New Design and Implementation for the Crawlers of CoMet System

Independent Study Report – Spring, 2015

Ang Chen

Supervisors: Dr.PeterBrusilovsky

Chirayu Wongchokprasitti

Contents

Abstract 3
Goals for the independent study 3
Introduction to Comet system and its crawler program 4
  Comet system 4
  Crawlers program 4
  Development environment techniques 5
Design of the crawlers program 5
  Original Structure 5
  New structure and design 6
  Crawler design 8
Implementation of each crawler 9
Future work 10
Acknowledgments 10
Appendix 11

Abstract

Comet is a Web-based social system for sharing information about research talks and seminars. The data source of the Comet is collected by a list of different crawlers. In this project, we rewrite the crawlers which no longer function well and create new crawlers according to the new requirement. We replace the old structure of the program with the new one under the idea of Object-Oriented Programming to avoid the single point failure. All the crawlers are functioning well under the new structure.

Goals for the independent study

Since the Google Calendar API stop providing services, all the crawlers in the Comet system depend on this API cannot run well anymore. So they need to be rewritten in order to function well without the Google Calendar API. Also due to the change of format of data, some other crawlers needed to be rewritten as well. Another problem for the crawler is that when any one of the crawlers failed to function, all of the others stop working. It is due to the bad design of the whole crawler program.

So the goal of the independent study can be divided into three parts. One is to rewrite the crawlers which are dependent on the Google Calendar API. Another one is to rewrite the crawlers which are no longer function well due to the change of the format of the data. The last one is to change the structure of the program to assure that once one crawler stops working the other functions can still work well. The ultimate goal is to make sure that the Comet system is robust, efficient and works well.

Introduction to Comet system and its crawler program

Comet system

The scree shot of Comet system

"Comet is a Web-based social system for sharing information about research talks and seminars organized at University of Pittsburgh, Carnegie Mellon University, and other research organizations in Pittsburgh."

Crawlers program

The Comet needs the input of the talk information. One major resource is the google calendar sources published by each of the academic unit in the University of Pittsburgh and Carnegie Mellon University like CMU Statistics Department, University Honors College and so on.

So the goal for the crawler program is to parse the information on those google calendar, process it and store it in the local database.

Also, since each academic unit has its own preference to publish the events, each calendar source need one corresponding crawler.

Development environment techniques

a. Java 6
b. Jsoup 1.8 (To provide the convenience of parsing the XML source)
c. Eclipse Luna
d. Java Multithread

Design of the crawlers program

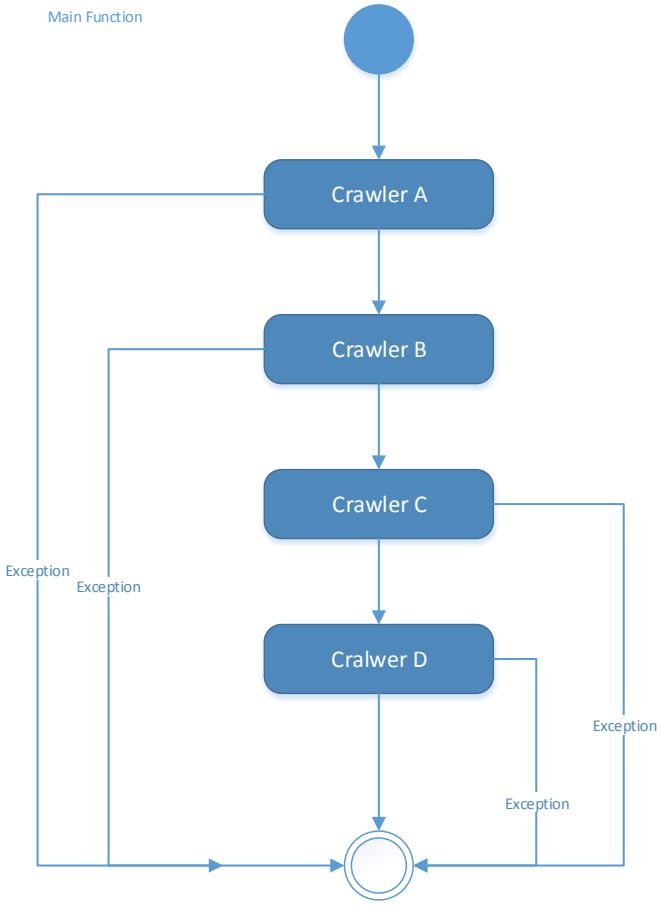
The program can be separated into two parts:

1. Each Crawler for each google calendar source.
2. The main function to put the output of each crawler to one large List. So it can be saved to the database which is already implemented.

Original Structure

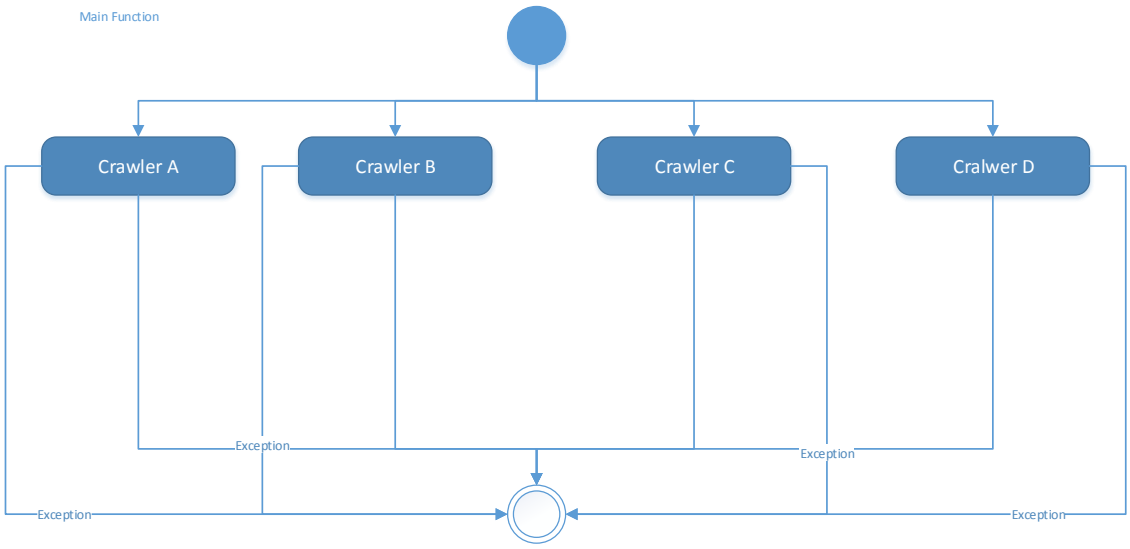
In the original structure, it is linear structure so if one crawler throws the exception, all the others stop as well which should be blamed to the bad design of the program.

We can see from the graph below:
If any one of the crawler throw an exception, it goes to the end without running other crawlers.

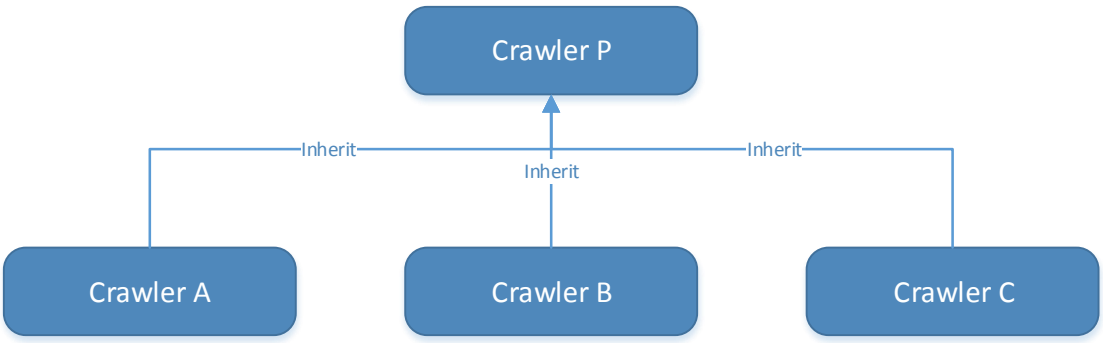


New structure and design

In order to prevent the situation could happens in the old structure, we decide to implement the crawler program with a new parallel structure. So each of the crawlers only take charge of its own duty without affecting others which is shown below.



Also, under the concept of OOP (Object-Oriented Programming) and code reuse, I decide to use the inheritance structure. So crawlers with similar need will inherited from one parent level crawler. (As shown below)



In order to provide convenience to coding and also act according to the OOP, I write the POJO (Plain Old Java Object) class called RawTalk to hold the data. It is much better than use separate parameters to hold the data.

```
public class RawTalk {


    private String series="";
    private String speaker="";
    private String affiliate="";
    private String title="";
    private String description="";
    private String startTime="";
    private String endTime="";
    private String locationAdd="";
    private String talkUrl="";
    private String originSponsor="";
    private String source="";

    private ArrayList<String> sponsor = new ArrayList<String>();
```

Crawler design

For each crawler, there are three main parts.

1. Parse the data source and use the RawTalk to hold it.
2. Process it for better extraction
3. Call the function in Calendar to save it.

a. Implementation of crawler program

In the old structure, each crawlers is a function written in one class called Calendar which makes the java file about 6000 lines long. It also violates the rules of separation of concerns.

In order to prevent the problems mentioned above, I decide to write each crawlers in a single class. It not only substantially decrease the number of lines in Calendar which makes it easy to manage for future improvement, also it enables the inheritance structure implementation mentioned above.

Another issue is to make it parallel. For this issues, I decide to make each crawler as a Thread. Every thread can run at the same time with the implementation of Thread. It echoes the parallel design mentioned above.

But when doing the coding, we find another problem. In the old structure, all the crawlers will put the results in to one large List and then the program will put the all the information hold in this List to the local database. It is a bad design and it causes a problems here. Since we use the Multi-Thread structure, the function in Calendar do not know whether the crawlers is finished or not. It will not wait for crawling so it saves nothing to the database as a result .But in order not to change so much of the original program, we decide to not to modify this part (since it still works well) but to focus on how to change the crawlers to get the work done.

Then we decide to implement the Callable interface.

"The Callable interface is similar to Runnable, in that both are designed for classes whose instances are potentially executed by another thread. A Runnable, however, does not return a result and cannot throw a checked exception."

So once we bring in the Callable interface, the save to database function will not run until all the crawlers finish their own jobs.

Implementation of each crawler

For each crawler, the input data format is XML. With the use of Jsoup, we can directly extract the elements we want form the data source. After the crawler extracts the data, it puts the data into a public List. Upon all the crawlers finish their work, the program will call another function to save the public List to the database. Then all the work is done for this time

Future work

All the sources are similar in general but they are still different for the program to parse. Make a general crawler and use templates to satisfy the difference between sources will be a rewarding work. Because it would enable non-programmers create a new crawler as new source coming in.

Also, when look into the codes, there are many violation to OOP and general program conventions. To elegantly modify the code will make the system more robust.

Acknowledgments

I would like to thank my advisor, Dr. Peter Brusilovsky and Chirayu Wongchokprasitti for offering me this great opportunity and give the valuable advice and support during the task.

Appendix

List of crawlers being created or fixed under this project

Name	Source
Astrophysics Seminars	Google Calendar
Biophysics Seminars	Google Calendar
CMU Stat Seminars	Google Calendar
CMU CNBC	Google Calendar
Condensed Matter Physics Seminars	Google Calendar
CPCB Seminar	Google Calendar
Particle Physics Seminars	Google Calendar
PIER Seminars	Google Calendar
Quantum Seminars	Google Calendar
CMU Statistics in Education Research Group	Google Calendar
CMU Psychology	Google Calendar
University Center for International Studies	Official Website
Theory Seminars	Google Calendar
University Honors College Events	Google Calendar
University Lecture Series	Google Calendar