
Web Lab Collaboration Server and Web Lab Website

Wioletta Holownia (wh226)

Michal Kuklis (mpk25)

Natasha Qureshi (nq24)

Table of Contents

1. INTRODUCTION	3
2. DATABASE	3
3. WEB DATA COLLABORATION SERVER	3
3.1 GUI INTERDEPENDENCIES	3
3.2 GUI DEVELOPMENT	6
3.3 WEB LAB SERVICES	8
3.4 WEB LAB SERVICE INTERFACES	9
3.4.1 SEARCH SERVICE	10
3.4.2 RESOURCE SERVICE	11
4. WEB LAB WEBSITE	12
4.1 INTERFACE	12
4.2 TOOLS	12
4.2.1 GETPAGES	12
4.2.2 PROGRESS MONITOR	13
4.2.3 COLLECTION MANAGER	14
4.3 DATABASE STATISTICS	16
5. COLLABORATION SERVER & WEB LAB SERVER - COMMUNICATION	17
6. FUTURE WORK	17
7. ACKNOWLEDGEMENTS	18
8. REFERENCES	18
9. APPENDIX	19
APPENDIX A: SETUP WEBCOLLAB PROJECT ON LOCAL MACHINE.	20
APPENDIX B: SETUP LOCAL WINDOWS WEB DEVELOPMENT ENVIRONMENT FOR THE WEBLAB WEBSITE PROJECT.	23

1. Introduction

This document outlines the development that has been made to the Web Lab website and the Web Data Collaboration server in the spring 2008 semester. There were two main goals for this semester; one was to develop the Web Lab website into a fully functional tool for researchers to allow them to search for pages, retrieve the results and to save them for future purposes. The second goal was to extend the Web Data Collaboration Server to improve the interaction between the GUI interfaces. The Web Data Collaboration Server can be used by different clients written in languages such as Java, Javascript, PHP and more.

2. Database

The data in the scidata1.tc.cornell.edu database is sourced from the Internet Archive. The database contains several smaller collections such as Amazon, Cornell, Columbia and others that use the same database schema. Detailed information about the database schema and specifications can be found at <http://weblab1.tc.cornell.edu/documentation/schemadocs/>

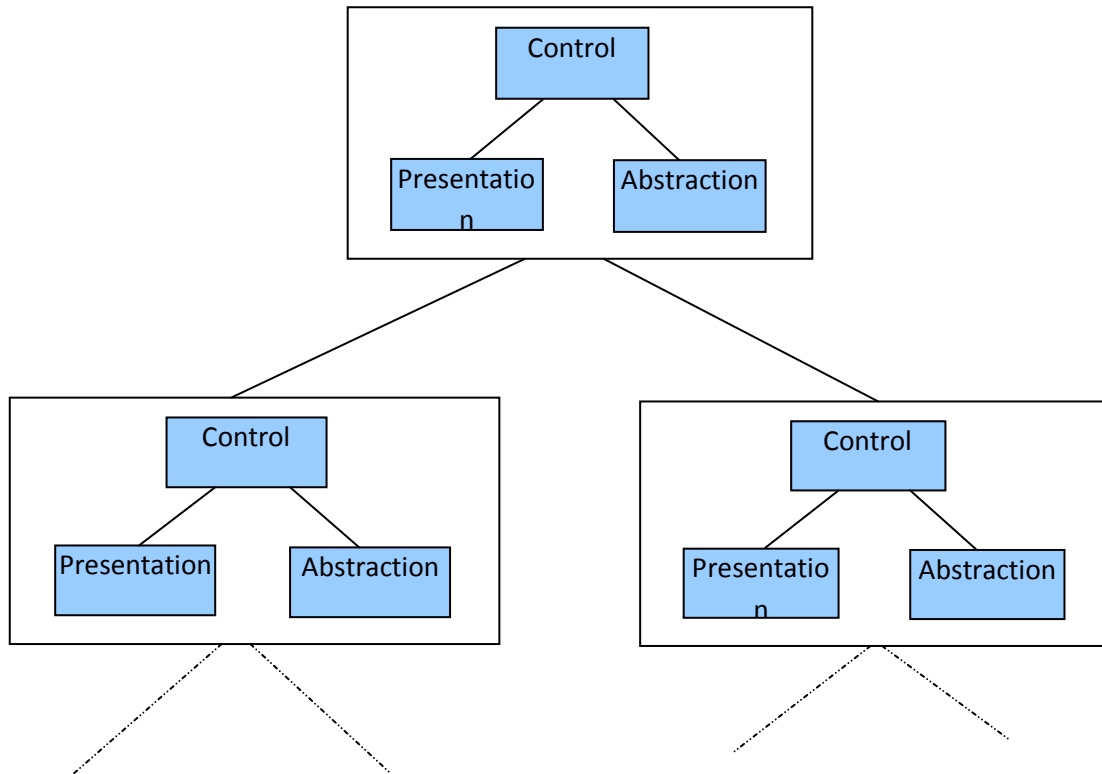
3. Web Data Collaboration Server

3.1 GUI Interdependencies

Every complex GUI poses similar questions. How should the interdependent components interact with each other? What is the best way to handle the interdependencies? During the development process of the WebData Collaboration Server (WebCollab[1]) we were challenged with the same questions. The WebCollab is a Rich Internet Application (RIA[2]) built on top of the Google Web Framework (GWT[3]). The WebCollab's GUI is based on many interacting UI components.

We explored different design patterns, frameworks and techniques to answer these questions and to find the solution that would work for the WebCollab application.

The design patterns we experimented with were Model-View-Controller pattern (MVC[4]) and Hierarchical MVC (HMVC[5]). The idea of the HMVC is based on the hierarchical structure of agents. Each agent consists of the model, view and control layer. The agents can communicate with each other through the control layer. The structure is based on the tree with the top root agent as the main application entry.



The structure of an application with HMVC

Although the structure looked very promising and it allowed for a nice separation between components, it would have introduced a plethora of potential changes to the Webcollab's structure. Some of the components are deeply nested inside other components. In order to send a message from a deeply nested element to an element on a completely different branch, other agents would need to be involved in the event propagation which complicates the overall communication between components.

Another paradigm that was explored was the aspect oriented programming (AOP[6]). The idea of the AOP is to allow for the separation of concerns. Separation of concerns breaks down the application into distinct modules to avoid functional overlapping. Object oriented programming (OOP) supports some level of separation by introducing packages and classes which allow for better code encapsulation. AOP goes

one step further and provides modularization across multiple concerns. In other words we are able to create modules across multiple classes. The modularization unit in AOP is called *aspect* and is equivalent to the class unit in OOP. The aspect is usually triggered by some event.

The current design of the Webcollab allowed us to use AOP to solve the GUI problems without major changes to the current Webcollab structure. We decided to use *Rocket GWT* library which supports AOP in the browser environment. The advice in Rocket GWT is defined by the target element and its methods, and the advisor - the interceptor which is triggered when one of the target's methods is executed. In order to solve the interdependencies between components we applied aspects on the event listeners. The aspect definitions were written in XML (ApplicationBeanFactory.xml). The example of the aspect definition is shown below:

```
<aspect
  advisor="pageMenuInterceptor"
  target="showPageListener"
  methods="onCellClicked" />
```

executes when row clicked on the PageTable component

The Interceptor (pageMenuInterceptor in this case) is a class which implements *rocket.beans.client.aop.MethodInterceptor* interface. The example of this class is presented below:

```
/**
 * <p>
 * Shows {@link PageMenu} items when at least one page is currently open
 * and the wrapper is selected on the {@link WrapperGUI}
 * </p>
 *
 * @author mpk35
 * @version $Id: PageMenuInterceptor.java,v 1.2 2008/05/01 04:22:46 mpk35 Exp $
 * @since Wrapper nbsp;
 * @see
 **/
```

```
public class PageMenuInterceptor implements MethodInterceptor {
```

```
/** *
 * Overrides @see MethodInterceptor#invoke(MethodInvocation).
 * */
```

```
public Object invoke(MethodInvocation invocation) throws Throwable{
```

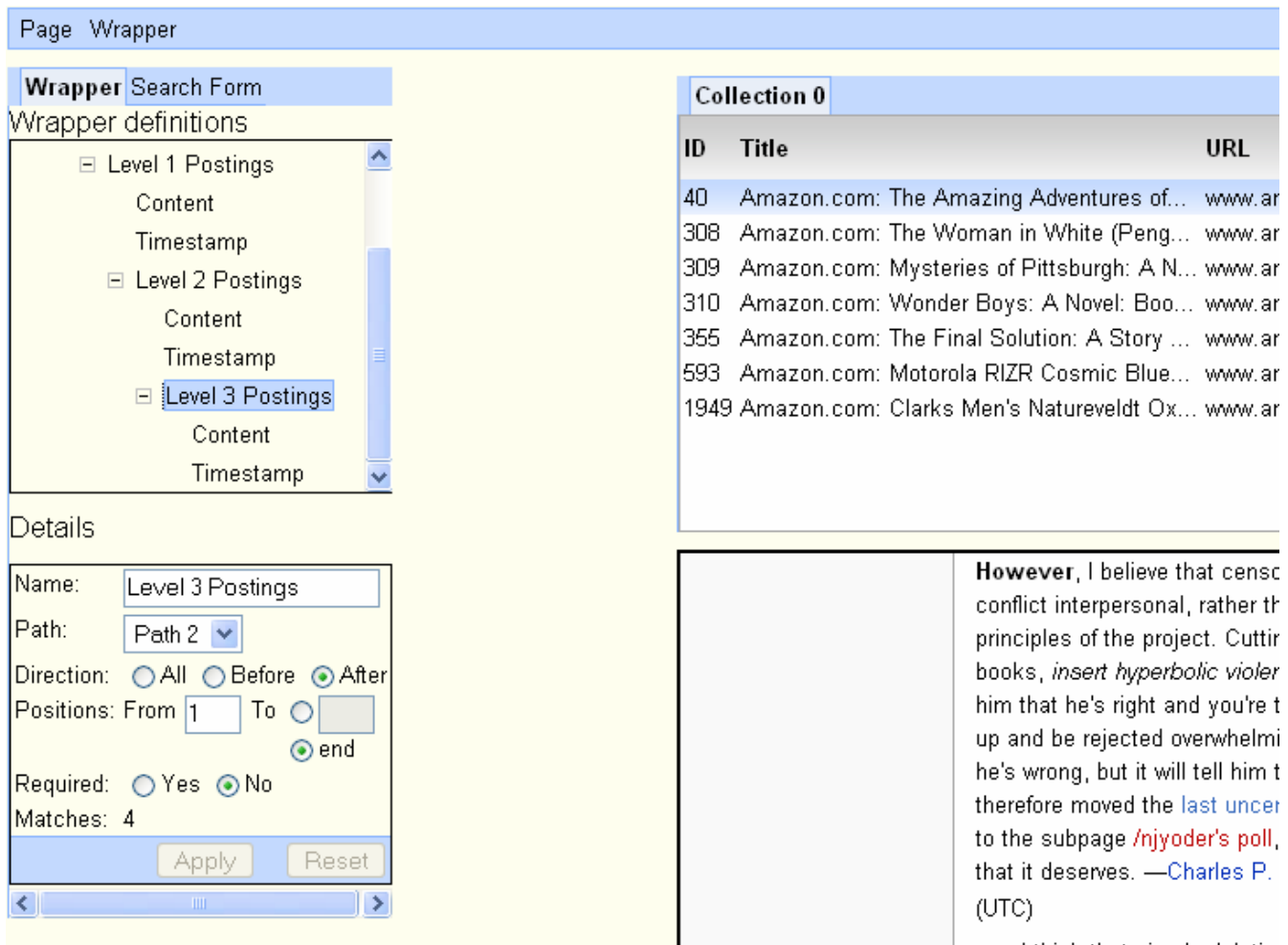
```
...
```

```
}
}
```

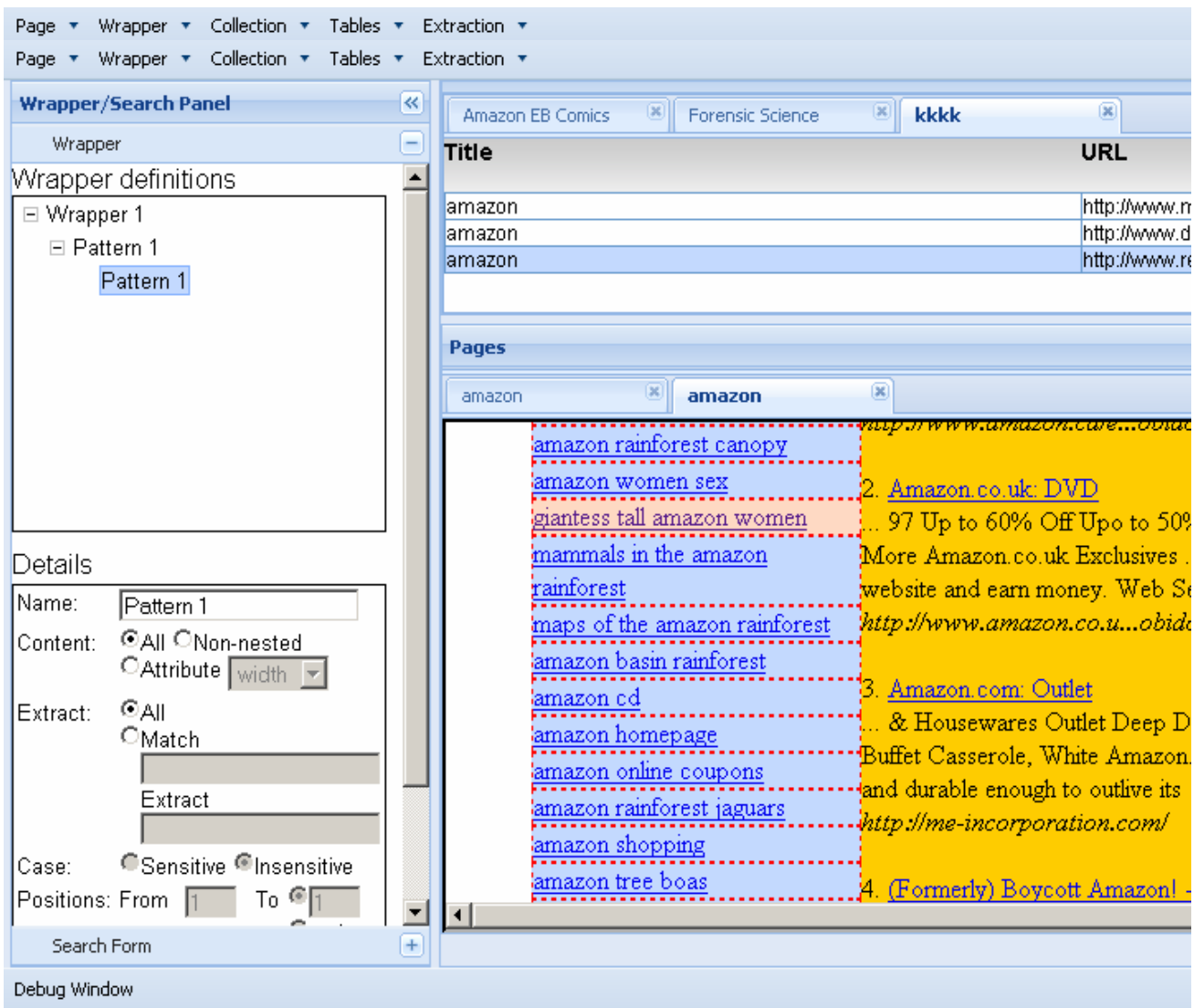
We believe that Aspect Oriented Programming helped us overcome aforementioned problems. Most of the interdependencies were successfully solved. We also discovered that some interceptors could be applied with multiple aspects which allows for more code reuse.

3.2 GUI Development

A portion of the semester was dedicated to the new GUI development. We decided to make use of GWT-Ext[8] library. GWT-Ext is a collection of widgets that can be used with the GWT framework. The GWT-Ext widgets are richer and provide a better user experience over the standard GWT components. Below we present the difference between the older and current GUI.



Previous GUI based on GWT widget

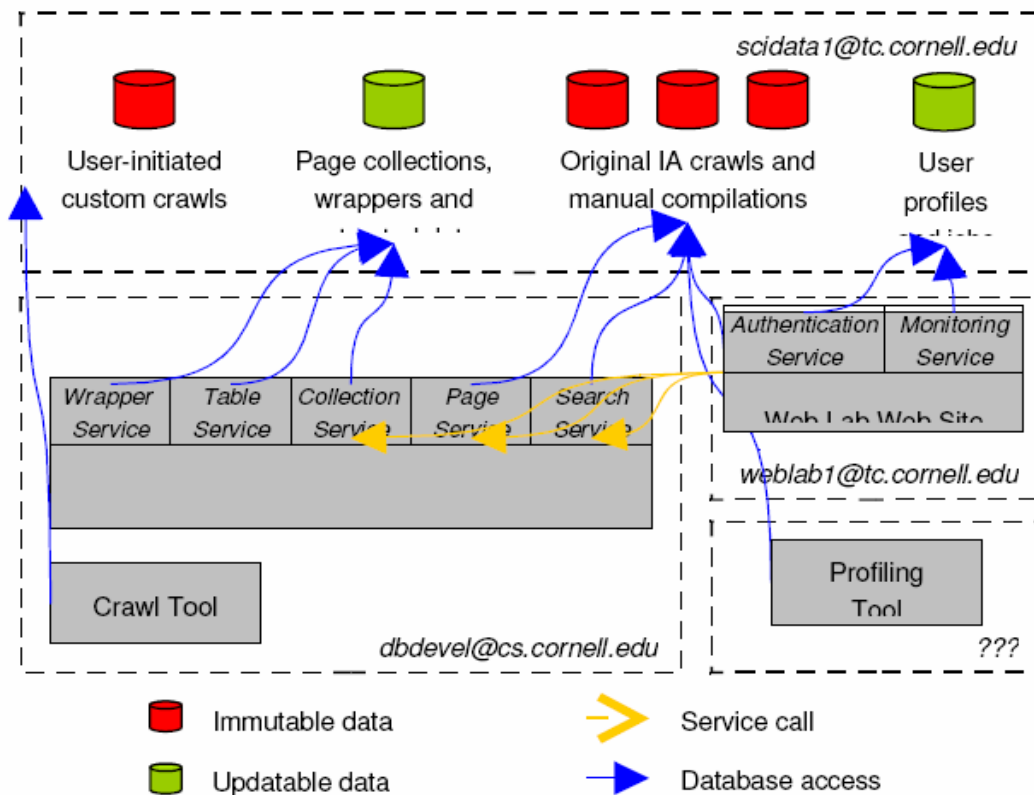


Current GUI based on GWT-Ext widgets

3.3 Web Lab Services

Currently many disjoint applications need to access the web lab database at scidata1 and there is no single interface that allows them to do so. These applications consist of the website, extraction tool, crawl tool and the profiling tool amongst others. Due to a lack of a single uniform interface, the various applications have duplicated services and can also access the database directly, which is not recommended from a security standpoint.

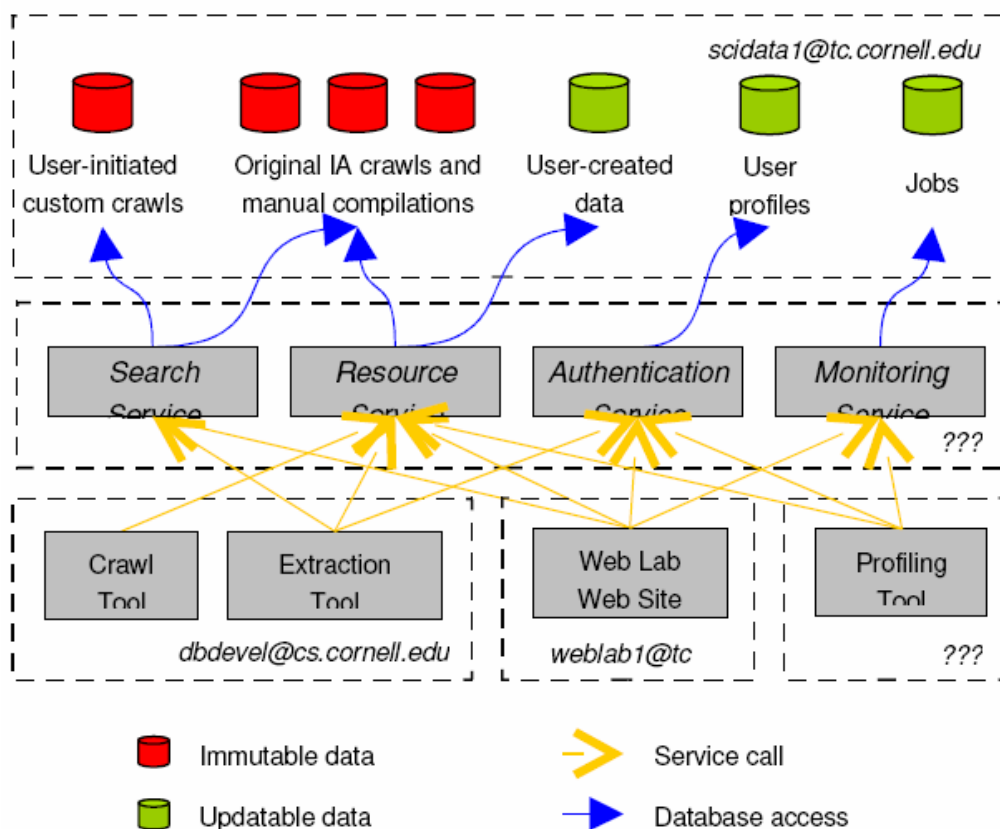
The current system resembles the following structure:



The web lab website is currently exposed to the Search Service and Collection Service that it calls for the GetPages and the Collection Manager Tool.

The new proposed architecture will create a uniform web service interface that will interact with the various applications. These services will include the Search, Extraction, Resource, Authentication and Monitoring service. Applications will not need to access

the database directly and in the future these services can be extended for further functionality. The following diagram details the architecture of the new Web Lab system.



3.4 Web Lab Service Interfaces

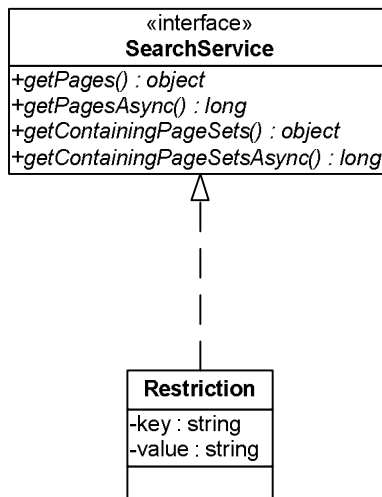
We have implemented the interface for the Search Service and the Resource Service. The primary objects that need to be manipulated are crawls, collections (a set of crawled web pages), Page sets (a sub-collection), extracted data (user retrieved data), wrappers (used for data extraction from web pages), user profiles and jobs performed by each application. These objects are all types of Resources since they share some common attributes and behavior.

Most of the Web Lab service operations can be executed in both synchronous and asynchronous mode. An operation executed in the synchronous mode blocks the application until the execution is complete and returns the actual result of the operation. The asynchronous mode however, immediately returns the ID of the job while the execution of the process continues.

3.4.1 Search service

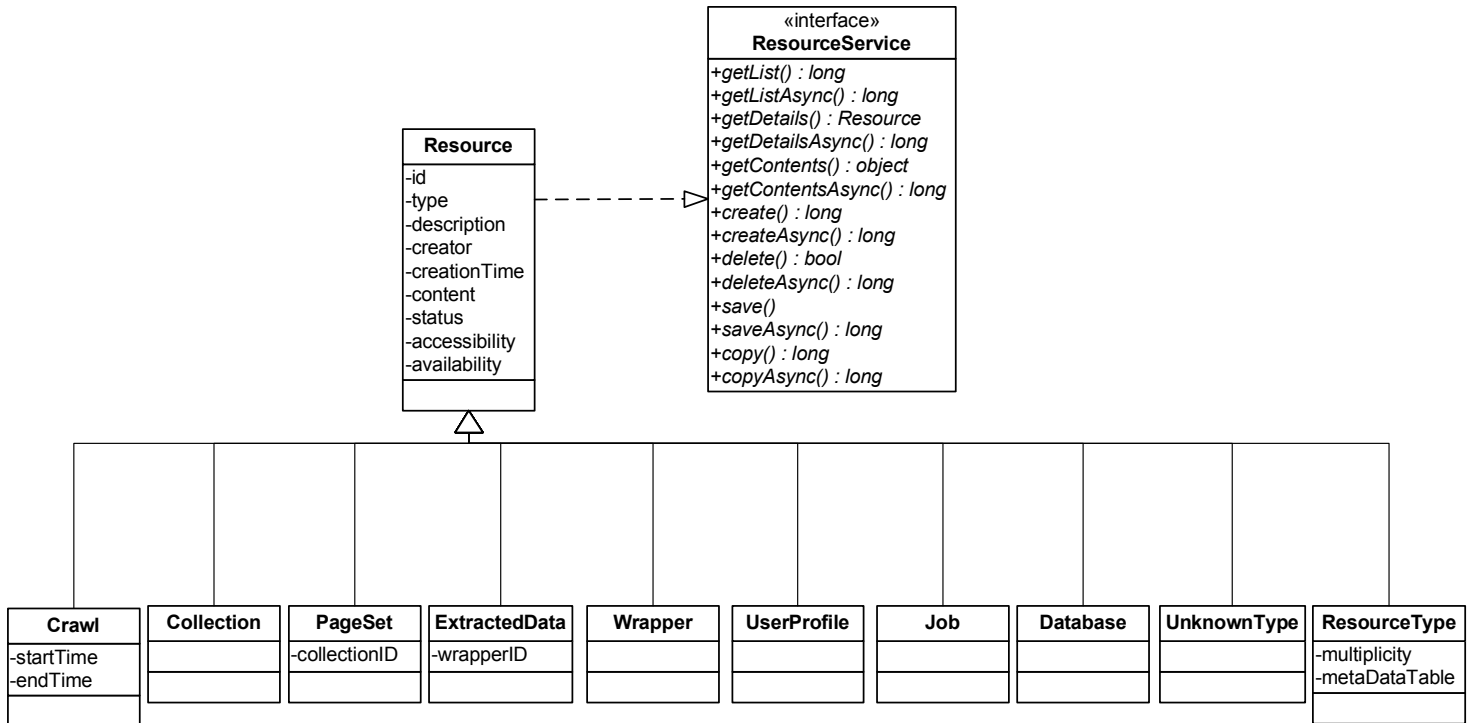
The search service is used to retrieve pages from the Page Table that satisfy the given metadata restrictions. Using this service it will also be possible to retrieve pages according to their actual content (and not metadata) and also to retrieve the metadata of those page sets that contain a particular page.

The class for the SearchService has the following structure:



3.4.2 Resource service

The Resource service will be used to manage all the different types of resources (objects). Each resource will have metadata along with some particular variables for the particular type of resource. The Resource service will provide a consistent platform for all the Web Lab applications to create, copy, delete and manipulate resources.



4. Web Lab website

The Web Lab website is built on PHP and follows the model view controller architecture [9]. The website also interacts with the Web Data Collaboration Server to call the Collection and Search Services that have been exposed. The following additional work was completed on the website this semester to make it fully functional and more user-friendly and accessible for researchers.

4.1 Interface

The interface for the GetPages tool was structured so that the query fields were put into mental categories that would be more understandable for users. In addition to this the documentation or the help section for the GetPages tool was revised to reflect the change in the interface. It also provided example queries to help the user to get accustomed to the search criteria.

4.2 Tools

Web Lab tools are the backbone of the website. It consists of the GetPages tool, the Progress Monitor and the Collection Manager. These three tools allow the user to query for pages in the different collections available to them, browse through previous collections, and also allows them to save, rename, preview and download the content of the collections.

4.2.1 GetPages

GetPages allows users to specify search criteria, which can as vast as an entire web library or as restrictive to a single crawled page, to retrieve a subset of the matched pages from the Web Lab database. Returned results can be viewed on-line or directly from the database. This semester, a major improvement was made to implement asynchronous searches using GetPages. Earlier, the search call to the database was synchronous and due to this the search would take a long time if the retrieved pages were very large in number. To make the search process as convenient for the users, an asynchronous search call is now made and the search results are now displayed using the Progress Monitor (section 4.2.2).

The tool is available at <http://www.weblab.infosci.cornell.edu/tools/getpages> .

The documentation for this tool is available at <http://www.weblab.infosci.cornell.edu/documentation/getpagesdocs> .

4.2.2 Progress Monitor

Progress Monitor is used to monitor and display collections that have been created with the GetPages tool for authenticated users. As explained in section 4.2.1 the Progress Monitor also displays the status for those searches that are currently in progress.

Progress Monitor contains a list of following attributes:

- Collection Title the name of temporary collection.
- Collection Description the description of temporary collection.
- Collection Details the link to detail view about temporary collection, only available for unsaved and saved collections.
- Temp CollectionID the system-wide ID of temporary collection e.g. dbo.c23.
- Metadata DB the name of the database/library containing the pages in its collection.
- #no of Returned Pages the number of returned page Ids from the collection content table.
- Creation Time the time that temporary collection was created.
- Modification Time the time that temporary collection was last modified.
- Search Results Progress Status:
 - INCOMPLETE status indicates that collection tuple has been created and associated with it the collection content table is in progress.
 - UNSAVED status indicates that collection tuple has been created and associated with it the collection content table is complete.
 - SAVED status indicates that collection tuple has been saved as or renamed.
 - FAILED status indicates that collection tuple has been created but creation of the collection content table associated with it has failed.

Progress Monitor Tool gives a quick feedback to the user about the queries that have been submitted via GetPages tool or preview previous created temporary collections. The Progress Monitor allows users to access the Collection Manager (section 4.2.3) for each of their collections.

The tool is available at <http://www.weblab.infosci.cornell.edu/auth/progressmonitor>.

4.2.3 Collection Manager

The Collection Manager is a tool that previews, modifies and manages single collections. The collection manager interacts with the Web Data Collaboration Server to access the Collection services using the JSON protocol and uses these to delete and modify collections.

On the level of the website the Collection Details view provides a single integrated webpage, which the user can rename, save, preview and download collections.

4.2.3.1 Create Collection

When a GetPages request is submitted the tool sends an asynchronous request using JSONRPC protocol to communicate with a remote method `getSearchResult(String username, String library, Collection collection)` from the Java SearchService to perform creation of new temporary collection.

Result:

- Search service creates a new tuple in the `dbo.collections` table and corresponding to it collection content table holding subset of PageIDs from specified library.
- Each subset is stored in the WebExtraction database as a temporary collection.
- New subset sets collection status for UNSAVED as default unless creation of new collection has failed and then the status is set to FAILED.
- Each user has its own set of temporary collections stored in the `dbo.collections` table specified by the pair of keys {id and owner}.

4.2.3.2 Delete Collection

From the Collection details view a user/owner is allowed to delete unwanted temporary collections that they have created.

A call to 'Remove Collection' sends an asynchronous request using JSONRPC protocol to communicate with a remote method `removeCollection(String username, int collectionID)` to the Java CollectionService.

Result:

- `CollectionService.RemoveCollection` method deletes specified by collectionID key collection tuple and corresponding to it collection content table.

4.2.3.3 Rename / Save Collection

From the Collection details view a user/owner is allow to rename / save temporary collection by specifying collection name and description.

'Add/Edit Collection Title and Description' action sends an asynchronous request using JSONRPC protocol to communicate with a remote method `renameCollection(int collectionID, String username, String name, String description)` from the Java `CollectionService`.

Result:

- `CollectionService.RenameCollection` method renames specified by `collectionID` key collection tuple by adding name, description and last modified time.
- When temporary collection is renamed the status of the collection tuple is changed to `SAVED`.

4.2.3.4 Preview Collection

From the Progress Monitor Tool a user is allowed to preview metadata about created collections.

Preview collection contains a list of following attributes:

- Full Path: Protocol + Host + Path
- Port: this is a port number for example: 80
- Archive Time: this is the time at which the webcrawler archived the page.
- IP Address: this is the IP Address of the server from which the page was retrieved.
- MimeType: indicates the Internet media type of the message content for each page, consisting of a *type* and *subtype*, for example: *text/plain*.
- Crawl Name: this is a unique `CrawlName` corresponding to each crawl in the database.
- Doc. Title: this is the title of the webpage.
- Language this is the language of the webpage.
- Path: this is the path of the webpage following the hostname.
- Extension: URL Suffix which defines the type of page.

In some cases, created collections are very large in size and therefore the preview collection is paginated so that only a certain number of pages are displayed on each page. The entire collection can be viewed by browsing through the pages.

4.2.3.5 Download Collection

The download collection tool complements the preview collection as it allows the results for entire collection to be downloaded into a comma separated file. In addition to the attributes that the preview tool displays, the downloaded file contains all the metadata attributes for each page. These include the Page IDs, URL IDs and Arc data (from the crawl results).

4.3 Database Statistics

The database statistics tool allows researcher to view the different database statistics. These statistics are retrieved from the WebLibraryMetadata database and are not static contents on the webpage. The following are the different database statistics that are visible to the user:

1. Crawl Statistics: these include crawl names, ids, time of crawl and size of crawl. Available at http://www.weblab.infosci.cornell.edu/tools/crawl_sizes_statistics
2. Database Size: Detailed information on the size of each crawl database. Available at http://www.weblab.infosci.cornell.edu/tools/database_sizes_statistics
3. Web Lab Collection Profile: Basic statistics on the different web lab collections. Available at http://www.weblab.infosci.cornell.edu/tools/page_frequency_distributions

5. Collaboration Server & Web Lab Server - Communication

5.1 Asynchronous Methods

The Web Lab website submits the search request for GetPages asynchronously and also uses the Search Service at the collaboration server. This has been achieved by forking a process at the PHP level to create a child process which submits the search requests and then wait for the results, while the parent process allows users to browse and return to the site. The forked process then calls the SearchServerAsync Service to access the getSearchResult().

5.2 JSON-RPC protocol

JSON-RPC (jabsorb [10] library) protocol was used to accomplish the communication between Search Server and clients. Unlike the XML-RPC protocol or the SOAP protocol it allows for bidirectional communication between the service and the client, treating each more like peers and allowing peers to call one another or send notifications to one another. It also allows multiple calls to be sent to a peer that may be answered out of order.

Additional advantages of using JSON-RPC protocol include the following:

- simple and elegant schema
- works with many programming languages like: Javascript, Python, PHP and others
- supports complex data types
- supports remote method invocations (RPC)

6. Future Work

Forthcoming work should be focused on discovering new interdependencies between components and applying AOP paradigm to solve them. New GUI development was focused on replacing the main WebCollab widgets. We believe that there is still plenty of room for GUI improvements and GWT-Ext is a great library to achieve it.

On the website end, we envision to build a GetLinks tool. This tool can be used to retrieve back-links or front-links, depending on the search criteria, for a single webpage or a collection of webpages. It will be beneficial for researchers as it will allow them to perform various other analyses, such as determining the popularity of the webpages using the back-link count over time. This tool can use the progress monitor and collection manager tool that are already in place to display and download results.

7. Acknowledgments

This work is part of the Web Laboratory, which is a joint project of Cornell University and the Internet Archive. The Cornell Web Lab is funded in part by National Science Foundation grants CNS-0403340 SES-0537606, IIS-0634677, and IIS-0705774. We would like to acknowledge the guidance and constant support by our advisers: Professor William Arms, Dr. Felix Weigel and Manuel Calimlim.

8. References

- [1] Web Data Collaboration Server developed by the Cornell Database Group in the context of the WebLab project: <http://www.infosci.cornell.edu/SIN/WebLab/>
- [2] Rich Internet Application: http://en.wikipedia.org/wiki/Rich_Internet_application
- [3] Google Web Toolkit - Google Code: <http://code.google.com/webtoolkit/>
- [4] Model-view-controller: <http://en.wikipedia.org/wiki/Model-view-controller>
- [5] Hierarchical MVC: <http://en.wikipedia.org/wiki/Presentation-abstraction-control>
- [6] Aspect Oriented Programming:
http://en.wikipedia.org/wiki/Aspect-oriented_programming
- [7] Rocket GWT: <http://code.google.com/p/rocket-gwt/>
- [8] GWT-Ext: <http://code.google.com/p/gwt-ext/>
- [9] Code Igniter – Model View Controller for PHP <http://codeigniter.com/>
- [10] <http://jabsorb.googlecode.com>

Appendix

Appendix A: Setup webcollab project on local machine.

Pre-requirements

1. Eclipse SDK 3.3.x <http://www.eclipse.org/>
2. Tomcat Server 6.0.16: <http://tomcat.apache.org/>
3. Eclipse plugin for Tomcat: <http://www.eclipsetotale.com/tomcatPlugin.html>
4. Optional (if accessing scidata1 MSSQL database server outside Cornell) CS VPN:

[http://www.cs.cornell.edu/support/netconn/index.php?filename=Remote-Access/Setting up PPTP for Winxp and 2k.htm&toggle=Remote-Access](http://www.cs.cornell.edu/support/netconn/index.php?filename=Remote-Access/Setting_up_PPTP_for_Winxp_and_2k.htm&toggle=Remote-Access) using CUCS Radius Service

5. Gforge account <https://gforge.cis.cornell.edu/> with granted permission to the CVS repository for:
 - /cvsroot/javautils
 - /cvsroot/webarchive

Installation Steps

- 1) Checkout the webcollab project from CVS in eclipse:
 - a) The repository for the core functionality for the webcollab project:
`:extssh:developname@gforge.cis.cornell.edu:/cvsroot/webarchive checkout`
*NOTE:*The webarchive project contains functionality for the server and the client therefore it should be setup as 2 separate projects in eclipse environment: Wrapper.Client and Wrapper.Server.
 - b) The repository for the additional java utilities for the webcollab project:
`:extssh:developname@gforge.cis.cornell.edu:/cvsroot/javautils checkout`
NOTE: The GWT (version 1.4) supports java 1.4 on the client side therefore the javautils projects contains 2 folders: 1_4 and 1_5. The 1_4 contains functionality used by the client side and 1_5 contains functionality for the server side. The javautils project should be setup as 2 separate projects in eclipse. The example project structure is presented below:
 - JavaUtils.1_4/
 - 1_4
 - JavaUtils.1_5/
 - 1_5

2) Project Settings.

- a) There are numerous files associated with the project which are used by eclipse environment and GWT framework. All settings can be found in CVS
:extssh:developername@gforge.cis.cornell.edu:/cvsroot/webarchive/webarchive/webcollabSetup checkout
- b) The folder contains a zipped file with the project settings for windows environment. There are 4 folders: JavaUtils.1_4, JavaUtils.1_5, Wrapper.Client, Wrapper.Server. Each folder contains files specific for its project and needs to be copied over to project location.
- c) The properties file used by ANT Wrapper.Server./build.yourname-laptop.dbdevel.properties – this file is used by building process and it should be adjusted to the local settings.
- d) The additional files responsible for code formatting, code templates and styles which need to be imported into eclipse environment.

Compiling and running Wrapper.Client

In order to compile and run Wrapper.Client:

Run As -> Open Run Dialog -> WrapperGenerator-windows:

- Arguments: -style DETAILED
-out www.edu.cornell.cs.webarchive.wrapper.WrapperGenerator

Options Main:

- com.google.gwt.dev.GWTShell – in order to run the client from GWT shell
- com.google.gwt.dev.GWTCompiler – in order to compile client

Compiling and Running Wrapper.Server

1. The compiling, building and deploying process is done in ANT. The build file can be found in Wrapper.Server/build.xml. This file uses external properties described in the project settings section. The properties file can be set as follows:

Run -> External Tools -> Open External Tool Dialog -> Properties:

- uncheck “use global properties”
- add files: select your .properties file -> /Wrapper.Server/build.yourname-laptop.dbdevel.properties

The ANT targets used to compile and deploy project are presented below:

Ant -> remove (undeploy old version of the project)

Ant -> clean

Ant -> compile

Ant -> install

Appendix B: Setup Local Windows Web Development Environment for the Weblab Website Project.

1. Install php5 (or most current version)
2. Configure remote connection to scidata1 MSSQL database server
3. Configure php5 and driver to the mssql database
4. Install Apache
5. Configure Apache with CodeIgniter php framework.
6. Checkout weblab project from SVN

1. Install php5 (or most current version)

Download the newest version of the php from <http://www.php.net/downloads.php> and install on your local machine, for more information please see <http://www.php.net/manual/en/install.windows.php>

Changes in php.ini file:

- Windows Extensions: turn on extension=php_mssql.dll
- Directory in which the loadable extensions (modules) reside. extension_dir = "C:\php\ext" make sure you have mssql dll in the ext/php_mssql.dll extension folder or extension path is set correctly.

2. Configure remote connection to scidata1 MSSQL database server outside of Cornell

If accessing scidata1 MSSQL database server outside Cornell you need to set up CS or any Cornell VPN

Suggested to use:

http://www.cs.cornell.edu/support/netconn/index.php?filename=Remote-Access/Setting_up_PPTP_for_Winxp_and_2k.htm&toggle=Remote-Access using CUCS Radius Service.

3. Configure php5 and driver to the mssql database

SQL Server 2005 Driver for PHP

<http://www.microsoft.com/sql/technologies/php/default.mspx>.

Please read more about it at: <http://www.php.net/mssql>.

4. Install Apache

Download the latest version of Apache Webserver from:

<http://httpd.apache.org/download.cgi>

Instruction how to install, run and test Apache can be found at

<http://httpd.apache.org/docs/1.3/windows.html>

5. Configure Apache with CodeIgniter php framework.

Configuration Apache with CodeIgniter php framework can be found at

http://codeigniter.com/wiki/mod_rewrite/

Note: http.conf must have turn on mod rewrite module.

CodeIgniter Tutorial can be found at: http://codeigniter.com/user_guide/

6. Checkout weblab project from SVN

Source code - SVN on forge.cornell.edu => svn checkout --username

<https://forge.cornell.edu/svn/repos/weblab>

Note: you must have an account on th [forge.cornell](http://forge.cornell.edu) first and be added as a member to the WebLab Project located at https://forge.cornell.edu/sf/projects/clo_weblab_project.