSONService.java

Figure 4.1 Web Service Class list

4.1 Web Service Singleton

Using the web service singleton is important to the flow of the program. The singleton and the classes that interact with it support many features including asynchronous calls, fire-and-forget calls, and callbacks. It is simple to use the singleton object. The first thing you want to do when creating a new class for a different service is to create the property in the interface file, **Figure 4.2** shows the existing properties and the new one will just follow that format.



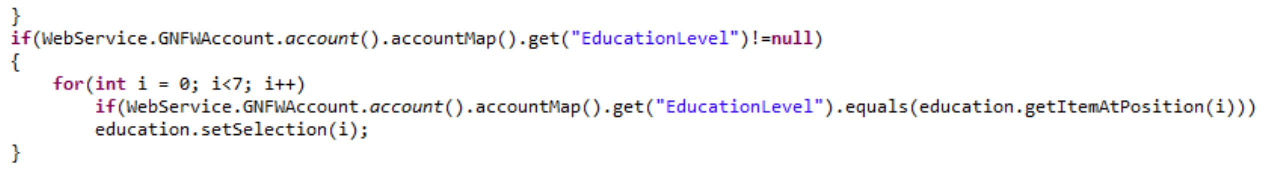
```
public class GNFWWebService
{
    //GNFWWebService Singleton
    private static GNFWWebService sharedWebService = null;

    //HTTP Service
    public HTTPJSONService httpService;

    //SubServices
    public GNFWDailyQuoteService dailyQuoteService;
    public GNFWAccountService accountService;
    public GNFWOverviewService overviewService;
    public GNFWSurveyService surveyService;
    public GNFWFortWorthResourceService fortWorthResourceService;


    private GNFWWebService()
    {
        httpService = new HTTPJSONService();
        dailyQuoteService = new GNFWDailyQuoteService(this);
        accountService = new GNFWAccountService(this);
        overviewService = new GNFWOverviewService(this);
        surveyService = new GNFWSurveyService(this);
        fortWorthResourceService = new GNFWFortWorthResourceService(this);
    }
    public static GNFWWebService webService()
    {
        if(sharedWebService == null)
        {
            sharedWebService = new GNFWWebService();
        }
        return sharedWebService;
    }
}
```

Figure 4.2 GNFWCMSWebService

By initializing all of the separate service classes in the singleton, you are able to access the methods contained in these classes from anywhere. This is important, because it only creates a single instance of these objects, and the developer does not need to worry about ARC (Automatic Reference Counting), or garbage collection, from destroying your instances of these objects before they can call back.

4.2 Create new GET call

Getting information from the content management system efficiently is quintessential to the functionality of the app. To ensure that a stable connection is established, a single method was created in the HTTPJSONService class to get the data from the CMS. The JSONService was already created in the singleton, and it contains an exposed method, which takes the URL from the CMS API, the http method. Refer to **Figure 4.4.**



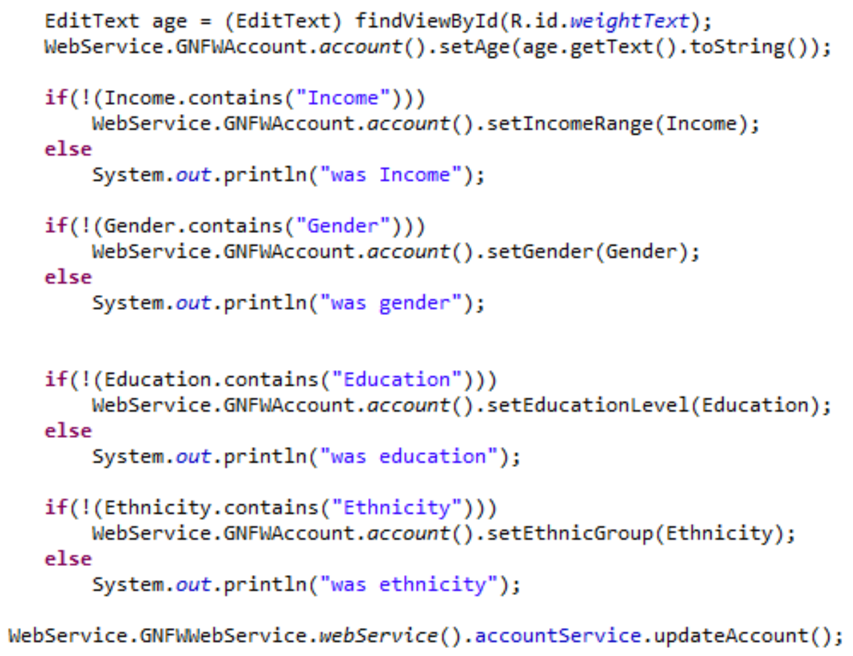
```
}
if(WebService.GNFWAccount.account().accountMap().get("EducationLevel")!=null)
{
    for(int i = 0; i<7; i++)
        if(WebService.GNFWAccount.account().accountMap().get("EducationLevel").equals(education.getItemAtPosition(i)))
        education.setSelection(i);
}
```

Figure 4.4 Example GET Call

Figure 4.4 shows an example GET call. Here you can see how the handler works. If there is no error, it will come back as null, and all of the errors are handled on the back end. The context is not used by any calls, but it can be used to store any additional data that may need to come back. Finally, the sender is the original class that made the call, and it allows for callbacks. When the web call finishes, everything will return here. Again, you can check to see if there was an error, and handle anything that may be needed there. You can use the built in JSON parser with the received data to generate appropriate objects when you get the JSON object back from the parser.

4.3 Create new POST call

Creating a POST call is very similar to a GET call. The only difference is for httpMethod, you need to put .set followed by the appropriate field to set, and you need to include the data from the JSON parsed dictionary. Also you must update the account with the updateAccount() function. **Figure 4.5** shows the general format of a POST call.



```
   EditText age = (EditText) findViewById(R.id.weightText);
   WebService.GNFWAccount.account().setAge(age.getText().toString());

   if(!(Income.contains("Income")))
       WebService.GNFWAccount.account().setIncomeRange(Income);
   else
       System.out.println("was Income");

   if(!(Gender.contains("Gender")))
       WebService.GNFWAccount.account().setGender(Gender);
   else
       System.out.println("was gender");


   if(!(Education.contains("Education")))
       WebService.GNFWAccount.account().setEducationLevel(Education);
   else
       System.out.println("was education");

   if(!(Ethnicity.contains("Ethnicity")))
       WebService.GNFWAccount.account().setEthnicGroup(Ethnicity);
   else
       System.out.println("was ethnicity");

WebService.GNFWWebService.webService().accountService.updateAccount();
```

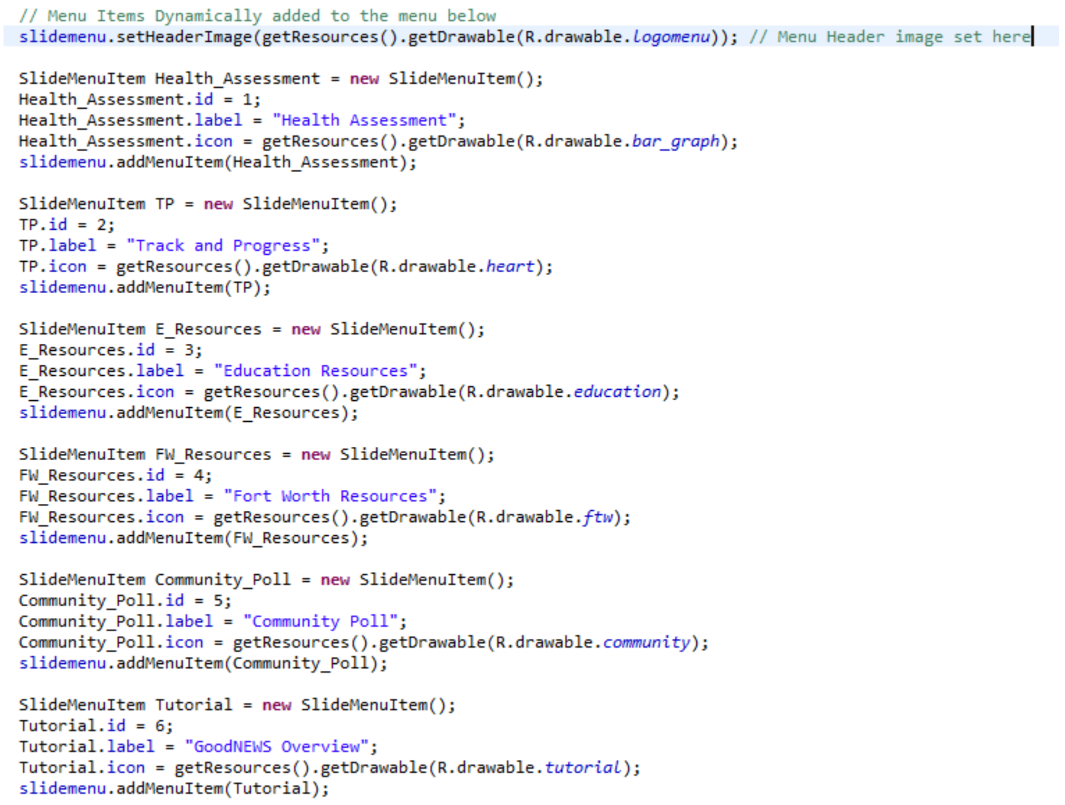
Figure 4.5 Example POST Call

5. Adding/Removing from the Main Menu

This section will give an overview to the developer on how to add/remove and modify components on the main menu.

5.1 The Main Menu Arrays

Look in the Home_Screen class, and in the onCreate method. There are a series of code blocks that set each menu item. Each menu item has an id, label, and icon to be set. The code below how each of these can be set for each menu item.



```java
// Menu Items Dynamically added to the menu below
slidemenu.setHeaderImage(getResources().getDrawable(R.drawable.logomenu)); // Menu Header image set here

SlideMenuItem Health_Assessment = new SlideMenuItem();
Health_Assessment.id = 1;
Health_Assessment.label = "Health Assessment";
Health_Assessment.icon = getResources().getDrawable(R.drawable.bar_graph);
slidemenu.addMenuItem(Health_Assessment);

SlideMenuItem TP = new SlideMenuItem();
TP.id = 2;
TP.label = "Track and Progress";
TP.icon = getResources().getDrawable(R.drawable.heart);
slidemenu.addMenuItem(TP);

SlideMenuItem E_Resources = new SlideMenuItem();
E_Resources.id = 3;
E_Resources.label = "Education Resources";
E_Resources.icon = getResources().getDrawable(R.drawable.education);
slidemenu.addMenuItem(E_Resources);

SlideMenuItem FW_Resources = new SlideMenuItem();
FW_Resources.id = 4;
FW_Resources.label = "Fort Worth Resources";
FW_Resources.icon = getResources().getDrawable(R.drawable.ftw);
slidemenu.addMenuItem(FW_Resources);

SlideMenuItem Community_Poll = new SlideMenuItem();
Community_Poll.id = 5;
Community_Poll.label = "Community Poll";
Community_Poll.icon = getResources().getDrawable(R.drawable.community);
slidemenu.addMenuItem(Community_Poll);

SlideMenuItem Tutorial = new SlideMenuItem();
Tutorial.id = 6;
Tutorial.label = "GoodNEWS Overview";
Tutorial.icon = getResources().getDrawable(R.drawable.tutorial);
slidemenu.addMenuItem(Tutorial);
```

Figure 6.1 Main Menu Information Arrays

6. Submitting to the App Store

In order to submit to the Google Play store, one will have to export there project to an apk file. They will also have to generate keys and sign their project. This can be accomplished by using keytool and Jarsigner. It is also recommended they use the zipalign tool to optimize the final APK package. It is easiest to use the built in functionality of eclipse. One can simply export a project from eclipse to an apk file and it will automatically generate keys, sign, and align your project for you. Then they must sign on to their Google account with developer permissions. There is a one-time fee of 25$ to upgrade a regular Google account to a developer account. Then they must create a new app section, write a description for their app, upload an icon picture, at least three screenshots from their app, and their apk file. To update their app, they will simply upload a new apk file. Google will save older apk versions to revert back to if necessary. It is important to update the android version code and version name in one’s manifest.xml file before each upload. It usually takes less than a day for an upload to be put on the app store.

7. Testing on a device

In order to test on a device, one must connect their android device up to their computer using a micro USB cord. Then when they click the Run button, they must select their device from the list of usable devices to test on. This will install the application with a debugging key.