CptS 323 – Software Design
Spring 2013
Project Description

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

Contents:
- Introduction
- Required Features
- Performance Metrics
- Awards and Extra Credits
- Project Management and Milestones
- References

Introduction

Term project is an important component in a software design class. You will complete the project in a team environment (each team will have max 3, min 2 students). The project is to provide a platform on which you can excise software design and development methodologies in Software Engineering. Your project will be to 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 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 design 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.) You can also try Google's online RSS reader for free.

MapRSS offers two interfaces to browse and read the RSS articles:
1. A standard RSS reader interface (similar to Google's online RSS reader ) 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.
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 two 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 client must be capable of reading feeds in the most common formats (RSS 1.0, RSS 2.0 and Atom 1.0 – for information on RSS formats: [http://msdn.microsoft.com/en-us/magazine/cc163989.aspx](http://msdn.microsoft.com/en-us/magazine/cc163989.aspx)). In addition, the user can enter RSS subscriptions, manage/organize feeds into channels, and setup preferences through this interface. An example interface screenshot is shown in Figure-1.

   ![Figure 1 - Example screenshot for the MapRSS main interface](image)

   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 will 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 need to locate the RSS items for the United States only. You need to build a local database to store all state names/abbreviations, and city names along with their coordinates in the United States. 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 Google Earth Plug-in (http://www.google.com/earth/explore/products/plugin.html). A sample code (see the link below) for embedding Google Earth plug-in in a C# forms application is available with GNU Lesser General Public License.

winforms-geplugin-control-library : A library of controls to work with the Google Earth Plugin API in managed code. http://code.google.com/p/winforms-geplugin-control-library/

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. More information on location decoding will be provided.

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

3. **Time Filtering:** Both interfaces (main and map interface) should allow the user to view the articles on 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.
Awards and Extra Credits

Each team may enter its MapRSS client into a class contest for one or more of the following awards:

Best design: awarded to the solution with the best design at the source-code level.
Most usable: awarded to the solution with the best user interface design.
Best feature: awarded to the solution with the most imaginative new feature.

Performance Metrics

Around 65% of your performance comes from your team performance, and the remaining 35% comes from your individual performance in your team. The requirements implemented by your team decide the maximal points of your project score.

Team Performance: Three major sources for your team performance are: (1) the quality and design of the finish product; (2) project documentation, including those for requirements, design, planning; (3) how well you have consciously applied design principles and techniques; and (4) the team and quality assurance processes developed during the course of the project.

Individual Performance: Three major sources for your individual performance are: (1) your participation in team activities; (2) the peer-reviews from your team members; and (3) the percentage breakdown of your contribution to the project. The TA will look in your GitHub activity to evaluate your contribution throughout the semester. And during the demonstration session each of you will be asked to talk about what parts of the project you have worked on. Your demonstration score will be based on both the team and individual performance.

Peer Review: At the end of the semester each individual team member is required to fill in peer-review forms for his/her team members.
Project Grading:

The weights of the project deliverables are as follows:

1. System Requirements Specification Document ……… 15% (team grade)
2. Software Design Document
   a. Phase I …………………………………….…... 10% (team grade)
   b. Phase II ……………………………...…….…... 15% (team grade)
3. Design Patterns Specification…………………………...………...  5% (team grade)
4. Project Code……………………………………...……. 20% (team grade)
5. Project Demonstration…………………………………. 35% (team+individual grade)
6. Peer Review…………………………………..………… 5% (individual grade)

I.1. TOTAL…………………………………………………...100%

All grades will be given based on a 100 point grade scale. You also need to submit a “report on version control and issue tracking” at the end of the semester which will count towards your homework grade.

Project Management and Milestones

The project has several milestones with their respectively due date. At each milestone, you may be asked to demonstrate your progress by a set of software artifacts defined at that milestone. At the end of each milestone, we will decide objectives for the next milestone. The final product is due in the final week of lecture.

Milestone Objectives

You'll do your project in phases, with the following milestones:

Preliminary

Due date: February 1st.

Objectives:

- Team selection: Select your partners to form a team of three. If you don't have a team by the due day, I will assign you to a team.
  - Select a team liaison and report the name to the TA.
  - Pick a name for your team.
- Setup your eecs.wsu email. (How to setup your eecs email?)
- Login to the EECS GitHub server at https://github.eecs.wsu.edu using your EECS account credentials.
  - Use the repository under organization "Cpts323Spring2013".
  - Get familiar with the EECSGitHub server.
- Establish the development environment. Install necessary tools.
  - Decide your choice of programming language: Java or C#.
  - Get familiar with RSS feeds and RSS formats (RSS 1.0, RSS 2.0 and Atom 1.0)
- Download and install Google Earth Plug-in (GE plug-in in short) and read the Google Maps/Google Earth APIs Terms of Service. Look for example codes that embed Google Earth Plug-in in standalone C# or Java applications.
- A sample code (see the link below) for embedding Google Earth plug-in in a C# forms application is available with GNU Lesser General Public License. Download and compile the source and understand how the GE plug-in is integrated into the C# managed code. Explore the functionality and the control features that the control library provides.
  - winforms-geplugin-control-library: A library of controls to work with the Google Earth Plugin API in managed code. http://code.google.com/p/winforms-geplugin-control-library/

Milestone-1

Due date: February 18.

Deliverables: underlined items are deliverables.

Objectives:

- **Communication activities:**
  - Develop “requirement specification”
    - Provide use cases
    - Provide itemized requirements
    - Template for the “Requirements Specification Document” will be posted on ANGEL next week.
    - Deliver a good quality document: completeness, feasibility, etc

- **Planning activities:**
  - Make an initial version of project schedule.
    - Identify the main tasks of the project (The complete set of milestone steps for implementing your project will be available next week. You need to create a more detailed listing of tasks based on the given milestones.)
  - **Provide a Gantt chart to identify your work plan.**
    - In the Gantt chart you don’t need to partition responsibilities to team members yet. Later you will revise your Gantt chart and assign a team member to each task.

- **Construction activities:**
  - Start to develop the “Main interface”.
  - Add menu items and implement the following functionality:
    - Add RSS subscription (get RSS feed URL from user, add an item for it in the RSS feed list – you don’t need to fetch and display RSS feeds yet)
    - Rename existing subscription,
    - Add new channel, rename channel, delete channel,
    - Organize RSS feeds into groups (add/delete RSS feeds to/from channels)
  - Write the code to parse RSS feeds (at least finish parsing RSS 2.0 feeds. Your final software should support RSS 1.0, 2,0 and Atom 1.0).

Milestone 2

Due date: March 4.
Deliverables: underlined items are deliverables.

Objectives:

- **Communication activities:**
  - Develop “**Software Analysis Document**”
    - Template for the “Software Analysis Document” will be posted on ANGEL.
    - Deliver a good quality document: completeness, feasibility, etc
  - Planning activities:
    - Refine your **Gantt chart**, add new tasks if needed, assign a team member to each task.
  - Construction activities:
    - Fetch data from a (subscribed) RSS feed.
    - List the RSS items from an RSS feed. Display the following for each RSS item: item title, link to the original article, publication date of the original article, and item description (displaying the description is optional). You may limit the number of items you list from each feed. For example, list only the most recent 20 items from each feed. User may provide the number of items to be listed (optional).
    - Display the full article (when the user clicks on an article link). Requires rendering web pages.
    - The user sets the “update period” for the feeds (i.e. how often will the feed data be refreshed for the subscribed RSS feeds). You may also require a different “update period” for each individual feed (optional). Use a default update period if user doesn’t set one.
    - Refresh the RSS feed data for each subscribed RSS feed, according to the user defined “update period”.
    - **Upload your current code on GitHub before the deadline.**

Milestone 3

Due date: March 25.

Deliverables: underlined items are deliverables.

Objectives:

- **Communication activities:**
  - Develop “**Software Design Document**”
    - Template for the “Software Design Document” will be posted on ANGEL.
    - Deliver a good quality document: completeness, feasibility, etc
  - Planning activities: TBA
  - Construction activities:
    - Parse other two RSS formats (RSS 1.0 and Atom 1.0)
    - Keep track of the read and unread articles. Use a visual indication that shows read and unread articles. Users should be able to list the articles that have not yet been read yet.
    - Save the current configuration and user preferences to a config file. You may save the following: the collection of feeds, their (user given) names, feed update periods, organization into channels
    - Load a previously saved config file.
    - Build a local database to store all state names/abbreviations, and city names along with their coordinates in the United States. You need to create this database once.
Cover United States map with bounding boxes and query the Geonames webservice to obtain location names. Insert keywords and locations (latitude/longitude) into the database.

Upload your current code on GitHub before the deadline.

Sample code (in C#) for handling database operations (create database, insert/search to/from database) will be provided by instructor.

Milestone 4
Due date: April 8.

Deliverables: underlined items are deliverables.

Objectives:

- Communication activities: NA
- Planning activities: TBA
- Construction activities:
  - Search the local location database for the words that appear in the titles and descriptions of the RSS feed items. If one or more words from the RSS feed item exist in the location database, then retrieve the associated location(s). If none exists, then no location can be associated with that feed item.
  - Embed the Google Earth Plug-in in your application. You may use the winforms-geplugin-control-library, available at http://code.google.com/p/winforms-geplugin-control-library/. This is a library of controls to work with the Google Earth Plugin API in managed code. It is available with GNU Lesser General Public License.
  - Upload your current code on GitHub before the deadline.

Milestone 5
Due date: April 22.

Deliverables: underlined items are deliverables.

Objectives:

- Communication activities:
  - Develop “Design Patterns Specification” (Template is available on ANGEL)
  - Develop “Report on Version Control and Issue Tracking”
- Planning activities: TBA
- Construction activities:
  - Display the locatable RSS feed items on the map interface. Have an icon at the associated location for each of those feed items. The locatable RSS feed items include those that have associated location information (latitude/longitude).
  - Open a pop-up message on the map interface where you display the title and description of the article, along with a link to the article. The user should be able to view the article in the “main interface” by clicking that link.
- Implement time filtering (optional) Both interfaces (main and map interface) should allow the user to view the articles on 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.
- **Upload your current code on GitHub before the deadline.**

**Product Release and Presentations**

Due date: April 22-24.

Objectives:

- **Upload your final code on GitHub before April 24th.**
- **Communication activities:**
  - Present your project in class

**How to setup your ECS email account?**

Every student has an EECS email account that is automatically created for them upon enrollment into our program/classes.

Their EECS email address is the same as their EECS username (i.e. the user name the use they logon to systems in our lab). Typically, their user name is the first letter for the first name and up to the first seven letters of their last name. There are always exceptions to every rule; for example your user name would have likely been "say" -- individuals with two last name typically end up with a first, middle and last name from the registrar’s office. Passwords, are by default, your student ID number.

There are several ways to access your EECS email. The simplest way is with a web browser: mail.eecs.wsu.edu, use your full email address "user@eecs.wsu.edu." For IMAP/POP it is: mail.eecs.wsu.edu. For SMPT it is: mail.eecs.wsu.edu. (We like to keep things simple.) There are some more details within the KB: https://support.eecs.wsu.edu/KB/a64/configuring-thunderbird-for-new-zimbra-servers.aspx?KBSearchID=0.

Students can also stop by the helpdesk if they don't know their username, but every students have an EECS account.

**References**