Homework: Feed Reader (With and Without Offline Capability)

1. Objectives

- Become familiar with the processing of AJAX requests.
- Use a combination of HTML, CSS, DOM, XML, JSON, SQLlite, and Java Servlets;
- Provide a Google Reader-like interface for reading feeds, and make it work offline.

2. Background

2.1 Google Gears

Google Gears is an open source browser extension that lets developers create web applications that can run offline. It includes three modules:

- LocalServer - Cache and serve application resources (HTML, JavaScript, images, etc.) locally;
- Database - Store data locally in a fully-searchable relational database;
- WorkerPool - Make your web applications more responsive by performing resource-intensive operations asynchronously..

The "Google Gears API Developer's Guide" homepage is available at:

http://code.google.com/apis/gears/

2.2 Atom/RSS

ATOM and RSS are two different XML formats for syndicating information on the web. They convey similar information but vary slightly in syntax.
ATOM is the standard that Google data APIs are based on, so familiarity with the ATOM protocol will aid you in working with any of the Google data APIs.

More info on RSS: http://en.wikipedia.org/wiki/RSS_(file_format)
More info on ATOM: http://en.wikipedia.org/wiki/Atom_(standard)

3. Description of Exercise

To implement an off-line Feed Reader interface you are required to write a combination of HTML, JavaScript, JSON and Java Servlet programs.

3.1 Feed Reader without Gears

UI Requirements:

- Text input for typing feed URL
- Button for loading feed
- Sidebar for displaying loaded feed titles
- Main area for displaying entries from selected feed

- CSS to distinguish between read and not-read entries

Code requirements:

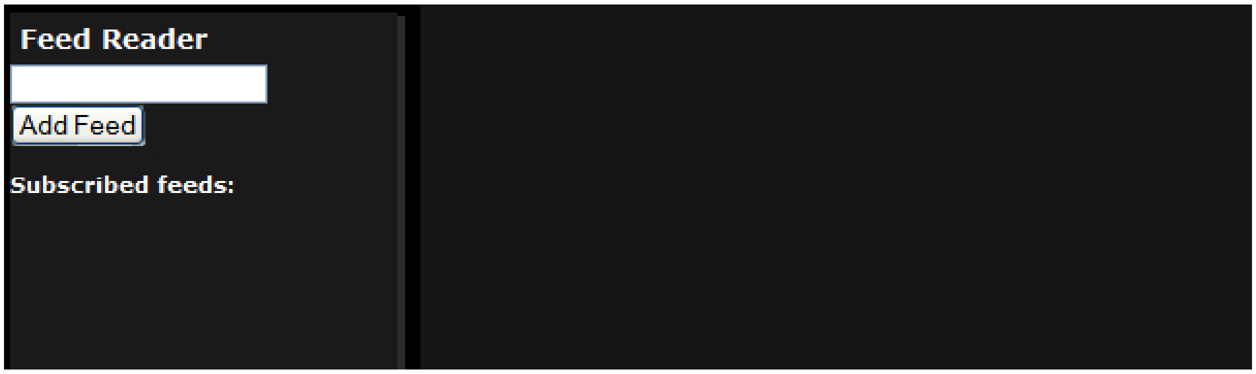
- When user enters valid feed URL and presses button, the code will go out and load the feed via a Java servlet running in Tomcat.
- The Java servlet will open the URL, sense whether the feed is ATOM or RSS, and output JSON in the following format:

```
{feed:
 {title: "Feed title",
  link: "http://google.com/feed.xml",
  entries: [
   {title: "Entry #1 title",
    link: "http://google.com/entry1.html",
    content: "Content of exciting entry"},
   {title: "Entry #2 title",
    link: "http://google.com/entry2.html",
    content: "Content of other exciting entry"},
  ]}
}
```

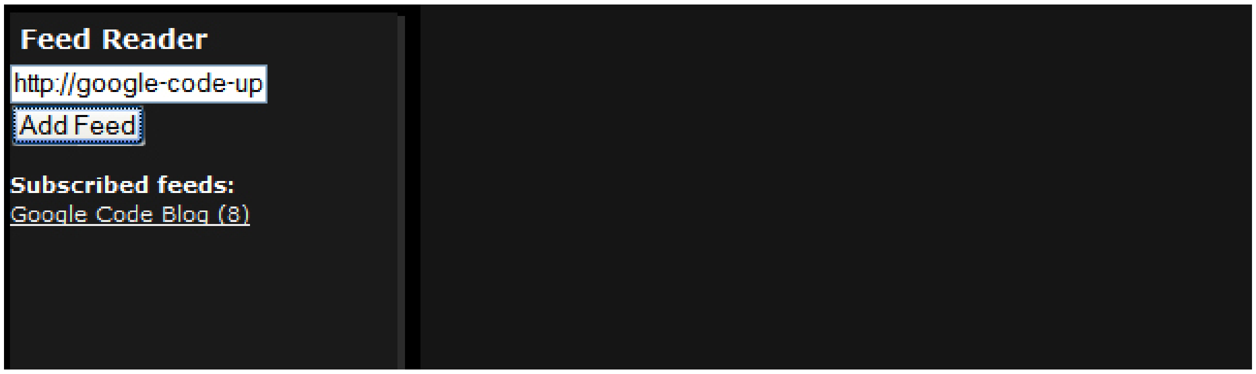
- The translation between the ATOM/RSS elements and the JSON output is the following:
  - title
    - The feed title. Corresponds to the <title> element in Atom and the <title> element in RSS.
  - link
    - The URL for the HTML version of the feed. Corresponds to the <href> attribute of the <link> element in Atom and the <link> element in RSS.
  - `entries[]`
    A list of all of the entries in the feed. Corresponds to the <entry> element in Atom and the <item> element in RSS.
    - `title`
      The entry title. Corresponds to the <title> element in Atom and the <title> element in RSS.
    - `link`
      The URL for the HTML version of the entry. Corresponds to the <link> element in Atom and the <link> element in RSS.
    - `content`
      The body of this entry, including HTML tags. Since this value can contain HTML tags, you should display this value using `elem.innerHTML = entry.content` (as opposed to using `document.createTextNode`). Corresponds to the <content> or <summary> elements in Atom and the <description> element in RSS.
- When the Java servlet returns the JSON, the JavaScript will add the feed to a global data structure, and add a link to the sidebar for that feed.
- When the user clicks on a feed title in the sidebar, the code will iterate through the feed entries and display each of them (with title, content, and link) in the main area. There should be a button that the user can click in each entry to mark it as read. The CSS should change depending on whether an entry is read or not.
- When a user marks an entry as read, the sidebar should update to reflect the number of un-read entries in a feed.

Screenshots of User Experience:

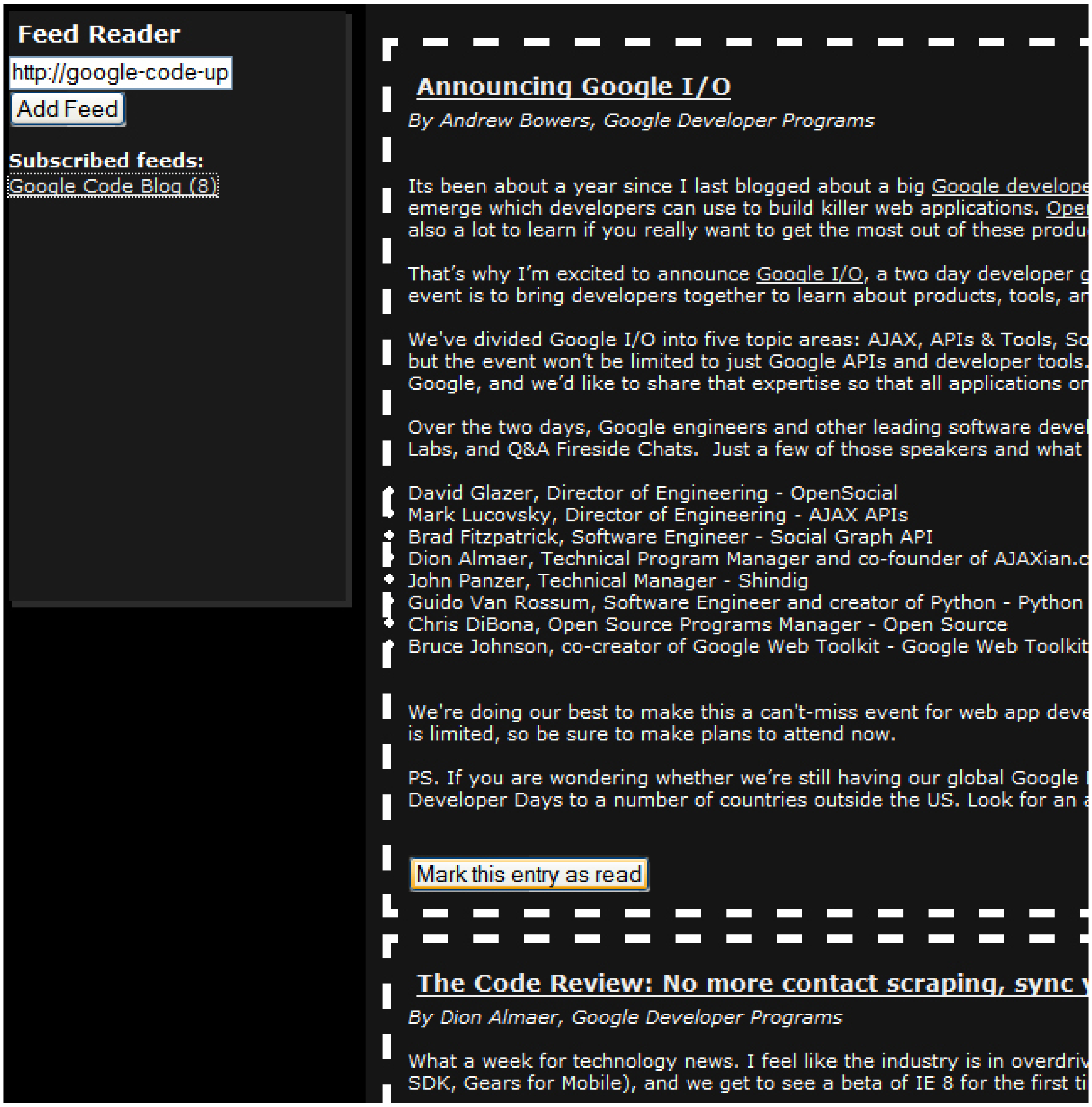
Before the user has done anything:



After the user enters a valid feed URL and clicks the "Add Feed" button. Notice the "Google Code Blog (8)" link. 8 refers to the number of unread entries (all of them, to begin with)



After the user clicks on the feed title in the sidebar. Notice the "Mark Entry as Read" button.


Feed Reader

http://google-code-up

Add Feed

Subscribed feeds:
Google Code Blog (8)

Announcing Google I/O

By Andrew Bowers, Google Developer Programs

Its been about a year since I last blogged about a big Google develope
emerge which developers can use to build killer web applications. Oper
also a lot to learn if you really want to get the most out of these produ

That's why I'm excited to announce Google I/O, a two day developer g
event is to bring developers together to learn about products, tools, an

We've divided Google I/O into five topic areas: AJAX, APIs & Tools, So
but the event won't be limited to just Google APIs and developer tools.
Google, and we'd like to share that expertise so that all applications or

Over the two days, Google engineers and other leading software devel
Labs, and Q&A Fireside Chats. Just a few of those speakers and what

- David Glazer, Director of Engineering - OpenSocial
- Mark Lucovsky, Director of Engineering - AJAX APIs
- Brad Fitzpatrick, Software Engineer - Social Graph API
- Dion Almaer, Technical Program Manager and co-founder of AJAXian.c
- John Panzer, Technical Manager - Shindig
- Guido Van Rossum, Software Engineer and creator of Python - Python
- Chris DiBona, Open Source Programs Manager - Open Source
- Bruce Johnson, co-creator of Google Web Toolkit - Google Web Toolkit

We're doing our best to make this a can't-miss event for web app deve
is limited, so be sure to make plans to attend now.

PS. If you are wondering whether we're still having our global Google
Developer Days to a number of countries outside the US. Look for an a

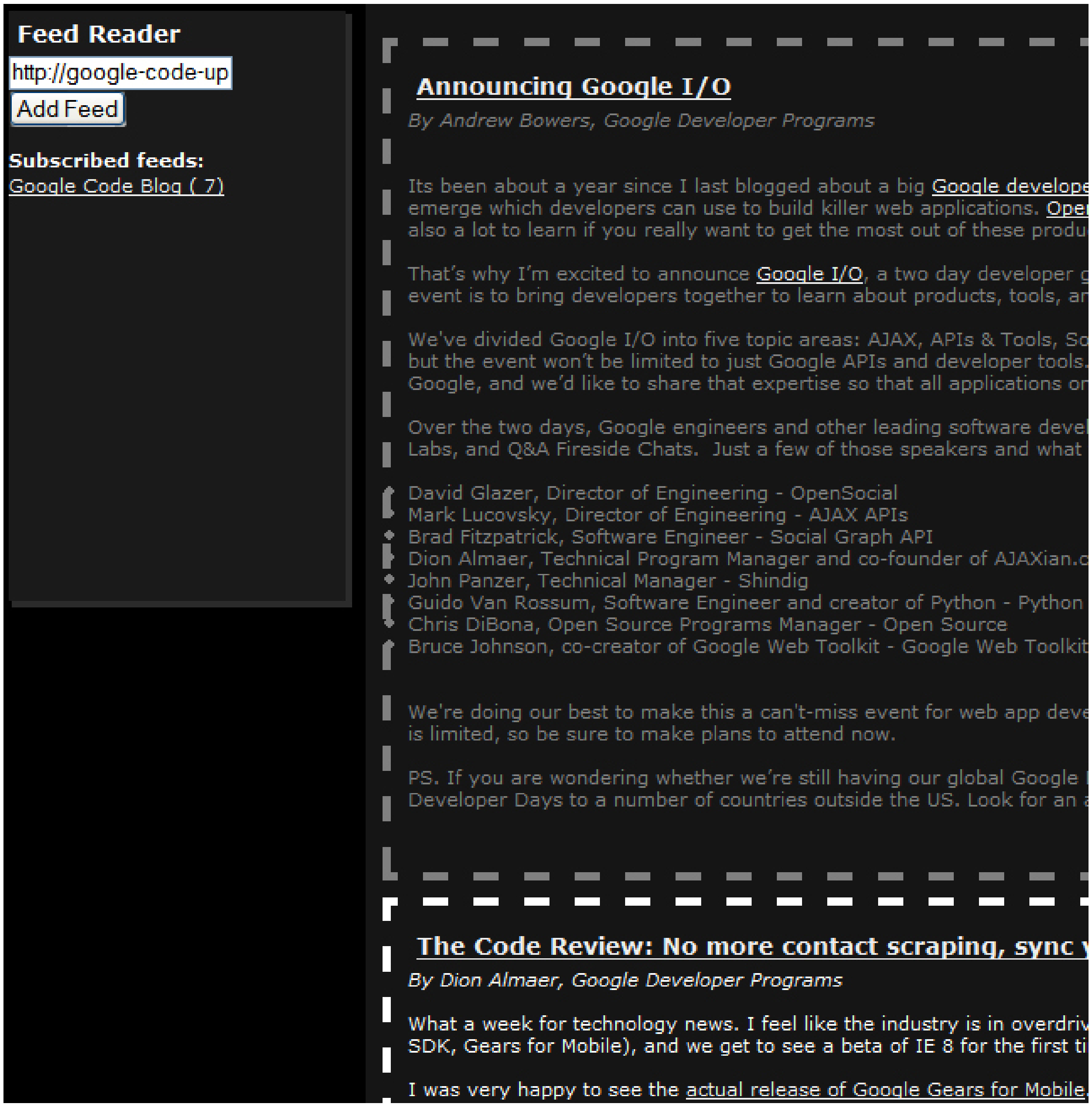
Mark this entry as read

The Code Review: No more contact scraping, sync y

By Dion Almaer, Google Developer Programs

What a week for technology news. I feel like the industry is in overdriv
SDK, Gears for Mobile), and we get to see a beta of IE 8 for the first ti

After the user clicks "Mark this entry as read" for the first entry. Notice the entry now has a grey border and grey font color and the "Mark this entry as read" has disappeared.


Feed Reader

http://google-code-up

Add Feed

Subscribed feeds:
Google Code Blog (7)

Announcing Google I/O

By Andrew Bowers, Google Developer Programs

Its been about a year since I last blogged about a big Google develope
emerge which developers can use to build killer web applications. Oper
also a lot to learn if you really want to get the most out of these produ

That's why I'm excited to announce Google I/O, a two day developer g
event is to bring developers together to learn about products, tools, ar

We've divided Google I/O into five topic areas: AJAX, APIs & Tools, So
but the event won't be limited to just Google APIs and developer tools.
Google, and we'd like to share that expertise so that all applications or

Over the two days, Google engineers and other leading software devel
Labs, and Q&A Fireside Chats. Just a few of those speakers and what

- David Glazer, Director of Engineering - OpenSocial
- Mark Lucovsky, Director of Engineering - AJAX APIs
- Brad Fitzpatrick, Software Engineer - Social Graph API
- Dion Almaer, Technical Program Manager and co-founder of AJAXian.c
- John Panzer, Technical Manager - Shindig
- Guido Van Rossum, Software Engineer and creator of Python - Python
- Chris DiBona, Open Source Programs Manager - Open Source
- Bruce Johnson, co-creator of Google Web Toolkit - Google Web Toolkit

We're doing our best to make this a can't-miss event for web app deve
is limited, so be sure to make plans to attend now.

PS. If you are wondering whether we're still having our global Google
Developer Days to a number of countries outside the US. Look for an a

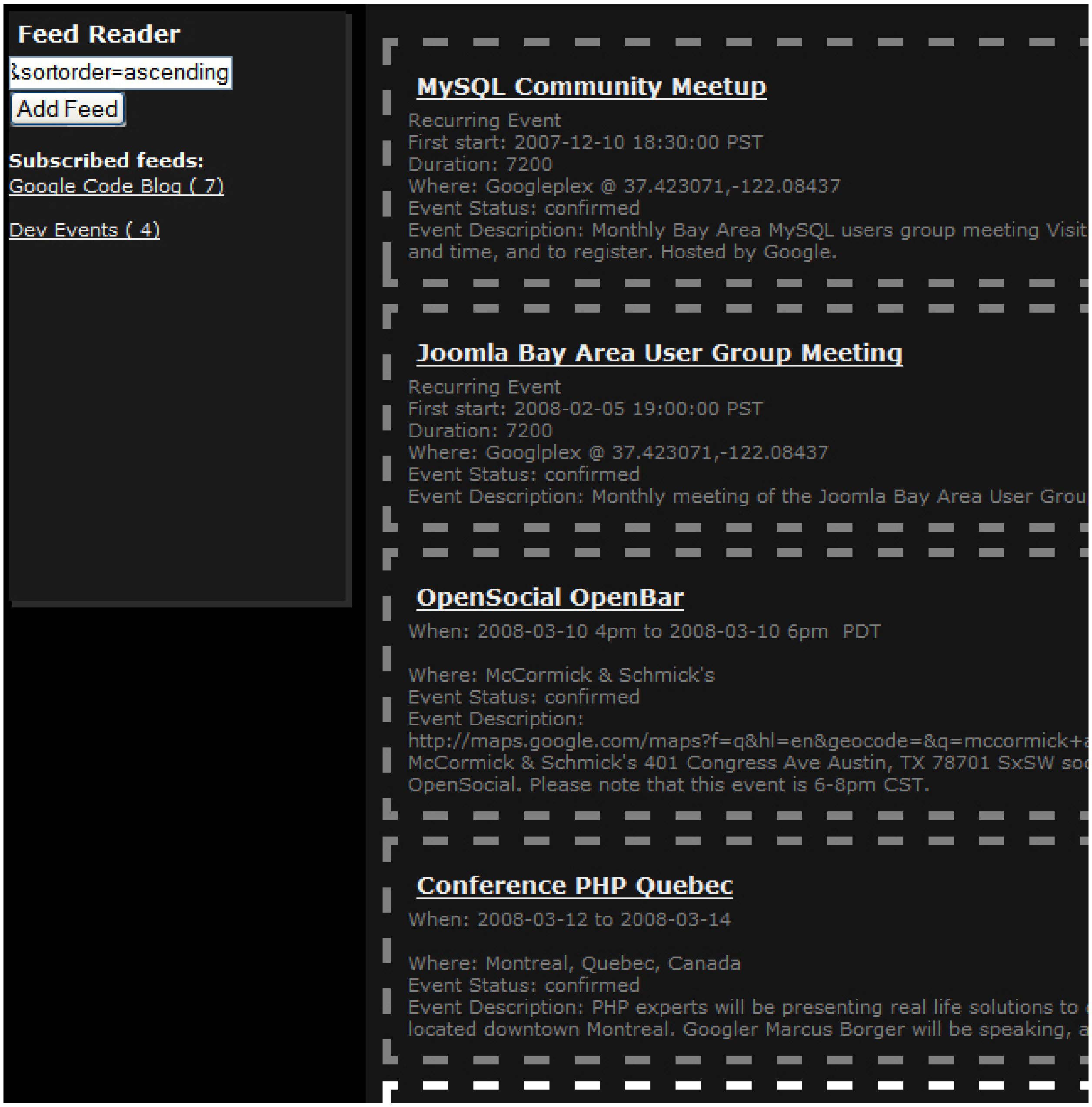
The Code Review: No more contact scraping, sync

By Dion Almaer, Google Developer Programs

What a week for technology news. I feel like the industry is in overdriv
SDK, Gears for Mobile), and we get to see a beta of IE 8 for the first ti

I was very happy to see the actual release of Google Gears for Mobile.

After the user adds another feed URL, clicks on it in the sidebar, and marks the first four entries as read. Notice the 'Mark Entry as Read' button isn't displayed for them and that the sidebar says 'Dev Events (4)'.



Subscribed feeds:
Google Code Blog (7)

Dev Events (4)

MySQL Community Meetup
Recurring Event
First start: 2007-12-10 18:30:00 PST
Duration: 7200
Where: Googleplex @ 37.423071,-122.08437
Event Status: confirmed
Event Description: Monthly Bay Area MySQL users group meeting Visit
and time, and to register. Hosted by Google.

Joomla Bay Area User Group Meeting
Recurring Event
First start: 2008-02-05 19:00:00 PST
Duration: 7200
Where: Googlplex @ 37.423071,-122.08437
Event Status: confirmed
Event Description: Monthly meeting of the Joomla Bay Area User Grou

OpenSocial OpenBar
When: 2008-03-10 4pm to 2008-03-10 6pm PDT

Where: McCormick & Schmick's
Event Status: confirmed
Event Description:
http://maps.google.com/maps?f=q&hl=en&geocode=&q=mccormick+a
McCormick & Schmick's 401 Congress Ave Austin, TX 78701 SxSW soc
OpenSocial. Please note that this event is 6-8pm CST.

Conference PHP Quebec
When: 2008-03-12 to 2008-03-14

Where: Montreal, Quebec, Canada
Event Status: confirmed
Event Description: PHP experts will be presenting real life solutions to
located downtown Montreal. Googler Marcus Borger will be speaking, a

3.2 Feed Reader with Gears

UI Requirements:
- The UI will look exactly the same, except for the addition of two buttons: "Clear Store" and "Clear Database." This will be useful for your own debugging and the graders'.

Code Requirements:

- When the page loads up, the code will use Gears to create a Local Store and Database.
  - The LocalStore will store all the static files associated with the app: any HTML, CSS, images, JS.
  - The Database will have two tables for storing data retrieved via the app:
    - Feeds - Should store the feed URLs, titles, and number of unread entries.
    - Entries - Should store the associated feed URL (parent feed), an ID or number that identifies its position in the feed, title, link, content, and whether it's been read.
- Upon page load, the code will also create sidebar links for any feeds already in the Feeds table from previous use.
- When a user adds a feed, the code will add that feed to the Feeds table and each of its entries to the Entries table. It will also create a sidebar link for that feed.
- When the user clicks a feed, the code will fetch the entries information from the Entries table and display it in the main area.
- When the user clicks the "Mark as read" button, the code will update that entry in the Entries table to indicate that it's been read.

3.3. Browser Requirements

The program must work in both IE 6 and Firefox 2. If Google Gears is not installed, only the "online" behavior should be displayed (see section 3.1).

3.4. Bonus Steps #1

- Also let users "star" items and have an option in the sidebar to view all the starred entries.
- Detect whether entry has been read based on scrollbar position (like Google reader does).

Implementing any of the above, besides 100% of the exercise, will earn an additional point, for a maximum of 2 bonus points.

3.5. Alternative Feed Reader with Client-side database storage

Apple implemented a portion of the HTML5 Working Draft called "Client-side database storage" in their just released Safari 3.10 web browser for OS X and Windows. See these links for the details:

http://webkit.org/blog/126/webkit-does-html5-client-side-database-storage/

http://www.whatwg.org/specs/web-apps/current-work/multipage/section-sql.html

HTML5 Client-side database storage provides functionality almost identical to Google Gears, without the need to download and install a plug-in or add-on. If you implement **both** the Google Gears and the HTML5 Client-side database storage versions, you will be entitled to an additional 5.5 points (for a total of 11 points for Part II!). If you implement only one of the two off-line versions, you will be entitled to 5.5 points, as described in the grading guidelines.

If you implement this functionality, create an additional hyperlink entitled "Homework #8 -

Safari 3.1" in your class exercises home page.

4. Implementation Hints

Step 1: Use the Java servlet to retrieve a feed

- Follow the 'Deployment structure' guide below to setup a sample servlet. (Tip: Copy HelloWorld.java, rename, add new servlet mapping).
- Define your doPost function in the servlet such that it retrieves the feed from the user-input URL, determines the format (Atom or RSS) and outputs a JSON for the feed using the Atom-JSON or RSS-JSON mapping provided above.

Step 2: Use JavaScript to save a feed

- Write a function in Javascript which inserts a JSON into a global data structure and adds a link to the sidebar for that feed.

Step 3: Write the webpage to interact with the Java servlet

- First create the HTML for the form interface as shown in the above screenshots.
- Write a function to *retrieve the feeds,* which shows a link for each feed on the sidebar and a number of un-read entries in the feed.
- Write a function to *retrieve the entries in a feed* and display them in the main window with a "Mark this entry as read" button at the bottom of each entry. Use a CSS to control the color of each entry i.e. white for unread and grey for read.
- Write a function to *save feeds,* which issues a POST request with the user-input URL to the servlet, which in return outputs a JSON.

Step 4: Install Google Gears

- See section 5.4, *Prerequisites*, for the steps needed to obtain a Google Account and download Google Gears. Once you download and run the Google Gears installer, restart your browser. When using Firefox 1.5+ Google Gears will be listed in the Add-ons.
- Tips for using Google Gears: Disable the cache in your browser, and learn where the "Work Off-line" button is. Firebug on Firefox is also quite helpful, as it displays all the HTTP requests in the NET tab, with their headers and response.

Step 5: Write the gears version of the webpage

- Create a new HTML page based on the version you just created. Download 'gears_init.js' to your server and include it in a script tag.
- Write a function to setup the database, and call on page load. Your database table schema should consist of the Feeds and Entries tables.
- Write a function for saving the feed to the local database table, using the SQL 'insert' command.
- Write a function for retrieving from the local database table, using the SQL 'select' command.
- Write a function that will update the number of unread entries of a feed using the SQL 'update' command.
- Once everything is working, use the LocalServer to store the page files in a local store. You can test this by going into offline mode and then reloading the page. Note: this will cache your page forever, so you'll want to write a function to clear the local store if you'll ever make changes to the page after you store it and want to refresh your cache. Similarly, write a function to clear the database using the SQL 'delete' command.

5. Prerequisites

This homework requires the use of the following components:

1. A servlet-based web server, Tomcat 4.1.27. Instructions on how to load Tomcat 4.1.27 can be found here: http://www-scf.usc.edu/~csci571/2007Fall/tomcatinstall.html. A tar version of Tomcat 4.1.27 can be found here: http://www-scf.usc.edu/~csci571/download/jakarta-tomcat-4.1.27.tar.
2. The Java Servlet library, which has functionality similar to Perl's LWP library, to perform HTTP transactions using methods such as doGet() or doPost() from Java.
3. A Java XML parser library. You may use the JDOM 1.0, an object model that uses XML parsers to build documents, available in the Download section of the class website. Additional information on JDOM is available at http://www.jdom.org/. You may also use JAXP, the Java API for XML Parsing, version 1.1, included in the Java JDK 1.4 (import javax.xml.parsers.*) and documented at: http://java.sun.com/xml/jaxp/dist/1.1/docs/api/. A good tutorial on JAXP is available at http://www-106.ibm.com/developerworks/xml/library/x-jaxp1.html.
4. You will need to download install Google Gears. You can visit http://code.google.com/apis/gears/install.html to download and install Google Gears on your computer (Windows XP, Windows Vista or Mac OS X). You will need to have Firefox 1.5+ or Internet Explorer 6.0+ to install Google Gears.

6. Deployment Structure

To write your own Java Servlets program using Tomcat 4.1.27, you need to:

1. Successfully install Tomcat 4.1.27 on your machine.
2. Go to $CATALINA_HOME/webapps/examples directory.
3. Place the HTML, CSS and JavaScript (.js) files in the Tomcat servlets subdirectory.
4. Place your Java Servlets file (.java) in the /WEB_INF/classes folder. So the path of your Servlets file is http://server_name:port/examples/servlet/your_servlet_name
5. Add appropriate sections to the WEB-INF/web.xml file, as in:

```
<servlet>
<servlet-name>offline_feed</servlet-name>
<display-name>Offline Feed</display-name>
<servlet-class>OfflineFeed</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>offline_feed</servlet-name>
<url-pattern>/servlet/offline_feed</url-pattern>
</servlet-mapping>
```

6. To avoid UTFDataFormatException during file IO operation, you have to use JDK 1.3 or later for Tomcat. In the .cshrc file under your home directory, add the entries:

```
setenv JAVA_HOME /usr/j2se

setenv PATH /usr/j2se/bin:${PATH}
```

7. Before you issue a request to your Java Servlet file, you need to compile it. You might need a Java Servlet class to compile your code, so open the .cshrc file, and

add "`/home/scf-22/csci571/servlet_classfiles/j2ee.jar`" to your `CLASSPATH` variable.

8. Then run "`source .cshrc`" and restart your Tomcat server.

7. Material You Need to Submit

On your course homework page, your link for this homework should go to a page that includes your JavaScript/HTML program (a page similar to the one depicted in the picture in section 3). You should submit all source code files including HTML (.html), Cascading Style Sheets (.CSS), JavaScript (.js), Java Servlets (.java) and a README file electronically to the csci571 account so that it can be graded and compared to all other students' code via the MOSS code comparison tool.