CptS 322 Spring 2018
Project B Description

Project Title: MapRSS: An RSS Reader client with Map Interface

Contents:

Introduction

Term project is an important component in a software engineering class. You will complete the project in a team environment. The project is to provide a platform on which you can exercise methodologies and principles in Software Engineering. Your project will be to analyze, design, document, build, and release a standalone, desktop RSS client that visualizes location-enabled RSS feeds on map interface (see Figure-2). Your team will analyze the requirements, design the client and also implement the design. The outcomes of project will be demonstrated not just by the quality of your design and the final product, but also by the quality of software process and the documents developed during the course of the project.

Required Features

This handout describes the **MapRSS,** an RSS reader with a map interface.

The MapRSS client downloads *articles* (such as, news headlines, blog entries, etc.) from the RSS feeds and displays them. The articles are organized into *feeds*. A feed is identified with a URL, and, at any one time, accessing that URL returns a file containing the articles. Each article typically has a title, a summary, and a link to a web page for further information. The feed file is formatted in XML, using one of several RSS standard formats. (To see an example of a feed, look at http://www.cnn.com/services/rss/ which is an RSS 2.0 feed for http://www.cnn.com)

MapRSS offers three interfaces to browse and read the RSS articles:

1. A standard RSS reader interface where articles are organized into feeds

2. A map-based interface where articles are organized based on their locations on a map. The links to articles are displayed on the map, as clickable icons. The MapRSS client will search the RSS feeds looking for place names and will display items on the map when they’re locatable.

3. A topic-based interface, where articles from different sources are put into different categories/topics, that are defined by the user, using user provided keywords.

As new articles enter the feed, older articles are typically dropped out. Different sources update their feeds at different rates; a client usually polls the feed to determine whether it has been updated.

This project is very open ended; you are free to develop your MapRSS client with your own choice of features. Nevertheless, a base set of features is required:

1. **Subscription to feeds.** The user can enter feed addresses, associate informal names with them, and organize them into groups (usually called *channels*). The user can select the update period for individual feeds, which determines how often the client polls them. The collection of feeds, their names, organization into channels, and update periods can be saved as a config file, or loaded from a previously saved config file.

2. **Display of articles.** MappRSS offers three interfaces to browse and read the RSS articles:
   a) **Main interface:** This is the main interface to browse and read the articles. The user can display the titles, dates and summaries of articles in a feed, as well as the full articles (which will in general require rendering webpages). A visual indication shows which articles have been read, and users have the option of only displaying articles that have not yet been read. The user should also be able to reverse the "read" articles back to "unread", which in turn changes the visual indication as well. The client must be capable of reading feeds in the most common formats (RSS 2.0 for information on RSS formats: http://msdn.microsoft.com/en-us/magazine/cc163989.aspx).
   In addition, the user can enter RSS subscriptions, manage/organize feeds into channels/collections, and setup preferences through this interface. An example interface screenshot is shown in Figure-1.

Figure 1 - Example screenshot for the MapRSS main interface

   b) **Map Interface:** The user can alternatively choose to view the article links on a map interface, where the article links appear as icons on map, organized based on the location of the article. The MapRSS client will search the RSS feeds looking for place names and will display items on the map when they're locatable. You need to develop a location decoding algorithm, which reads the entries (title and description) of an RSS feed and tries to extract a location (i.e., the latitude and longitude) for the entry text. If a relevant location is found, a clickable icon is placed on the map. When the user click on a particular icon, your interface should bring up a pop-up message displaying the title and description of the article, along with a link to the article. The user can view the article in the "main interface" by clicking that link.

The client can use the GeoNames Webservices for obtaining the location names (http://www.geonames.org/export/ws-overview.html). GeoNames is a geographical database under a Creative Commons attribution license (cc-by) containing millions of geographical names and features. For simplicity, the client needs to locate the RSS items for the United States only.

A local database that stores all state names/abbreviations, and city names along with their coordinates in the United States will be provided. Your location decoding algorithm will search this database for the words in the RSS entry **titles** to find out the relevant location for the entry. Note that for most entries, no location word appears in the title, therefore no location can be decoded. More details on building and searching the location database will be provided.

For the map interface, the MapRSS client will integrate the Bing Map Plug-in (available with .NET), or NASA World Wind (http://worldwind.arc.nasa.gov/).

- Example for using Bing Map, as well as specifications on the SDK are available on MSDN. Here is the link to a tutorial page: http://msdn.microsoft.com/en-us/library/dd221354.aspx
- For guidance using NASA World Wind, refer to the SDK provided at: http://builds.worldwind.arc.nasa.gov/worldwind-releases/1.5/docs/api/index.html

Note: *For people who want to use Google Map/Earth, they can try the Google Earth plug-in at http://www.google.com/earth/explore/products/plugin.html.*

Figure-2 and Figure-3 exemplify a possible map interface integrating Google Earth plug-in.

Note that there is another emerging standard, named GeoRSS, for encoding location as part of a RSS Web feed. GeoNames offers the RSStoGeoRSS web service (http://www.geonames.org/rss-to-georss-converter.html) which can geocode RSS content from any RSS web link and return the content as GeoRSS. **As mentioned above, your RSS client will implement its own location decoding algorithm**. However, RSStoGeoRSS can be used to check the accuracy of your location decoding algorithm.

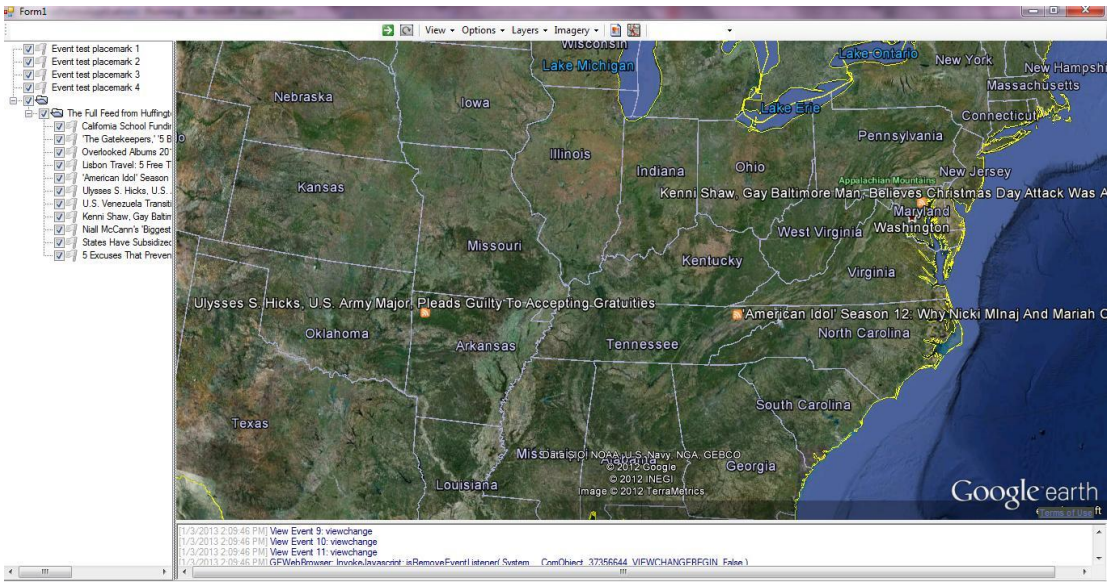

Figure 2 - An example interface to view RSS entries on map (Google Earth). Three RSS entries are marked as orange icons.

c) Topic Interface: Similar to the Main Interface, but in this interface, the articles from multiple RSS sources will be put into different categories of "topics". All the topics will be provided by the user, along with one or more keywords associated with the topics. For instance, the user can

define a topic named “Sports”, and setting keywords with: NFL, NBA, Tennis; Then, the client will search the content of all articles from all RSS feeds and see if they contain such keywords. If so, these articles are displayed under the “Sports” node. If none of the articles contains the keywords, then it simply displays nothing. Also note that, the same articles can belong to multiple topics. For example, a news piece about Coug Football can belong to both “Sports” and “WSU”.

The topics and keywords should also be saved within a config file. You can save it into the same config file as in Feature 1, or into a different file.

3. **Time Filtering:** All three interfaces should allow the user to view the articles on/within certain dates and/or time only. For example, if the user wants to see the most recent news articles posted within the last 2 hours, the main interface should filter the RSS feeds accordingly. Similarly on the map interface, only the icons for the articles posted within the last 2 hours should be displayed.

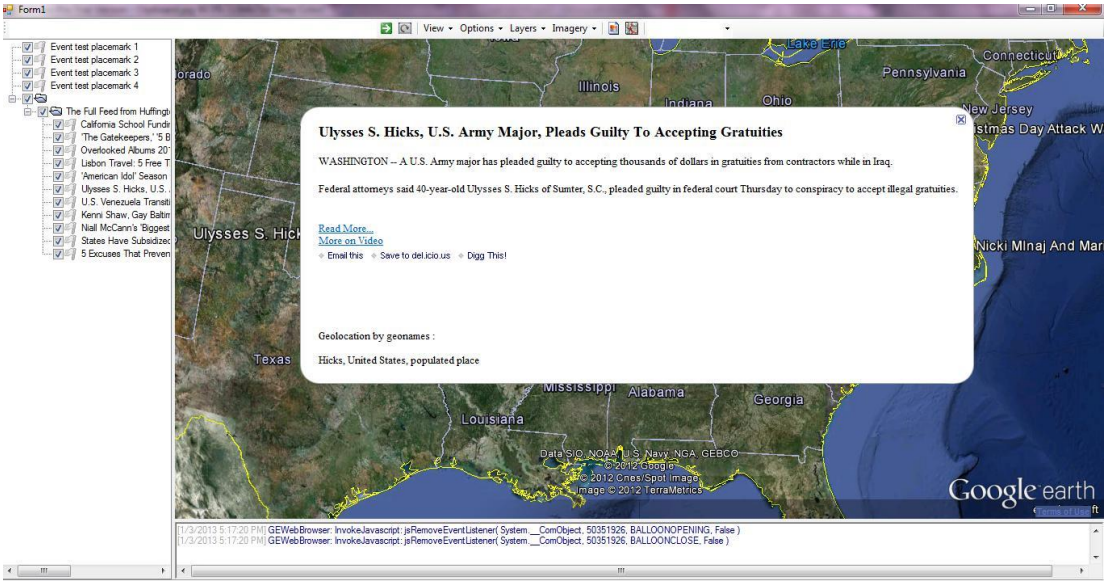

Figure 3 - When the user click on a particular icon on the map, a pop-up message displaying the title and description of the article, along with a link to the original article appears.

Performance Metrics

Project Grading:

The weights of the project deliverables are as follows:

1. Project Features (50%)
   within which,
   a. Feature 1…….. 10%
   b. Feature 2.a……....10%
   c. Feature 2.b…….15%
   d. Feature 2.c……..10%
   e. Feature 3…….5%
2. The quality of your documents submission along the way, including but not limited to use case diagrams, class diagrams, etc. This also includes the quality of your code, i.e. how well

you utilize the concept we discuss in class, such as OO programming approach and basic design principles. (40%)

3. Project management in section 3.2, i.e., on time with Milestones and updates, regular and efficient usage of GITHUB. (10%)

Extras:

4. Well designed and implemented GUI. (10%)
5. Extra credit: 10%. Demonstrate to me why you should earn this 10%.

Final demonstrations will be scheduled around the end of this semester to show the capabilities of your product. You will demo to me the features of your program, run some tests, and the grading will be based upon your adherence to the requirement, robustness (i.e., not easily broken), and designs of your code, among others.

Each team will get a team grade.
Then, each individual will get an individual grade for you project based on peer reviews from your team mates. 20% of your team grades will be used to adjust your individual grades.

Your overall course grade will not exceed 100 though.

Notes

1. Programming Language

C# is strongly recommended. (Specifically, I recommend using WPF when creating your project.) If you want to use other languages, do understand it is likely to be more challenging.

2. Academic Integrity

No "copy-pasting" from other teams, or any kind of plagiarism will be allowed, except specifically approved by me of using some open-source codes. Once found, all members in involving teams will immediately get an "F" for this course. Please take this seriously.

Available open-source projects could only be used for supplemental functions of your program, for example, a JSON/XML parser. Use other sources as reference if needed, but do not use entire segments of codes or any other behaviors that could be considered plagiarism.

Preliminary Resources

1. Bing Maps APIs: https://msdn.microsoft.com/en-us/library/dd877180.aspx

   Bing Maps Project tutorials: https://msdn.microsoft.com/en-us/library/dd221354.aspx

2. Git branching practice: https://www.atlassian.com/git/tutorials/using-branches

3. About .gitignore: https://www.atlassian.com/git/tutorials/gitignore

   Remember to ignore the typical Visual Studio/Eclipse files!!!

4. Please see the "SyndicationFeed" class for parsing RSS feed in the C# .Net library. You are not required to, and you shouldn't write the parser by yourselves.

5. Language-Integrated Query (LINQ): https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/introduction-to-linq
   Use LINQ to read data from the csv file.

Additional References

[1] GeoNames Web services. http://www.geonames.org/export/ws-overview.html

[2] Google Earth Plug-in (http://www.google.com/earth/explore/products/plugin.html

[3] NetBeans IDE http://netbeans.org

[4] RSS Wiki http://en.wikipedia.org/wiki/RSS