

WebLab site development and Researchers' Tools

Table of Contents:

Table of Contents	1
Abstract.....	2
Web Library Project Setup.....	2
Database.....	2
Researchers' Tools.....	2
Get Pages tool.....	3
Get Crawls tool.....	6
Service on the Web Data Collaboration Server.....	7
Clients.....	8
JSON-RPC protocol.....	10
JSON schema.....	10
Future Work.....	11
Acknowledgement.....	11
References.....	12

Abstract

The purpose of this document is to report the progress of the Web Lab site development and Researchers' Tools that were designed during the 2007 fall semester. One of the goals for that semester was to move the Web Lab website to the open source environment. Another purpose was to develop a search service as the part of the Web Data Collaboration Server [1] which could be used among different clients written in languages such as Java, Javascript, PHP and more.

Web Library Project Setup

The majority of the first half of the semester was dedicated to the website design and implementation. We were moving away from the C# ASP .Net environment to the open source software. The Web Lab website is written in PHP. The CodeIgniter [2] web application framework was used to develop the website structure. This framework is based on the model-view-controller pattern [3] and it introduces better code organization, offers quality documentation and can be easily adopted by future students.

The current version of the website (<http://weblab1.tc.cornell.edu>) is running on the Apache 2 (2.0.55) web server on the Linux (Red Hat) machine. In order to connect to the remote Microsoft SQL Server from the linux/php environment, the TDS protocol (FreeTDS [4]) was used. The PHP was configured with the mssql module which uses FreeTDS library.

The decision has been made to host the Web Data Collaboration Server on the same machine. The server has been deployed under Tomcat (Tomcat 6.0.14).

Databases

The data in the scidata1.tc.cornell.edu database is sourced from the Internet Archive [5]. The database contains several smaller collections such as Amazon, Cornell and others that use the same database schema. More information about database schema and specifications can be found at <http://weblab1.tc.cornell.edu/documentation/schemadocs/>

Researchers' Tools

The second half of the semester was designated to rebuild and improve existing tools from the old <http://weblab.tc.cornell.edu> site written as the C# API (Application Programmer's Interface) .

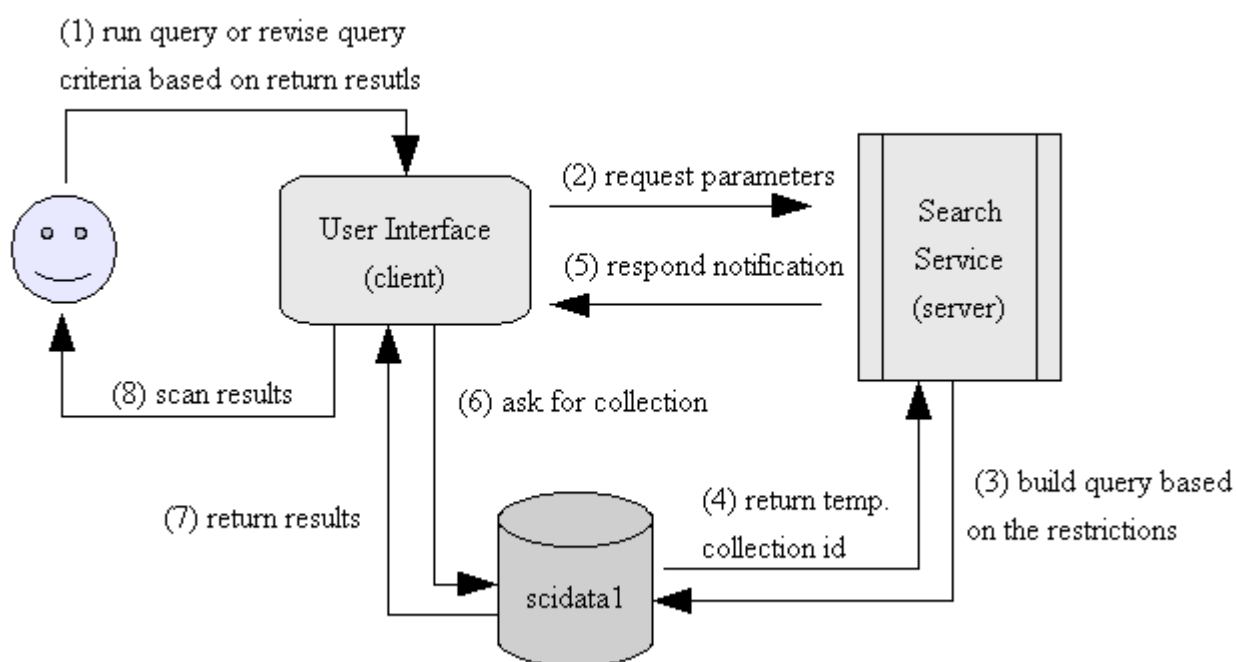
Researchers' Tools are essential part of the website purpose. There are two main sections of the users' tools that have been worked on:

- Input – allow the user to explore database by specifying search parameters;
- Output – provide the results as the effects of the users' input manipulation;

The design of the user interface must provide the basic functionality that can be efficiently utilized for the user needs. One of the goals in a good interface design was to satisfy the user necessities and provide quality feedback about the user data exploitation.

GetPages tool

This tool [6] is used to retrieve a subset of the data from a Web Lab database collections, by specifying restrictions that are used to select which data to retrieve. Returned results can be viewed on-line or directly from the database.



GetPages tool evaluation.

Properties of the fielded search for the GetPages tool:

- **PageID** is the hash of the full url with protocol and port and the archive time of the page;
- **UrIID** is the hash of the url of the page (w/o protocol or port);
- **HostID** is the hash of the hostname;
- **CrawlID** is a crawl number. Each crawl in the database has its own unique ID. `SELECT * FROM DBO.CRAWL` will return the list of all available crawls per particular collection;
- **ArchiveTime** is the time at which the web crawler archived the page. (e.g. '2005-02-09 15:42:43');
- **URL:Protocol** is the transfer protocol (e.g. 'http://');
- **URL:Host** is the url domain (e.g. 'edu' or 'org');
- **URL:Port** is the port number (e.g. ':80' or '8080');
- **Document Title** is the title of the web page. (This is tagged by a pair of tags `<title>` and `</title>`);
- **URL:Path** is the path of the url not including the filename extension (e.g. '/courses/cs430/index');
- **URL:Extension** is the filename extension (e.g. '.html' or '.asp');
- **IPAddress** is the IP Address of the server from which the page was retrieved;
- **QueryString** is some text after the "?" (e.g. '?fileid=7&download=US');
- **MIMETYPE** is the type of the web page (e.g. 'text/html');
- **Language** is the web page language translation (e.g. 'zh-CN.gb2312 0.70454 1597');

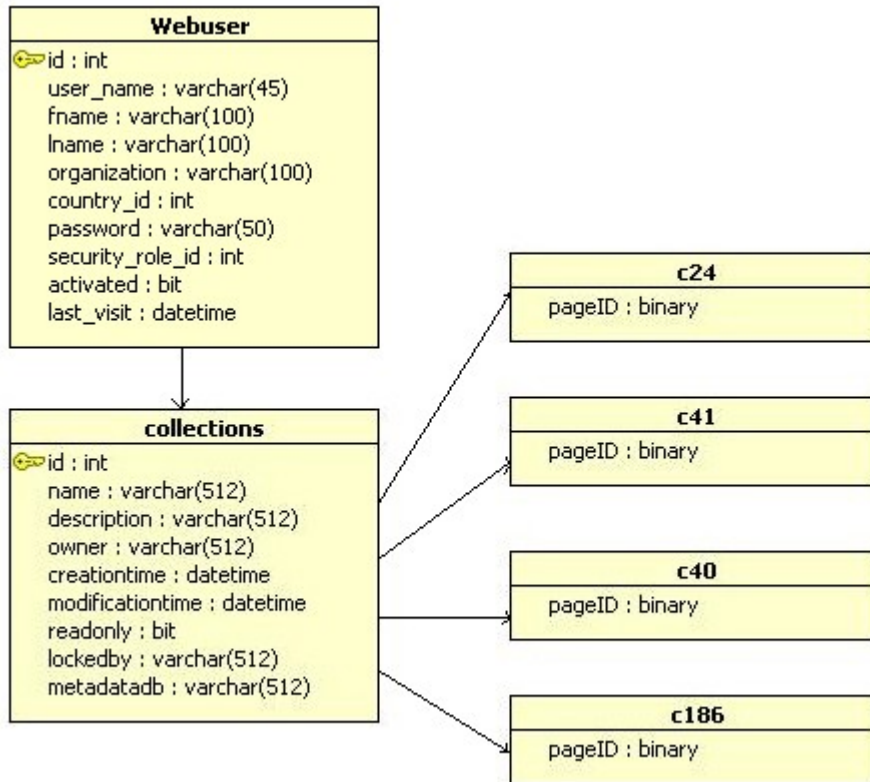
Each search field can be selected as a negate restriction, that is to select all data not matching a particular restriction. A blank restriction is treated as no restriction. To run a query at least one of the restrictions has to be selected.

GetPages interface functionality:

- Ability to select collection; currently available (WebLibraryAmazon and WebLibraryCornell)
- Allow user to specify search restrictions;
- Run revise queries;
- Store previous run queries (under development);
- View or download search results (under development);

Temporary Collections:

Each search results are stored as a subset of PageIDs in the temporary collection tables.



Temporary Collections Storage

Results:

- Search service creates a subset of PageIDs;
- Each subset is stored in the WebExtraction database as a temporary collection;
- Each user has its own set of temporary collections stored in the dbo.collections table specified by the pair of keys {id and owner};

Get Crawls

This tool [7] provides to the user information about database's Crawls. The main use for this tools is to view what crawls are available per specific database and what ids they were assigned.

Service on the Web Data Collaboration Server

The main goal of the search service design was to develop a service which could be used among different clients and different programming languages. Current Web Data Collaboration Server design was developed in GWT [8] environment. The GWT framework supports GWT-RPC protocol which works only in the GWT environment. In order to allow for the communication from other clients without changing the server design, different protocols were considered: SOAP, XML-RPC, JSON-RPC, HTTP. The best and the most flexible option was JSON-RPC protocol which is described in details later in this document. The main functionality of the search service is presented below:

SearchServer

Collection fetches a collection of archived pages, based on the given restrictions, from the specified database.

```
public Collection  
getSearchResult(String user, String  
database, Collection restrictions)  
throws DatabaseException,  
ResourceException;
```

StringBuilder generates SQL code for creating a temporary collection of pages from the specified database, based on the given search restrictions.

```
private StringBuilder  
createSearchCode(String user, String  
database, Collection<Restriction>  
restrictions) {}
```

Collection Server

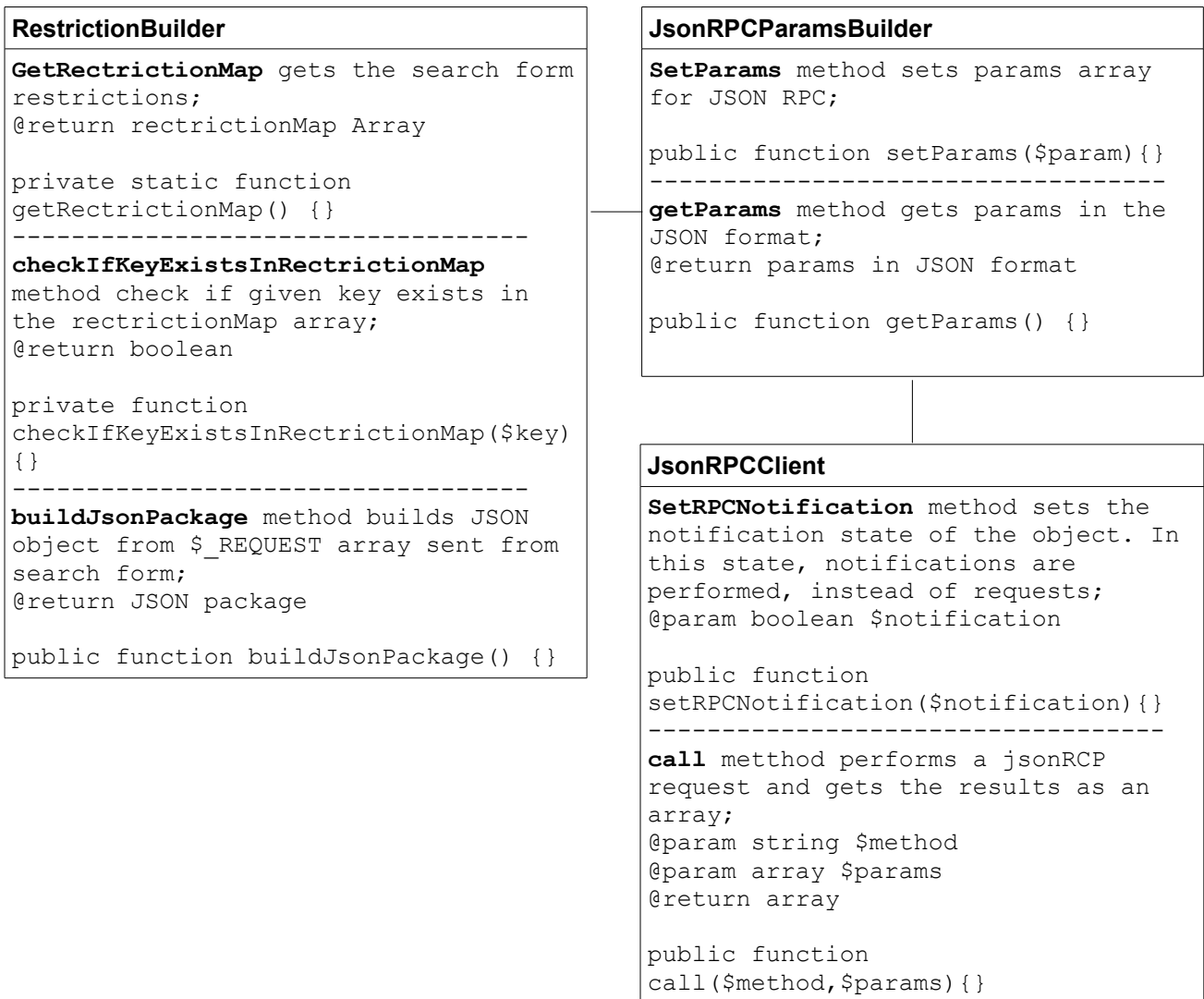
createCollection creates a new collection in the *extraction database.

```
public  
edu.cornell.cs.webarchive.wrapper.c  
ontent.Collection createCollection(  
String user,  
String database,  
StringBuilder command,  
String collectionName,  
String collectionDescription,  
boolean readOnly  
) throws DatabaseException,  
SQLException
```

Clients

Two clients were developed in order to use the search service presented above. The first client is part of the Web Lab website. The client was written in PHP and it uses JSON-RPC protocol to communicate with Search Server. The second client is part of the Web Data Collaboration Server and it is using Google Web Toolkit (GWT) framework and GWT-RPC protocol.

PHP client functionality:



GWT client functionality:

```

SearchGUI
Get the collection of restrictions
from the search fields table.

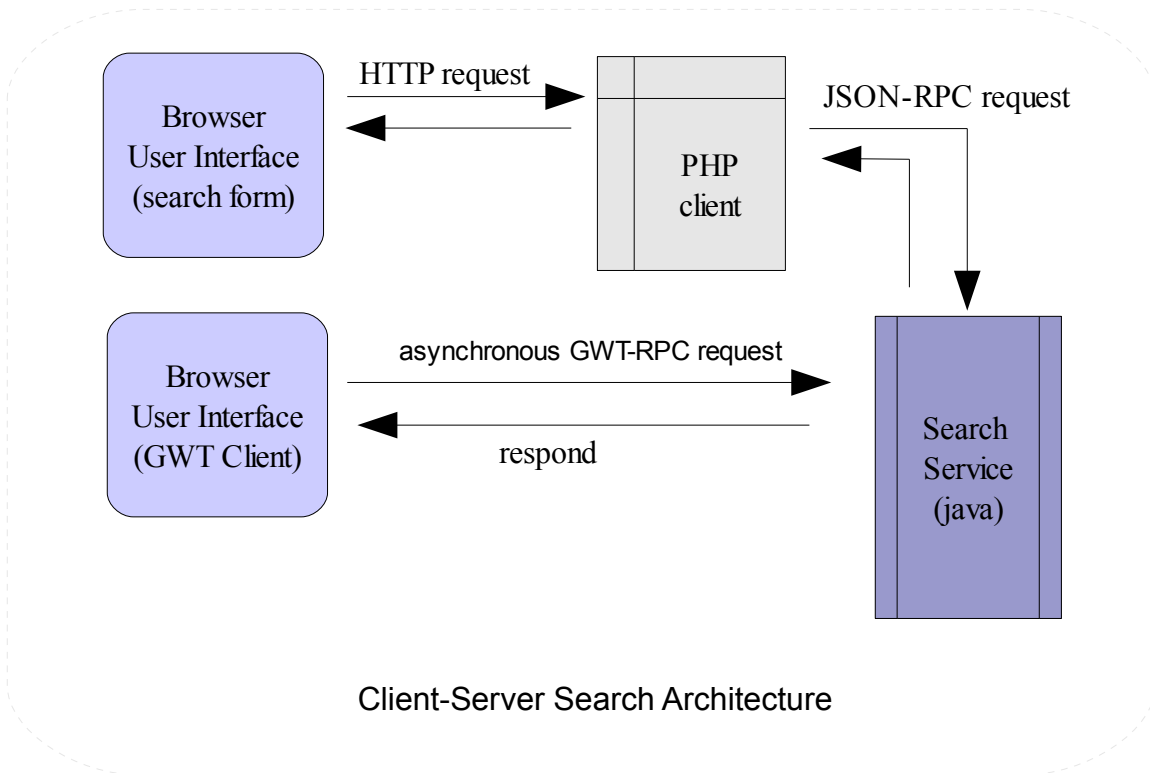
private Collection
createRestrictions() {}
    
```

```

SearchServerAsync
getSearchResult fetches a collection
of archived pages, based on the
given restrictions, from the
specified database.

public void getSearchResult(String
user, String database, Collection
restrictions, AsyncCallback
callback) throws DatabaseException,
ResourceException;
    
```

Client-Server Search Architecture for Web Lab Project



JSON-RPC protocol

JSON-RPC (jabsorb [9] library) protocol was used to accomplish the communication between Search Server and clients.

Advantages of using JSON-RPC protocol:

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

JSON schema used to communicate with Search Service

An Example of JSON [10] schema being used with Search Service:

```
{
  "method" : "searchService.getSearchResult",
  "params" : [
    "user1",
    "WebLibraryAmazon",
    {
      "javaClass" : "java.util.ArrayList",
      "list" : [
        {
          "javaClass" : "edu.cornell.cs.webarchive.wrapper.content.Restriction",
          "restrictionType" : "PageID",
          "negateRestriction" : false,
          "restrictionValues" : {
            "javaClass" : "java.util.HashMap",
            "map" : { "1" : "test1", "2" : "12" }
          }
        }
      ]
    }
  ],
  "id" : 1
}
```

The example above calls a remote method `getSearchResult(String user name, String library, Collection collection)` from the `SearchService`. The "list" element defines the method parameters. In this case the search method takes 3 parameters: String, String and Collection. Collection items are `edu.cornell.cs.webarchive.wrapper.content.Restriction`. In the case of regular Java Beans, an extra field `javaClass` is added that maps the typeless object back to the Java class.

Future Work

Forthcoming work from the Web Lab site point of view will require evaluation of Researchers' Tools and usability testing. There are several other components that are still under development such as temporary collection retrieval and security issues.

From the Web Data Collaboration side; current focus is on the Web Data Collaboration client (GWT-Client) and how to solve the interdependencies between various GUI components. Future work needs to focus on other web services in Web Data Collaboration Server.

Acknowledgement

This work is part of the Web Laboratory, which is a joint project of Cornell University and the Internet Archive. We would like to acknowledge the guidance and constant support by our advisers: Professor William Arms and Dr. Felix Weigel. Furthermore we would like to thank other members of the team: Manuel Calimlim, Daniela Balmus -Solomon and Blazej Kot.

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] CodeIgniter: Open source PHP web application framework: http://codeigniter.com/user_guide/
- [3] Model-view-controller: <http://en.wikipedia.org/wiki/Model-view-controller>
- [4] FreeTDS libraries: <http://www.freetds.org/>
- [5] Internet Archive: <http://www.archive.org/index.php>
- [6] The Web Lab: Researchers' Tools: GetPages: <http://weblab1.tc.cornell.edu/tools/getpages>
- [7] The Web Lab: Researchers' Tools: GetCrawls: <http://weblab1.tc.cornell.edu/tools/getcrawls>
- [8] Google Web Toolkit - Google Code: <http://code.google.com/webtoolkit/>
- [9] Jabsorb: <http://code.google.com/p/jabsorb/wiki/Manual>
- [10] JSON-RPC draft: <http://json-rpc.org/wd/JSON-RPC-1-1-WD-20060807.html>