HeatlhNews Documentation

I. Overview

HealthNews is a mobile application designed for use with the Google Android platform. It aims to deliver health related news to the reader via a clean and user friendly interface. In addition to providing news to the reader, the reader is designed to adapt to each individual users interests as indicated by their reading history. As usage of the application increases, it will become increasingly attuned to the users preferences.

II. High-Level Design

1. Main

In this section I will attempt to give a brief description of the design used for the application and an explanation of the design decisions used along the way. The application begins by launching Main.java an extension of the Android specific class Activity. This behavior is specified in the application's AndroidManifest.xml. As an implementation of Activity, this class must override the onCreate method which is called when an activity is started. This method is quite simple. It retrieves the appropriate layout (view), creates a StoryAdapter (which will be discussed in greater detail below), establishes click listeners, displays a progress dialog and starts a new thread which will go retrieve the desired stories.

Upon inspecting the Main.java source, it is important to keep in mind that much of the source contained here currently is for the implementation of future features and therefore not relevant to the current application functionality. This includes login and options menu functionality. These will be discussed in greater detail in the “Future Development Efforts” section.

Two other portions of the main class are important to the current application functions. First, notice that the click listener of type OnItemClickListener is the portion of the code that understands how to deal with a request (a click) and understands that the request must be routed to the StoryView class. The second item of note is the refreshNews method. This is the workhorse of the main application start page. Each time the page is loaded or reloaded, a thread which executes refreshNews is run. The method has a simple implementation which creates an instance of class StoryManager and calls it's method SetStories. We will cover the functionality of StoryManager next.

2. StoryManager

The purpose of the StoryManager class is to manager retrieval of the desired stories and sort them in the desired manner. The class has a single constructor, which requires the List of stories to be passed in (empty initially) and the location of the application's files directory. These two parameters warrant slight elaboration here. In the main class, an uninitialized list of stories is given as a paramter to the constructor StoryAdapter. This list of stories, whether empty or containing news items, will always be stories displayed to the user. Therefore, as we manage stories further down the application stack, we must pass this list with us always. It is the single list of which the user interface will be aware. Secondly, as we will discuss shortly, the application will require access to the device's file system. This path can be obtained from the application context, which can be retrieved from any class extending intent or activity. Seeing that our StoryManager is not an activity, it needs to be provided the file directory information from a class which is, i.e. the main.

The first purpose of the StoryManager is to create a List<Parseable>, simply a list of sites which implement the parseable interface, meaning we know how to parse stories from these sites. The is done within the SetStories method. Secondly, at the end of this method, the sort method is called, which organizes the stories based on the users previous reading habits. In order to perform this function, the open-source project Lucene is used heavily and therefore this will be discussed in greater detail in the "Third-Party Resources" section.

3. Parseable

The application requires that each site which is to provide news must implement the abstract class parseable and by extension the abstract method List<Story> refreshNews. The Parseable class contains several methods which can assist site specific implemenations in parsing html to obtain stories. These methods rely on the Jericho html parser which is an open source html parser. Jericho will be explained fully in the "Third-Party Resources" section.

4. Concrete Implementations of Parseable

Currently there are three concrete implemenations of Parseable. The implementation correspond to the sites which they parse, http://www.cnn.com/HEALTH/, http://www.nytimes.com/pages/HEALTH/, http://news.yahoo.com/health/. Each concrete implementation simply parses the stories and adds them to the provided collection.

5. StoryView

StoryView extends the google Android class Activity. The purpose of StoryView is to display the story selected by creating a WebView (google class) based on the url of the story clicked. The StoryView has one other repsponsibility, indexing the user's selected story for future reference. Once again, this involves Lucene interop and will be described below.

6. StoryAdapter

Google Android's ListView, which is the container comprimising the main view, allows you to set an ArrayAdapter which essentially takes an array of your desired object and formats it for display within the ListView. As such, the StoryAdapter class extends ArrayAdapter. It overrides the method getView of ArrayAdapter, which based on the view and position, creates each individual clickable story within the ListView.

7. Story

Story is simple class with a constructor requiring the stories title, site and url. It exposes getter methods for each field. The one interesting characteristic of this class is that it implements the java interface Comparable<T>. In this case it implements Comparable<Story>. This implementation allows stories to be sorted using the sort mechanisms of the collections class, i.e. *Collections.sort(stories);* This exact line actually appears in the StoryManager class. Classes which implement Comparable must override the method *compareTo(Object another)* This method is called when sorting to determine sort order. In this implementation we simply compare scores. Note that the scores are typically small decimals, so they are multiplied by 100 so that a meaningful int can be returned as required by the method contract.

III. Third-Party Resources

1. Lucene

Apache Lucene(TM) is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform. - lucene.apache.org

Lucene is described succinctly by the quote from the website (http://lucene.apache.org/java/docs/index.html) above. From a very simplistic viewpoint, Lucene allows you to index text for searching and then subsequently search against the index you have created. In this specific implementation, we handle indexing upon the user clicking a story. The StoryView class references the Indexer class (app-specific, not Lucene) which indexes each story. The searching takes place in the StoryManager class. The title of each story is retrieved from their respective websites and used as a query against the index. Future versions of the application will likely leverage more detail than contained in the title, perhaps using full documents to compare against the index. The Lucene functionality requires some slight tweeking to fit our specific use case. The searcher searches the index based on the query and returns the top documents in the index that match the given query. In our case, we want to score the query (title) as opposed to the documents in the index, so we then sum the score of each document and assign that value to the query string. This methodology will largely be replaced in futre development efforts as described below.

2. Jericho

Jericho HTML Parser is a java library allowing analysis and manipulation of parts of an HTML document, including server-side tags, while reproducing verbatim any unrecognised or invalid HTML. It also provides high-level HTML form manipulation functions. - jericho.htmlparser.net

Jericho's (http://jericho.htmlparser.net/docs/index.html) functionality is leveraged within the application to scrape the desired websites and extract the story information from each. HealthNews uses Jericho within the Parseable abstract class and within each of the site specific implementations of parseable. Jericho is primarily leveraged by creating an instance of the Source class and passing the url of the site as the parameter. Jericho uses the unit Segment, or inheriting from Segment, Element. These represent portions of the parsed source code. By identifying the tags and attributes that denote stories on each website, Jericho's methods can be leveraged to return the stories and create Story objects.

IV. Future Development Efforts

In order to deliver a viable appication with reasonable performance, certain comprises have been made with the design. Future releases will seek to mitigate some of these problems and improve the accuracy and efficiency of the sorting algorithms. Currently, the indexing, searching, scoring and sorting all take place on the mobile device at the time of user interaction. Although this approach is working well, it has certain limitations. The primary limitation is the level of comparison possible between the stories retrieved and the user's history. Subsequent releases will attempt to alleviate this problem by moving this processing from the device to a mobile computing cloud. This will enable not only additional computing power, but also (and more importantly) the ability to perform searching and sorting functions prior to user interaction and therfore be prepared to deliver pre-sorted content immediately upon application load.

The above shift in processing from the device to the computing cloud will not only improve the performance of the current functionality, but also enable more complex behavior. Of primary interest will be extending the querying of titles against the index to a more complete full document similarity algorithm.

In addition to the algorithm improvements, there are designs to provide the application with an options menu and potentially add support for additional sites.

V. Pre-Release Notes

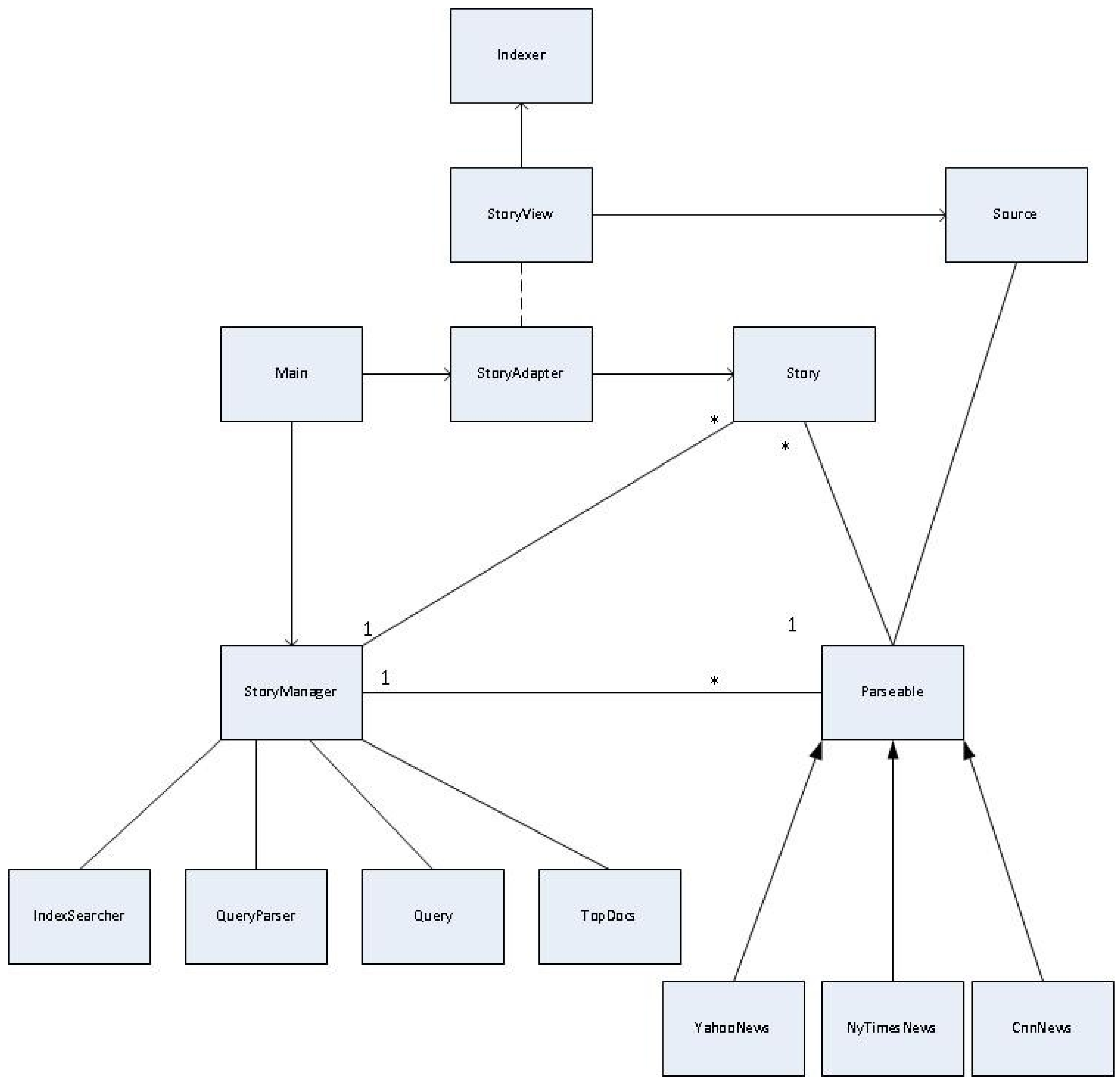
Currently, the application has a few idiosyncracies that are related to it being in a pre-release state. These are changes that could be made at any point to enable the application to be deployed and usable by users. The first and perhaps most obvious, is the retention of stories which are already read. The behavior of the sorting algorithm can often be very subtle. This is due in part to fact that the user does not know how the stories would be ordered without any sorting (no point of comparison) and second, because the titles of the stories are rarely directly correlated to indexed stories (typically only partial matches). In order to make the behavior more observable, stories that have been read are not removed from the list, creating a situation where read stories are sorted very highly in the display. This makes the algorithm's behavior easier to view, but is probably undesirable when offered to users. This will be mitigated via a simple boolean 'read' flag.

Finally, as mentioned above the options menu has yet to be implemented. The application currently does not even handle exceptions associated with trying to access the menu. This is desirable for debugging because it enables the developer to end the application completely (clearing any cached data), but obviously will need to be corrected for customer usage.

IV. Source Code

The source code is freely available on GitHub
https://github.com/tltjr/HealthNews

Appendix A – Basic Class Diagram


Indexer
StoryView
Source
Main
StoryAdapter
Story
*
*
1
1
1
*
StoryManager
Parseable
IndexSearcher
QueryParser
Query
TopDocs
YahooNews
NyTimes News
CnnNews