Fall, 2007 CIS 550

Database and Information Systems

Project Specifications

Your "mash-up" system will broadly consist of four components:

- **Web crawler**. You will need to fetch Web pages from the Philadelphia Inquirer and Craigslist in order to find apartment listings.
- **Database** (aka data warehouse, index). The database will maintain (1) apartment listing entries and all associated data, (2) user account information, and (3) user "bookmarks" (saved keyword queries that can be reloaded).
- **User interface**. The user interface will be Servlet based. To display a map, you will use the Google Maps Web API, passing along an XML file consisting of the apartment listings (as labels) and locations matching a query.

We briefly describe the components.

Web Crawler

The Web crawler starts by opening a network (TCP) socket to a Web server: in this case, the servers are for the Philly Inquirer and Craigslist, and the URLs to the default pages are:

- http://www.phillyforrent.com/rentals_results/All_Philadelphia
- http://philadelphia.craigslist.org/apa/index.rss

You will issue an HTTP "GET" request to fetch these pages. Details on Java sockets and HTTP are at:

- http://www.davidreilly.com/java/java_network_programming/
- http://www.jmarshall.com/easy/http/

Next, once you have read the page from the socket, you will need to do two things:

1. Identify URLs that point to other pages that need to be fetched ("Next" pages, Craigslist links).
2. Parse the current document to extract any apartment listing information.

Parser

Writing a parser for HTML or text content is quite challenging – this will give you a true feel for why integrating arbitrary Web data is hard!

You may want to make use of the JTidy parser: http://java-source.net/open-source/html-parsers/jtidy. This will convert an arbitrary HTML document into a DOM tree, much like the XML Data Model we described in lecture. You will receive a pointer to the root DOMNode, and from this you can ask for children, siblings, etc. in a recursive fashion.

Fortunately, the Philly Inquirer listings are inserted into an HTML table, and you can get database attributes from that. For instance, a portion of the HTML looks like the following:

```
<tr class="sr_oddrow">
<td class="sr_listingcol" valign="top">
<table …>
<tr>
<td valign="top" rowspan="3">…</td>
<td valign="top" style="width:100%;">…The Club at Locust Grove – T…</div>
<div class="sr_location">Deptford</div>…
</td></tr>
<tr><td valign="bottom">…
<td class="sr_listingcol" valign="top" ALIGN=center>
<div class=sr_bedrooms>3</div>
</td>
<td class="sr_listingcol" valign="top" ALIGN=center>
<div class=sr_bedrooms>2 Full</div>
</td>
<td class="sr_listingcol" valign="top" ALIGN=center >
<div class="sr_buildingtype">Apartment Unfurnished</div>
</td>
<td class="sr_listingcol" valign="top" ALIGN=center>
<div class=sr_price>$1,485</div>
</td>
```

You'll note that there is a lot of semantic information in the tags.

For Craigslist, your job is significantly harder: for each listing, you *can* get an exact location (including a Google Maps entry -- very useful!), but all other data is simply described in text. You will need to look for the following substrings in the text:

- "bed" in the title OR description: find a number prior to "bed" and you should get the number of bedrooms.
- "bath": as above.
- "$" in the title: the number after should represent the monthly price.
- extract the complete text as a description attribute.

Database

All listings will be archived in a database, with entries for price, number of bedrooms, location, contact info, zip code, and so on. These will ultimately be searchable through a Web form.

They will additionally be indexed by keyword so users can search by words. We expect the keyword inverted index to be stored in relational tables, using word IDs, document IDs, occurrence information, etc., as in a number of examples throughout the semester.

Additionally, you will need entries for users, passwords, and user bookmarks.

User Interface

There are several modules to the user interface, which should be implemented as a Java servlet.

Login/query screen. The user will be able to pose a query as an anonymous user, or log in with a password. Anonymous users will have the ability to query based on a combination of keywords and attribute constraints (price, location, number of bedrooms, pets allowed).

User home/query screen. For logged in users, there should be a list of saved searches ("bookmarks") as well as a query interface. Users will have the ability to query based on a combination of keywords and attribute constraints (price, location, number of bedrooms, pets allowed). Users should be able to change their password and to delete bookmarks.

Apartment-search screen. The results will be projected onto a Google Map and should also have clickable ``detailed'' entries. The Google Maps API can be found at:

- http://code.google.com/apis/maps/
- http://code.google.com/apis/maps/documentation/index.html
- http://www.econym.demon.co.uk/googlemaps/

If the user is logged in, he or she should be able to click a button to add the query to his or her bookmarks.

See http://www.housingmaps.com/ for an example of how *some* of this might look.

Apartment detail view. When a user clicks on a listing, he or she should be taken to a detailed view that shows the apartment listing, a street-level map of the apartment, and a list of other query results that share the same zipcode. (For extra credit, photos from Flickr that are geocoded to be within the same neighborhood should be shown.)

Implementation Requirements

We expect you to use Java Servlets running on the SEAS Apache Tomcat server for your project. We also expect your listing fetcher to convert from HTML to XML before loading the data into the data warehouse (which should be a database on the SEAS Oracle instance used for your homework assignments).

We expect that the keyword search capabilities will be implemented using an inverted index implemented over relational tables, and that your keyword search will be based on SQL queries.

Additionally, you may use Eclipse and SAXON (installed in the computer labs) to do your Java and XQuery development.

Milestones

We expect your team to submit your work at the following milestones:

1. Database schema design, due 11/14.
2. Crawler + parser for Philly Inquirer listings, due 11/21 (should be done in parallel with the first one!)
3. Crawler + parser for Craigslist, due 11/28 (should be done in parallel with the other tasks!)
4. User login screen, due 11/28.
5. Full project, due 12/14.