Web Search Interfaces

Web Search Interface

- Web search engines of course need a web-based interface.
- Search page must accept a query string and submit it within an HTML <form>.
- Program on the server must process requests and generate HTML text for the top ranked documents with pointers to the original and/or cached web pages.
- Server program must also allow for requests for more relevant documents for a previous query.

Submit Forms

- HTML supports various types of program input in forms, including:
  - Text boxes
  - Menus
  - Check boxes
  - Radio buttons
- When user submits a form, string values for various parameters are sent to the server program for processing.
- Server program uses these values to compute an appropriate HTML response page.
What's a Servlet?

- Java's answer to CGI programming for processing web form requests.
- Program runs on Web server and builds pages on the fly.
- When would you use servlets?
  - Page is based on user-submitted data e.g search engines.
  - Data changes frequently e.g. weather-reports.
  - Page uses information from a databases e.g. on-line stores.
- Requires running a web server that supports servlets.

Basic Servlet Structure

```java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SomeServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // handle get request
        PrintWriter out = response.getWriter(); // send content to browser
    }
}
```
A Simple Servlet

```java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("Hello World");
    }
}
```

Generating HTML

```java
public class HelloWWW extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<HTML>
        <HEAD><TITLE>HelloWWW</TITLE></HEAD>
        <BODY>
            <H1>Hello WWW</H1>
        </BODY></HTML>" );
    }
}
```

HTML Post Form

```html
<FORM ACTION="/servlet/hall.ThreeParams" METHOD="POST">
First Parameter:  <INPUT TYPE="TEXT" NAME="param1">
Second Parameter: <INPUT TYPE="TEXT" NAME="param2">
Third Parameter:  <INPUT TYPE="TEXT" NAME="param3">
<CENTER>
    <INPUT TYPE="SUBMIT">
</CENTER>
</FORM>
```
public class ThreeParams extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
            IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<UL>
" +
            "<LI>param1: " + request.getParameter("param1") + 
            "<br>
" +
            "<LI>param2: " + request.getParameter("param2") + 
            "<br>
" +
            "<LI>param3: " + request.getParameter("param3") + 
            "<br>
" +
            "</UL>
" +
    }
    public void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
            IOException {
        doGet(request, response);
    }
    }
}
Session Tracking

- Typical scenario – shopping cart in online store.
- Necessary because HTTP is a "stateless" protocol.
- Common solutions: Cookies and URL-rewriting.
- Session Tracking API allows you to:
  - Look up session object associated with current request.
  - Create a new session object when necessary.
  - Look up information associated with a session.
  - Store information in a session.
  - Discard completed or abandoned sessions.

Session Tracking API - I

- Looking up a session object:
  - HttpSession session = request.getSession(true);
  - Pass true to create a new session if one does not exist.
- Associating information with session:
  - session.setAttribute("user", request.getParameter("name"))
  - Session attributes can be of any type.
- Looking up session information:
  - String name = (String) session.getAttribute("user")

Session Tracking API - II

- getId
  - The unique identifier generated for the session.
- isNew
  - true if the client (browser) has never seen the session.
- getCreationTime
  - Time in milliseconds since session was made.
- getLastAccessedTime
  - Time in milliseconds since the session was last sent from client.
- getMaxInactiveInterval
  - # of seconds session should go without access before being invalidated.
  - Negative value indicates that session should never timeout.
**Simple Search Servlet**

- Based on directory parameter, creates or selects existing InvertedIndex for the appropriate corpus.
- Processes the query with VSR to get ranked results.
- Writes out HTML ordered list of 10 results starting at the rank of the start parameter.
- Each item includes:
  - Link to the original URL saved by the spider in the top of the document in BASE tag.
  - Name link with page <TITLE> extracted from file.
  - Additional link to local cached file.
- If all retrievals not already shown, creates a submit form for “More Results” starting from the next ranked item.

**Simple Search Interface Refinements**

- For “More results” requests, stores current ranked list with the user session and displays next set in the list.
- Integrates relevance feedback interaction with “radio buttons” for “NEUTRAL,” “GOOD,” and “BAD” in HTML form.

**Other Search Interface Refinements**

- Highlight search terms in the displayed document.
  - Provided in cached file on Google.
- Allow for “advanced” search:
  - Phrasal search (“..”)
  - Mandatory terms (+)
  - Negated term (-)
  - Language preference
  - Reverse link
  - Date preference
Clustering Results

- Group search results into coherent “clusters”:
  - “microwave dish”
    - One group on food recipes or cookware.
    - Another group on satellite TV reception.
  - “Austin bats”
    - One group on the local flying mammals.
    - One group on the local hockey team.
- Northern Light used to group results into “folders” based on a pre-established categorization of pages (like DMOZ categories).
- Alternative is to dynamically cluster search results into groups of similar documents.

User Query Length

- Users tend to enter short queries.
  - Study in 1998 gave average length of 2.35 words.
- Evidence that queries are getting longer.

Speech Queries are Longer

<table>
<thead>
<tr>
<th>Percentage of U.S. visits by number of keywords</th>
<th>Jan-06</th>
<th>Dec-06</th>
<th>Jan-09</th>
<th>Year-over-year percent change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 word</td>
<td>20.06%</td>
<td>20.70%</td>
<td>20.29%</td>
<td>2.42%</td>
</tr>
<tr>
<td>2 words</td>
<td>24.91%</td>
<td>24.13%</td>
<td>23.00%</td>
<td>-4.4%</td>
</tr>
<tr>
<td>3 words</td>
<td>21.05%</td>
<td>21.54%</td>
<td>19.10%</td>
<td>-8.46%</td>
</tr>
<tr>
<td>4 words</td>
<td>14.56%</td>
<td>14.60%</td>
<td>14.60%</td>
<td>0.00%</td>
</tr>
<tr>
<td>5 words</td>
<td>8.23%</td>
<td>9.37%</td>
<td>9.68%</td>
<td>3.60%</td>
</tr>
<tr>
<td>6 words</td>
<td>4.52%</td>
<td>4.67%</td>
<td>4.99%</td>
<td>6.80%</td>
</tr>
<tr>
<td>7 words</td>
<td>2.23%</td>
<td>2.16%</td>
<td>2.09%</td>
<td>-3.42%</td>
</tr>
<tr>
<td>8+ words</td>
<td>2.61%</td>
<td>3.61%</td>
<td>3.42%</td>
<td>-5.90%</td>
</tr>
</tbody>
</table>

Source: Nielsen, an Experian company.