2008

Firefox Extension to Add Contacts, Events, and View Addresses

Vijay Rao
San Jose State University

FIREFOX EXTENSION TO ADD CONTACTS, EVENTS AND VIEW ADDRESSES

A Writing Project

Presented to

The Faculty of Computer Science

San Jose State University

In Partial Fulfillment of the Requirement for the

Degree

Master of Science

By

Vijay Rao

May 2008

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Chris Pollett

Dr. Robert Chun

Sudip Chattopadhyay

Abstract

Users of the Firefox browser have the ability to download plugins to manage their contacts. This usually involves typing or copying the details from some source to add contacts. Event and meeting invitations are sent by mail and are added to the user's calendar once the user accepts the invitation. Users viewing address data on websites are limited to the mapping capabilities provided by the webpage viewed by the user.

We developed a Firefox extension that allows the user to select portions of text with contact or event information and add it as a contact or an event in the calendar of their existing mail client application such as: Microsoft Outlook, Thunderbird, etc. The data is automatically parsed to pick up relevant information such as name, street address, phone number, and email address in case of contacts and street addresses and event dates in case of event. The extension also allows users to right click on a webpage that has a tabular display of addresses and view these addresses on a maps application such as Google Maps.

Table of Contents

Index of Figures

Index of Tables

Index of Listings

1 Introduction

1.1 Overview

Emails, chats and web sites have mundane data such as contact information of friends and business contacts. Various events happening in certain areas are emailed or found on the web. Street Addresses present on web pages while viewing home listings or other information is usually not intuitive and dependant on the website providing a link to view the address in a map. Usually viewing one address at a time in a map can be tedious and counter-intuitive.

Various applications for managing contacts are present today including the standard mail clients such as Microsoft Outlook and Thunderbird. The calendars associated with such mail clients allow users to add events or meetings. The option of adding contacts usually involves the user either typing various details such as name, email address, phone number and other details or copying the text and pasting it in the correct placeholders in the form provided by the mail client for adding contacts. Similarly meeting invites are usually sent or received via email and get added to the user's calendar on acceptance of the same by the user.

This experience of adding contacts and events could be greatly enhanced if there was an automatic parsing ability provided by the browser which allowed the user to indicate a portion of the text as either a contact or event information and it was automatically added to the user's mail client application. The user could ideally edit the text to correct or enhance this information before it is saved.

Extending the functionality of the browser to meet these needs would be an ideal solution that makes the process very intuitive and natural, saving the user from typing this information. Also adding the ability to view tabular listing of US Street addresses in a cluster enhances the user's viewing experience allowing the user to interact and assimilate the data in a better manner. The user who is viewing these addresses usually has a reference or a landmark address that when viewed along with the other addresses which gives him a new perspective and context.

1.2 The Project

Mozilla's Firefox browser has a robust architecture for developers to build and deploy extensions. Our project involved extending the Firefox browser to allow users to easily and intuitively add contacts and events to their existing mail clients with minimal effort. Adding the ability for users to view a collection of addresses from a web page on a Google Maps application with the ability to add additional landmarks improved the user's browsing experience. This was possible by adding additional options in the context click of the mouse and the user selecting portions of text on a web page and then clicking the option of adding a contact or an event. The data that was selected on a web page was parsed and converted to industry standard format such as vCARD in case of contacts and ICS in case of event information. The user is prompted to open or save this file using the existing file save dialog of Firefox. The user can associate such files in Firefox to be opened with an installed mail client such as Outlook or Thunderbird. The user then has the ability to edit the data, make changes before saving it. A vCARD format is saved as a contact and an VCALENDAR format is added as an event to the user's calendar.

Users looking at webpages with tabular display of address data can use our extension's "Show Address" feature to view all the addresses on a given web page on a mapping application such as Google Maps. For example users viewing home listings on a web page would prefer to see them on a map to get a sense of location. Also having the ability to add new landmarks with the existing data allows the user to mash the address data on a mapping application such as Google Maps.

1.3 Report Overview

The report is divided into the following sections: Chapter 2 discusses various technologies used in the implementation of this project. Chapter 3 goes into detail about each component of the Firefox extension. It goes into the details of the implementation. Chapter 4 has the conclusion. Finally chapter 5 provides information on how this project can be extended to other areas.

2 Technology and Standards Used

2.1 Technology

Firefox has a robust architecture for creating extensions. XUL, Javascript and CSS were some of the technologies that were used for building an extension. Additionally J2EE technology was used to show the addresses on Google Maps.

2.1.1 XUL

Firefox uses XUL as the UI definition language to specify the various UI widgets and their placement. This is called an Overlay [Mozilla Developer Center XUL Overlays]. Overlays are used to either override small snippets of the UI without having to write the entire UI. XUL has a definition to draw various widgets seen in the Firefox browser. Each UI widget can be provided an “id” attribute and can be controlled using Javascript.

2.1.2 Javascript

Javascript was used to peform all the event handling functions. Once the option to either “Add Contact”, “Add Event” or “Show Address” was selected the event handling code written in Javascript was triggered to perform the necessary function.

2.1.3 XPCOM

XPCOM is a standard cross-platform object model provided by Mozilla that exposes a core set of components and interfaces to perform File I/O and other Content Handling Services. XPCOM makes it easy to access most aspects of the browser functionality programmatvCalendarly. Our extension uses XPCOM extensively to perform File I/O

and access the content handler window to download a .vcf or a .ics file format in a chrome environment.

2.1.4 CSS

CSS was used to style the context elements to have icon images in the context click of the browser. Icon images were provided by [Mark James].

2.1.5 JSP

Java Server Pages was used to receive the request parameters and query Google Maps API for geo-coding the addresses that was parsed from the web page. Once the addresses were geo-coded they were plotted on Google Maps using the Google Maps API[Google Map APIs].

2.1.6 Google Maps API

The Google Maps API was used extensively to geo-code addresses and plot them on Google Maps. Addresses that were geo-coded were shown on Google Maps using the Google Maps API [Google Map APIs].

2.2 Standards

Two industry standard file formats were used to capture data. Contact data was converted to the vCARD format for personal data interchange [vCard]. Event data was converted to ICS format to be added as a calendar event in the user's mail client [vCalendar].

2.2.1 vCARD Format

The vCARD format is an industry standard format for Personal Data Interchange. It is used to exchange "Person" objects[vCard]. This format begins with a "BEGIN:VCARD " string and ends with the "END:VCARD" string. A vCard has individual attributes that are known as Property. According to the [vCard], "A property takes the following format: PropertyName [';' PropertyParameters] ':' PropertyValue" (p. 5).

The property names and values are case insensitive. The property Name "N" indicates the name with the format being Last Name followed by First Name. The property "Title" has the title of the person. The telephone number can be specified using Property "TEL". A work number is distinguished from a home number by specifying the "Telephone Type". There are several other options for specifying a Fax or Cell number. An email address can be specified by the property name "EMAIL". The following example shows a vCARD format for a person named John Doe

```
BEGIN:VCARD
VERSION:2.1
N:Doe;John
FN:John Doe
TITLE:Sr. Director
TEL;WORK;VOICE:+1-111-111-1111
TEL;FAX:+1-222-222-2222
ADR;WORK:;;123 Some Way CA 94086
EMAIL;PREF;INTERNET:john.doe@nobody.com
END:VCARD
```

Listing 1 – Sample of a vCARD format

2.2.2 vCALENDAR Format

The vCalendar format is an industry standard format for defining calendar and scheduling information. A vCalendar format begins with a "BEGIN:VCALENDAR" string and ends

with a “END:VCALENDAR”. This element can contain other entities for calendaring and scheduling entities. A vEVENT object can be present in a VCALENDAR object and indicates a scheduled period of time for either an event or a meeting. An event format begins with a “BEGIN:VEVENT” string and ends with a “END:VEVENT” string. According to the [vCalendar], “A property takes the following form: PropertyName [‘;‘ PropertyParameters] ’:‘ PropertyValue” (p. 16). The date and time format can be specified in UTC timezone or if no timezone is provided the user’s timezone is assumed. According to [vCalendar] “The format for the complete, basic representation of a date and time value is written in the following sequence of characters: <year><month><day>T<hour><minute<second><type designator>” (p. 18). A date and time property that indicates a start of the event is “DTSTART”. A recurring rule that defines details of a recurring event is defined by the property “RRULE”. A “FREQ” property parameter defines the frequency of the event with values of “SECONDLY”, “MINUTELY”,”HOURLY”, “DAILY”, “WEEKLY”, “MONTHLY” and “YEARLY”. The “INTERVAL” rule part indicates the recurrence of the frequency and is indicated by a digit. For e.g. RRULE:FREQ=WEEKLY;INTERVAL=1 indicates a weekly rule occurring once every week. The “BYDAY” rule specifies the day of recurrence with valid values being “MO”, “TU”, “WE”,”TH”,”FR”. The “UNTIL” rule defines the end when the event ends. The other options for recurrence are “BYSECOND”, “BYMINUTE” and “BYHOUR”

3 Implementation

3.1 Setting up the framework

A standard Firefox extension requires a correct directory structure to be set up [Jonah Bishop]. The following figure shows the snapshot of our directory structure

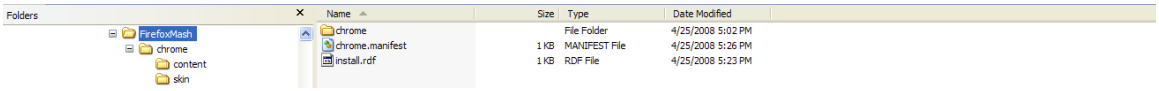

Figure 1– Directory structure for a Firefox extension

The install.rdf file gives details such as unique id to the extension as well as a description and other details to Firefox. The details of the install.rdf are shown below:

```
<?xml version="1.0"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:em="http://www.mozilla.org/2004/em-rdf#">
    <Description about="urn:mozilla:install-manifest">
        <em:id>firefoxmash@sjsu</em:id>
        <em:name>CS 298</em:name>
        <em:version>1.0</em:version>
        <em:targetApplication>
            <Description>
                <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
                <em:minVersion>1.5</em:minVersion>
                <em:maxVersion>2.0.0.*</em:maxVersion>
            </Description>
        </em:targetApplication>
        <!-- Optional Items -->
        <em:creator>Vijay Rao</em:creator>
        <em:description>An extension for CS298.</em:description>
<em:homepageURL>http://www.cs.sjsu.edu/faculty/pollett/masters/Semesters/Fall07/vijay/</em:homepageURL>
    </Description>
</RDF>
```

Listing 2 – install.rdf file

The chrome.manifest file provides details about the extension's package structure and which overlay's are being modified. Overlays in Firefox are used to provide additional functionality to the browser UI. New widgets could be placed and various places in the

UI using XUL overlays. The manifest file is shown below describes the content for this extension in a package called firefoxmash. It specifies an overlay over the browser's UI and provides a package for applying a skin to the extension for images and css.

```
content firefoxmash chrome/content/
overlay chrome://browser/content/browser.xul
chrome://firefoxmash/content/firefoxmash.xul
skin firefoxmash classic/1.0 chrome/skin/
```

Listing 3 – chrome.manifest file

The next step involved creating the XUL file. The XUL file described the UI elements including includes for various UI components including entries for the context menus. All events and actions were handled in javascript files.

3.2 Adding Contacts

The first phase of the project involved the user selecting the name and address portion of a web page and right clicking on it to add a contact to the webpage. The "Add Contact" menu was added to the context click of the user.

A depiction is shown below:

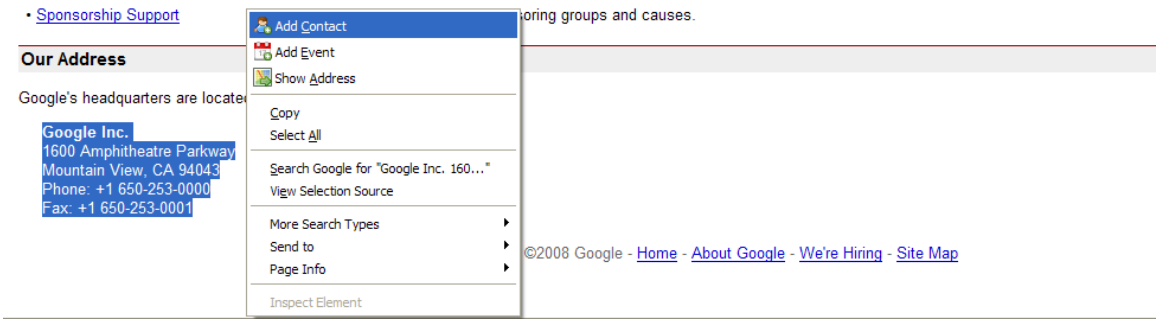

Figure 2 – Add a contact depiction

The implementation involved writing Javascript to capture the user's data that was selected on the web page. Once the data was captured, it had to be parsed to identify the various elements such as Name, Phone Number and Email Address.

3.2.1 Implementation

The first step involved adding an entry in the *firefoxmash.xul* file. The entry is shown below

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://firefoxmash/skin/foxmash.css"
type="text/css"?>
<overlay id="FirefoxMash-Overlay"

xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
    <script type="application/x-javascript"
            src="chrome://firefoxmash/content/DOMHelper.js" />
    <script type="application/x-javascript"
            src="chrome://firefoxmash/content/CleanDate.js" />
    <script type="application/x-javascript"
            src="chrome://firefoxmash/content/firefoxmash.js" />
    <!-- This is for the right click menu. -->
    <popup id="contentAreaContextMenu">
        <menuseparator id="firefoxmashabove"
insertafter="context-stop"/>
        <menuitem id="contact"
                  label="Add Contact"
                  class="menuitem-iconic"
                  accesskey="C"
                  insertafter="context-stop"
                  oncommand="addContact();"/>
    </popup>
</overlay>
```

Listing 4 – Adding an entry in the XUL file

The entry involved defining a menuitem element to create an option to add a contact. The class was set to “menuitem-iconic” to indicate a png image beside it. Icons were provided by Mark James [Mark James]. The next step involved collecting the text that was selected by the user. If no text is selected the code returns and nothing happens. The next step involves applying a regular expression to identify a STATE ZIP pattern to the text. The regular expression is shown below:

```
var statezipre =
/\b(AL|AK|AS|AZ|AR|CA|CO|CT|DC|DE|FM|FL|GA|GU|HI|ID|IL|IN|IA|KS|KY|LA|M
E|MH|MD|MA|MI|MN|MS|MO|MT|NE|NV|NH|NJ|NM|NY|NC|ND|MP|OH|OK|OR|PW|PA|PR|
RI|SC|SD|TN|TX|VI|UT|VT|VA|WA|WV|WI|WY|AA|AE|AP)\b \b[0-9]{5}(-[0-
```

```
9]{4})?\b/;
```

Listing 5 – Regular Expression to identify a STATE ZIP pattern

The pattern above can be broken down as follows

\b	Indicates a word boundary
(AL|AK|AS|AZ|AR|CA|CO| CT|DC|DE|FM|FL|GA|GU|HI| ID|IL|IN|IA|KS|KY|LA|ME|MH| MD|MA|MI|MN|MS|MO|MT|NE|NV| NH|NJ|NM|NY|NC|ND|MP|OH|OK| OR|PW|PA|PR|RI|SC|SD|TN|TX| VI|UT|VT|VA|WA|WV|WI|WY|AA|AE|AP)	Looks for a list of all US States in the two letter format
[0-9]{5}(-[0-9]{4})?	Looks for 5 numbers from 0 to 9 and an optional hyphen character and 4 numbers signifying a US zip code.

Table 1 – Breakdown for STATE ZIP Regular Expression

Once a match is found the text is split based on characters before the state zip and characters after this pattern match. Based on this an intelligent deduction is performed to look for phone numbers, email address. The regular expression pattern for email address is shown below

```
var emailre = /\b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}\b/;
```

Listing 6 – Regular Expression for matching an email address

This expression was published at [Jan Goyaverts] and was modified by us to identify lowercase characters as well. This regular expression is broken down below

\b	Indicates a word boundary
[a-zA-Z0-9._%+-]+	This matches any lower case or upper case characters as well as digits from 0 to 9 including the ., _ % + - characters. The + sign indicates one or more occurrences of these characters
@	This matches the @ character
[a-zA-Z0-9.-]+	This matches any lower case or upper case characters as well as digits from 0 to 9 including the ., and - characters. The + sign indicates one or more occurrences of these

	characters
`\.`	This matches the . character
`[a-zA-Z]{2,4}`	This matches the domain of the email address which can be of 2 to 4 characters in length

Table 2—Regular Expression breakdown for email address

The regular expression to identify a phone number is shown below

```
var telre = /\b\(?\d{3}\)?[-\s.]?\d{3}[-\s.]?\d{4}\b/;
```

Listing 7 – Regular Expression for US Telephone number

This regular expression is broken down below

\b	Indicates a word boundary
`\(?`	Indicates an optional bracket to identify area code
`\d{3}`	Indicates a digit ie character 0 to 9 and matches 3 digits
`\)?`	Indicates an optional closing paranthesis
`[-\s.]?`	Indicates an optional area code separator of either hyphen, space or a dot character
`\d{3}[-\s.]?`	Indicates another 3 digits followed by a separator of either a hyphen, space or a dot character
`\d{4}`	Indicates 4 digits

Table 3 – Regular Expression breakdown for US Telephone number

Once the data has been parsed using the regular expression described above a .vcf file is generated with the user selected data and the user is prompted to open or save the file in Firefox. Some of the challenges involved identifying a pattern to the data selected and making an intelligent deduction about the location of various elements such as phone email and telephone number. Several formats of contact information was tested to arrive at an accurate parsing probability. At this point XPCOM [Mozilla Developer Center XPCOM] was used to launch the browser's default File Save Dialog to ask the user to either launch the file with his preferred application or save it to disk. A snapshot is shown below:

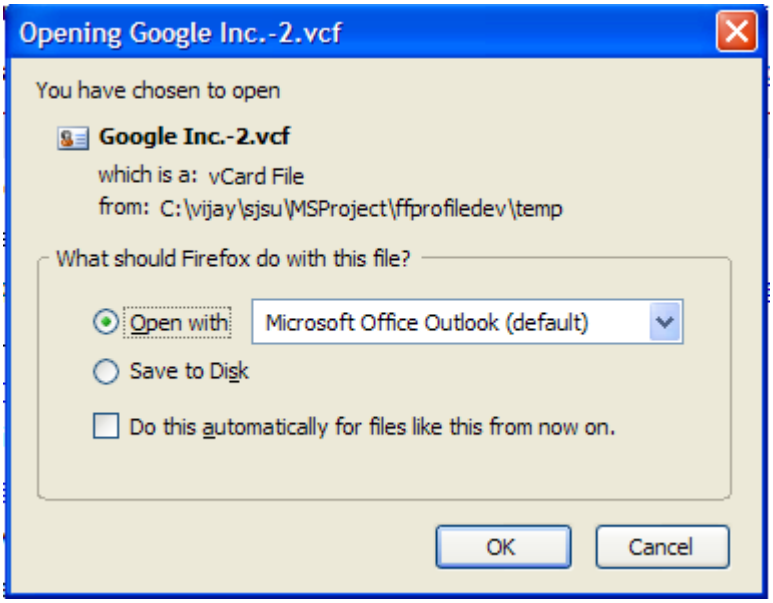

Figure 3 – Snapshot of the File Save Open With dialog in Firefox

The user can now choose to open this in the preferred mail client edit it, make changes to it and then save the contact. A snapshot of the screen in Microsoft Outlook is shown below

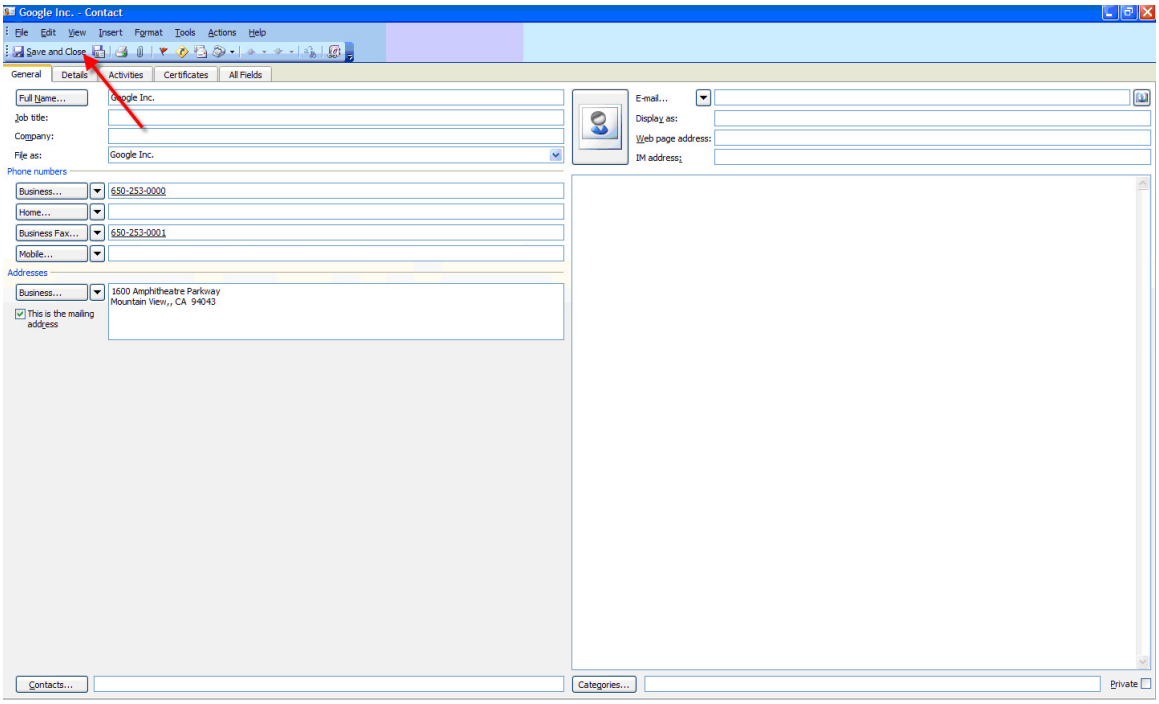

Figure 4 – Snapshot of the contact in Outlook

If the contact has no name defined as shown below then a string “unknown” is used in place of the name as shown below

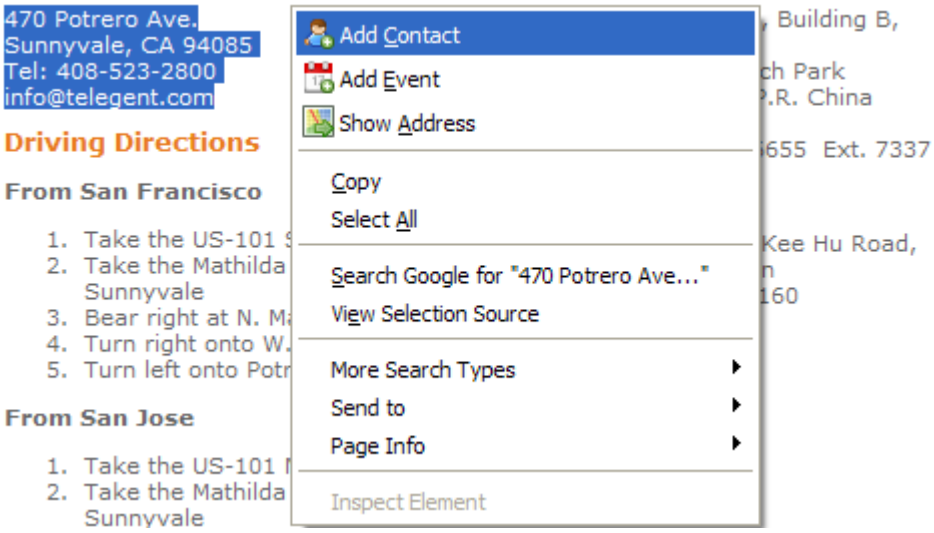

Figure 5 – User capturing portion of data without name

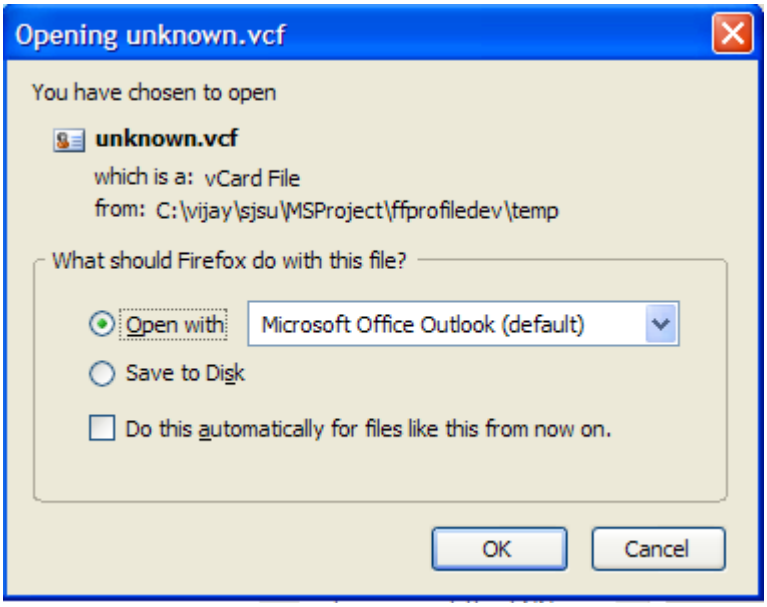

Figure 6 – Snapshot showing the download as unknown.vcf

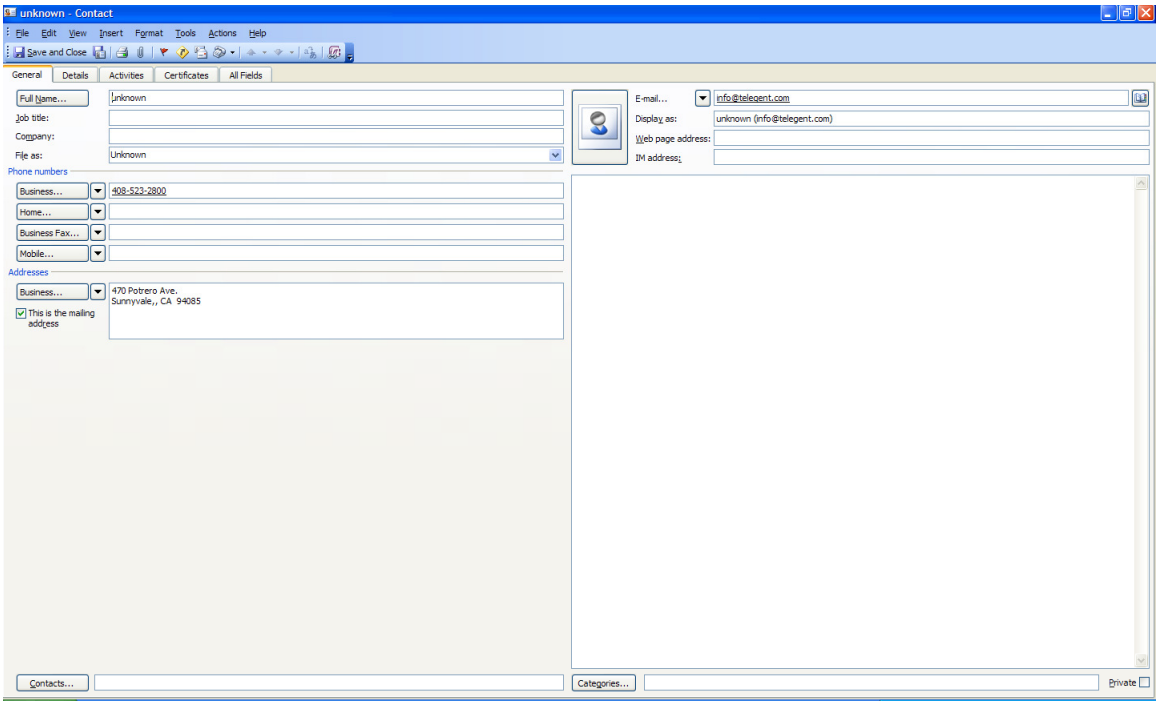

Figure 7 – Snapshot showing the name as being unknown

The File I/O in Firefox was done by using the api provided by [Captain's Mozilla XUL LOG]. The temporary files are written to a temp directory in the user's profile folder.

3.3 Add Events

The second deliverable of this project involved the user selecting text on a web page by right clicking on it to add an event to the webpage. The "Add Event" menu was added to the context click of the user. A depiction is shown below

Figure 8 – Add an event depiction

To implement this feature a Javascript function was written to capture the data that was selected on the web page. Once the data was captured, it had to be parsed to identify the "What", "When" and "Where" aspect of the event.

3.3.1 Implementation

The first step involved adding an entry in the *firefoxmash.xul* file. The enty is shown below

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://firefoxmash/skin/foxmash.css"
type="text/css"?>
<overlay id="FirefoxMash-Overlay"

xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
    <script type="application/x-javascript"
            src="chrome://firefoxmash/content/DOMHelper.js" />
    <script type="application/x-javascript"
            src="chrome://firefoxmash/content/CleanDate.js" />
    <script type="application/x-javascript"
            src="chrome://firefoxmash/content/firefoxmash.js" />
    <!-- This is for the right click menu. -->
    <popup id="contentAreaContextMenu">
        <menuseparator id="firefoxmashabove"
insertafter="context-stop"/>
        <menuitem id="event"
                  label="Add Event"
```

```
                    class="menuitem-iconic"
                    accesskey="E"
                    insertafter="context-stop"
                    oncommand="addEvent();"/>
    </popup>
</overlay>
```

Listing 8 – Adding an entry in the XUL file

The entry involved defining a menuitem element to create an option to add an event. The class was set to "menuitem-iconic" to indicate an icon style list element. Icons were provided by Mark James [Mark James]. The selected text was now parsed to separate the What, When and Where aspect of the event. The "What" aspect was to analyze what the event was about, the "When" aspect was to parse and find when the event was scheduled and the "Where" was to identify where the event was scheduled. To do this several regular expressions were used. The following regular expression was used to identify various date formats. The regular expression is shown below was modified from [Jan Goyaverts]

```
var whenre = ((0?[1-9])|(1[012]))[/](0?[1-9]|[12][0-
9]|3[01])[/](19|20)?\d\d\s\d\d?(:\d\d)?\s(AM|PM)(\s(to|TO)\s(((0?[1-
9])|(1[012]))[/](0?[1-9]|[12][0-
9]|3[01])[/](19|20)?\d\d\s)?\d\d?(:\d\d)?\s(AM|PM))?
```

Listing 9 – Regular Expression to identify the "When" aspect in selected text

The pattern above can be broken down as follows

`((0?[1-9])|(1[012]))[/](0?[1-9]|[12][0-9]|3[01])[/](19|20)?\d\d`	Looks for a valid date format. This format was reused from [Jan Goyaverts]
`\d\d?(:\d\d)?\s(AM|PM)`	Matches a time format
`(\s(to|TO)\s`	Matches a space and character "to"

Table 4 – Breakdown for Date and Timestamp Regular Expression

The matched category is used to populate the "When" part of the event. The "Where" part of the event is deduced by applying the regular expression to identify a US address.

The regular expression pattern for US street address is shown in Figure 25. Once the data has been parsed using the regular expressions described above a .ics file is generated with the user selected data and the user is prompted to open or save the file in Firefox. At this point XPCOM [Mozilla Developer Center XPCOM] was used to launch the browser's default File Save Dialog to ask the user to either launch the file with his preferred application or save it to disk. A snapshot is shown below:

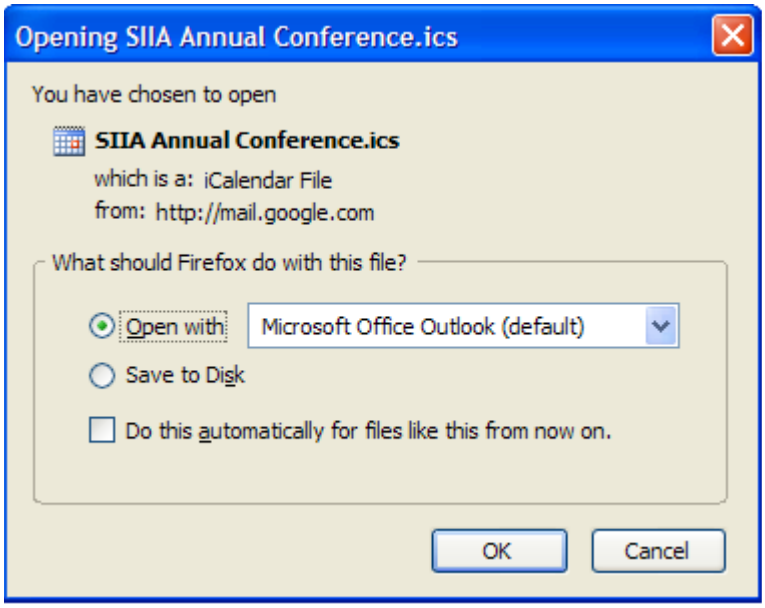

Figure 9 – Snapshot of File Save/ Open With dialog in Firefox

The user can now choose to open this in the preferred mail client edit it, make changes to it and then save the contact. The generation of the ics file was done using the api provided by [Captain's Mozilla XUL LOG]. The temporary files are written to a temp directory in the user's profile folder. The image below shows the data being parsed and displayed in the right fields.

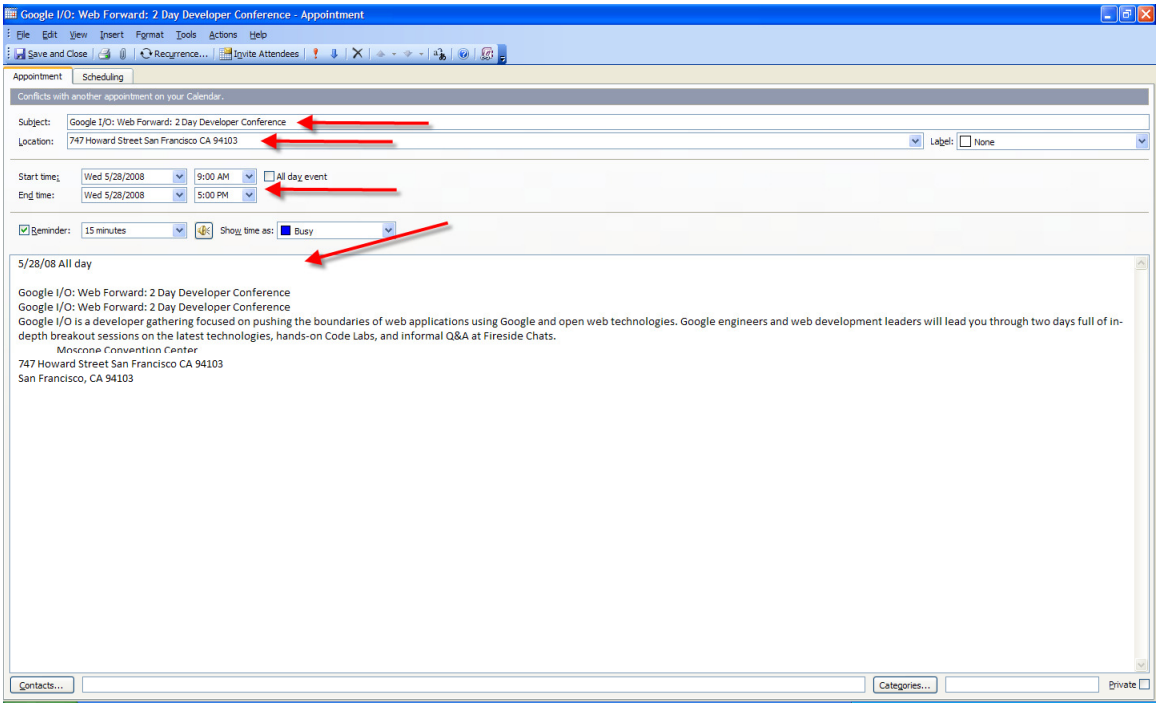

Figure 10 – Snapshot of the ics file opened in Microsoft Outlook

3.4 Show Addresses

A user of a web browser to view web pages often comes across tabular display of data wit addresses in it. Our extension tries to extend the browser to provide the ability of the user to view data in a mapping application such as Google Maps. Often data in context of location provides to be a lot more intuitive. Some examples of such scenarios are house listings. A user scouring the web looking at listings of homes may want to drop all the addresses on a map to see houses that have a proximity to a certain landmark such as a School or a Church. Another example would be where a user viewing the Megan's law database for a listing of sex offenders in the vicinity would prefer to see the addresses on an intuitive mapping application such as Google Maps and in context of another landmark such as the user's home. Our extension satisfies this need by parsing the web page to identify street addresses and plots these addresses on Google Maps. Our

extension also allows the user to add additional landmarks to the page so that the user can view the data with a perspective of the landmarks that are added on an as need basis.

3.4.1 Implementation

The first step involved adding an entry in the *firefoxmash.xul* file. The entry is shown below:

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://firefoxmash/skin/foxmash.css"
type="text/css"?>
<overlay id="FirefoxMash-Overlay"

xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
    <script type="application/x-javascript"
            src="chrome://firefoxmash/content/DOMHelper.js" />
    <script type="application/x-javascript"
            src="chrome://firefoxmash/content/CleanDate.js" />
    <script type="application/x-javascript"
            src="chrome://firefoxmash/content/firefoxmash.js" />
    <!-- This is for the right click menu. -->
    <popup id="contentAreaContextMenu">
        <menuseparator id="firefoxmashabove"
insertafter="context-stop"/>
              <menuitem id="showOnMap"
                  class="menuitem-iconic"
                  label="Show Address"
                  accesskey="A"
                  insertafter="context-stop"
                  oncommand="showAddress();"/>
    </popup>
</overlay>
```

Listing 10 – Adding an entry in XUL file

The entry involved defining a menuitem element in XUL to create an option to add a contact. The class was set to “menuitem-iconic” to indicate it as a list item with an icon beside it. The next step involved parsing the html on the webpage. The HTML DOM was queried to retrieve all the “th” elements from the page. The utility DOMHelper.js was created using all the methods provided by John Resig in his book [John Resig]. These methods are convenience methods for traversing a DOM structure. Regular Expressions

were used to identify Address or Street in the headers as placeholders for Addresses. A depiction of the extension for this functionality is shown below:

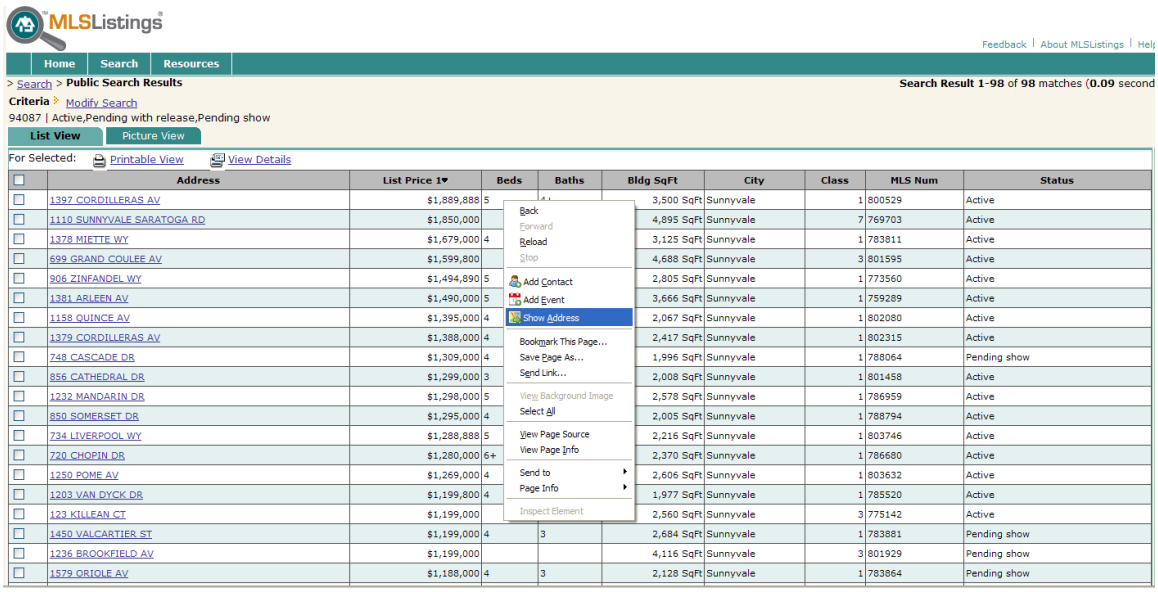

Figure 11– Show Address depiction

A regular expression to identify US Street addresses was used to identify addresses on the current web page.

The regular expression is shown below:

```
var stre =
/\d+\s[\w\s.]+\b(AV|AVE|AVENUE|WY|WAY|TE|DR|DRIVE|CT|COURT|RD|ST|LN|TL|
EL CAMINO REAL)\b/i
```

Listing 11 -- Regular Expression to identify US Street Address

The pattern can be broken down as follows

`\d+`	Indicates a digit from 0 to 9 occurring one or more times
`\s`	Indicates a space
`[\w\s.]+`	Indicates one or more occurrences of words and or spaces
`\b`	Indicates a word boundary
`(AV|AVE|AVENUE|WY|WAY|TE| DR|DRIVE|CT|COURT|RD|ST|LN|TL|EL CAMINO REAL)`	Indicates one or more options to identify an address
`I`	Indicates case-insensitive search

Table 5 – Regular Expression breakdown for US Street Address

The data that was collected after applying the regular expression above was sent to a web application. This was done by appending the data as a query string. Once this was received the web application queried Google's geo-coding service to geo-code these addresses [Google Map APIs]. Once the addresses were geo-coded they were plotted on Google Maps with a listing of addresses to the left. A depiction of this is shown below

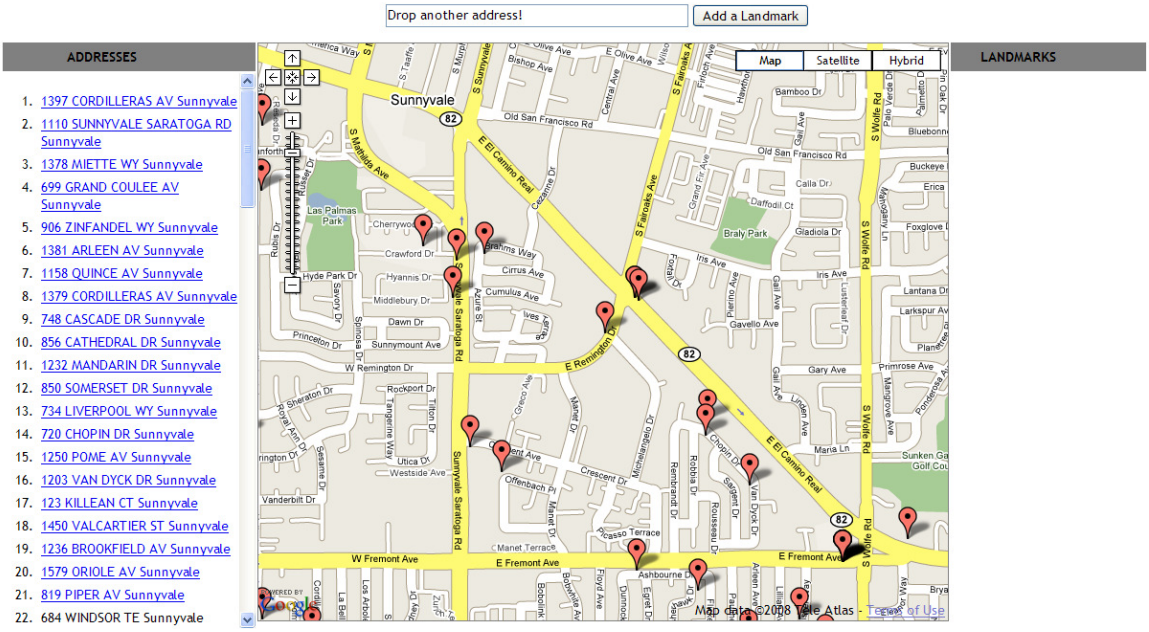

Figure 12 – Snapshot of all the addresses plotted on Google Maps

Addresses that were geo-coded by Google are plotted as links. The results are enclosed in a scrollable div allowing the user to scroll through the list. Clicking on any link pans the map to that address as shown below:

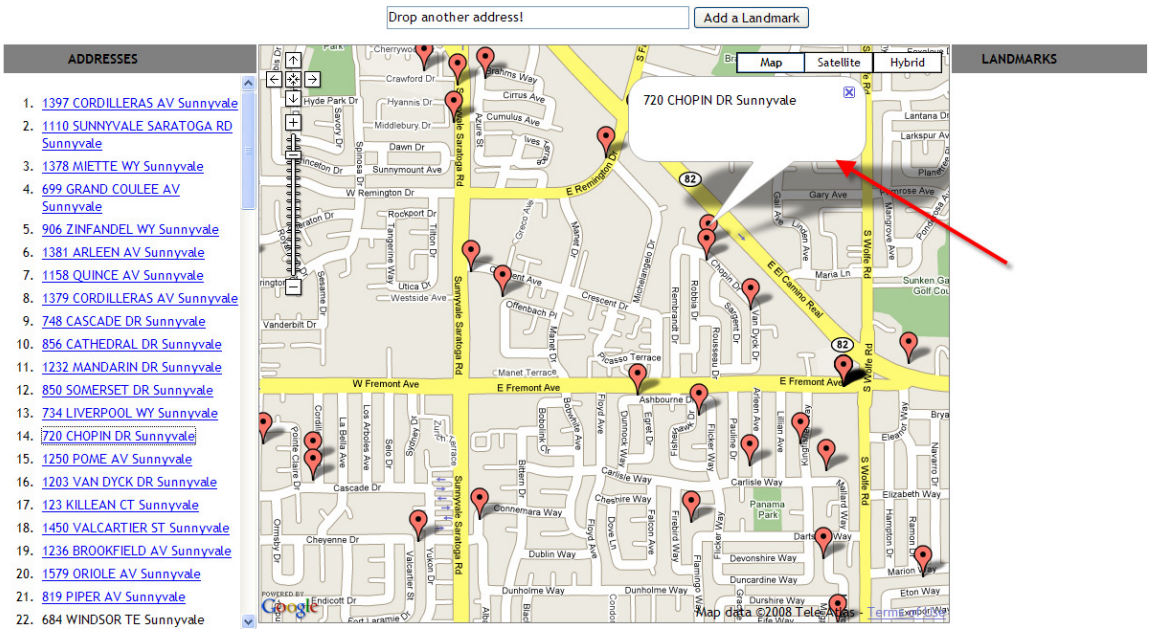

Figure 13 – Snapshot of the map panning to show the address link that was clicked

Addresses that were not geo-coded by Google are listed but not highlighted as links as shown below

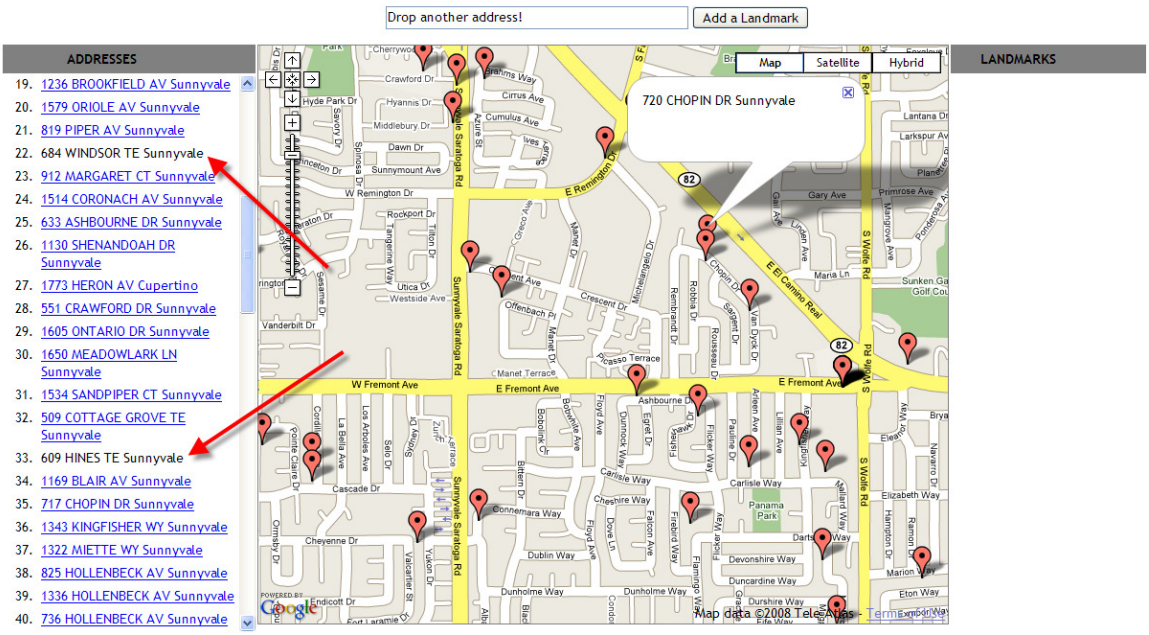

Figure 14 – Snapshot showing addresses that were not geo-coded by Google

The user can also add additional landmarks on the map by entering an address in the text field on the top and pressing the “Add a Landmark” button.

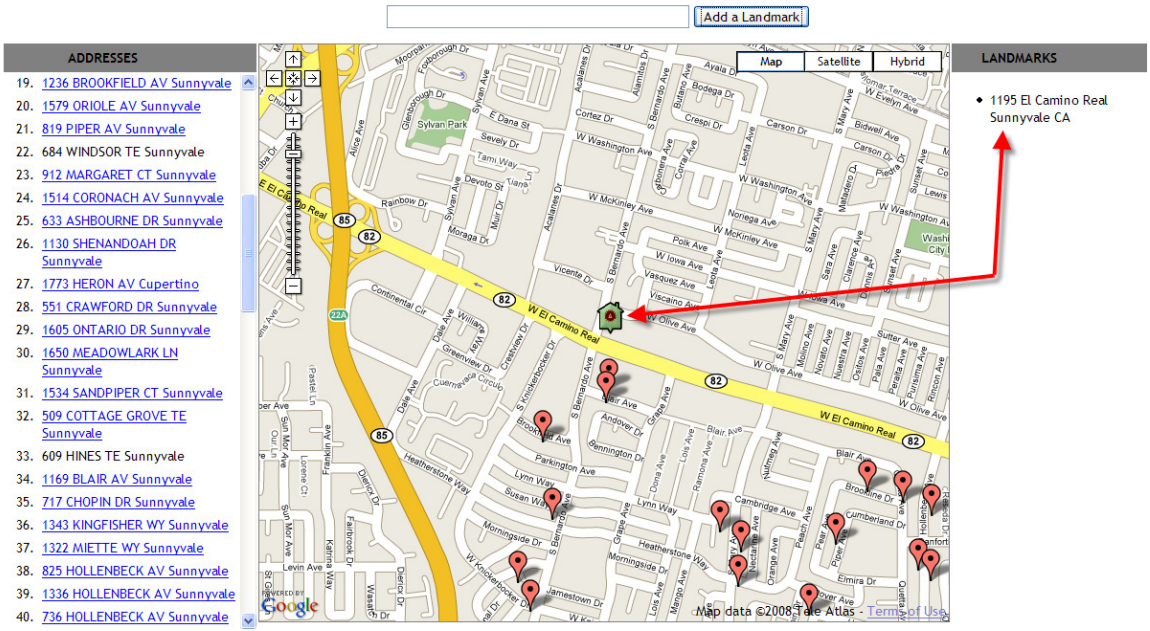

Figure 15 – Snapshot after adding a landmark

As soon as the user adds a landmark a special icon is dropped on the map indicating a landmark and the address entered by the user is listed on the right. If a landmark entered by the user cannot be geo-coded by Google a message is displayed to the user as shown below

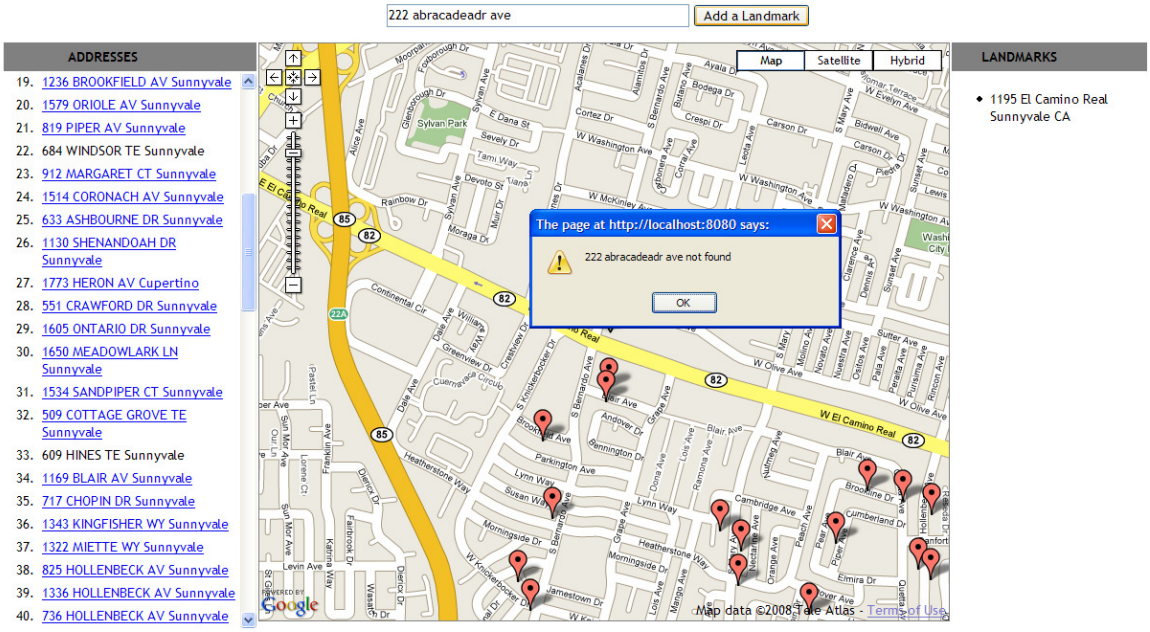

Figure 16 – Snapshot of an address that was not geo-coded by Google

There were several challenges involved in developing this. Parsing content on websites involved overcoming the issues related to non-standard markup. The fact that street address by itself could not be geo-coded from Google involved figuring out the City and passing it along with the address that needed to be geo-coded. The regular expression had to be expanded to capture majority of combinations of street endings.

4 Conclusion

In this writing project our goals were to extend the browser such that the user can save snippets of data from web pages. We identified adding of contacts and events to the mail client as a task that was routinely performed by the user. It was obvious that both these activities involved manual entry of forms to save the contacts phone, address, email address etc. An automated parsing of the information to retrieve the relevant data seemed like the best way to attain the objective.

Our extension extended the browser to capture information selected by the user and intelligently parsing the information and converting it to a standard data exchange format to be saved to the mail client.Another goal established at the beginning of the project was to provide the user the ability to mashup the addresses listed on a web page by displaying the addresses on a mapping application and allow the user to view the data in context of other landmarks. Our extension exceeded this goal providing the user the ability to view the addresses on Google Maps application with the flexibility of adding more landmarks.

This project had several challenges in terms of parsing of web pages and poor documentation on Firefox extension. The challenges involved identifying various patterns of data and being flexible in extracting relevant data from various formats. Other challenges involved logistical difficulties such as poor documentation on specific modules such as XPCOM.

5 Future Extension

The extension relies on a web application component. This could be hosted at a common location which could then allow the user to log in and view the address data. This could be enhanced to allow the user to save this data so that it can be viewed at a later time. Additionally saving the URL of the source or the list of parameters that produced the results could enhance the user experience by automatvCalendarly refreshing the results. The extension could be enhanced to accept newer address patters thus enabling the extension to learn

6 Bibliography

Jonah Bishop Firefox Toolbar Tutorial Retrieved May 01, 2008 from http://www.borngeek.com/firefox/toolbar-tutorial/

Mark James Retrieved May 01, 2008 from http://www.famfamfam.com/lab/icons/silk/

Jan Goyaverts (28th Aug 2007) Regular Expression Tutorial Retrieved May 01, 2008 from http://www.regular-expressions.info/tutorial.html

vCard Internet Mail Consortium Retrieved May 01, 2008 from http://www.imc.org/pdi/vcard-21.txt

vCalendar Internet Mail Consortium Retrieved May 01, 2008 from http://www.imc.org/pdi/vcal-10.txt

Captain's Mozilla XUL LOG Firefox Toolbar tutorial Retrieved May 01, 2008 from http://www.captain.at/programming/xul/

Google Map APIs Retrieved May 01, 2008 from http://code.google.com/apis/maps/

John Resig (2006) Pro Javascript Techniques Apress

Mozilla Developer Center XUL Overlays Retrieved May 01, 2008 http://developer.mozilla.org/en/docs/XUL_Overlays

Mozilla Developer Center XPCOM Retrieved May 01, 2008 http://developer.mozilla.org/en/docs/XPCOM

Appendix

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://firefoxmash/skin/foxmash.css" type="text/css"?>

<overlay id="FirefoxMash-Overlay"
         xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

    <script type="application/x-javascript"
            src="chrome://firefoxmash/content/DOMHelper.js" />
    <script type="application/x-javascript"
            src="chrome://firefoxmash/content/CleanDate.js" />
    <script type="application/x-javascript"
            src="chrome://firefoxmash/content/dateFormat.js" />
    <script type="application/x-javascript"
            src="chrome://firefoxmash/content/firefoxmash.js" />

    <!-- This is for the right click menu. -->

    <popup id="contentAreaContextMenu">


        <menuitem id="showOnMap"
                  class="menuitem-iconic"
                  label="Show Address"
                  accesskey="A"
                  insertafter="context-stop"
                  oncommand="showAddress();"/>

        <menuitem id="event"
                  label="Add Event"
                  class="menuitem-iconic"
                  accesskey="E"
                  insertafter="context-stop"
                  oncommand="addEvent();"/>

        <menuitem id="contact"
                  label="Add Contact"
                  class="menuitem-iconic"
                  accesskey="C"
                  insertafter="context-stop"
                  oncommand="addContact();"/>

        <menuseparator id="firefoxmashabove" insertafter="context-stop"/>

    </popup>

</overlay>
```

Listing 12 – firefoxmash.xul

```
String.prototype.trim = function() {   return this.replace(/^\s+|\s+$/g,"");}

function read(mime,filename) {
    try {
        netscape.security.PrivilegeManager.enablePrivilege("UniversalXPConnect");
    } catch (e) {
        alert("Permission to read file was denied.");
    }
    var file = Components.classes["@mozilla.org/file/local;1"]
        .createInstance(Components.interfaces.nsILocalFile);
    LOG(" IN READ FILENAME " + filename);
    file.initWithPath(filename);
    if ( file.exists() == false ) {
        alert("File does not exist");
    }
```

```
    var iosvc = Components.classes["@mozilla.org/network/io-service;1"].
            getService(Components.interfaces.nsIIOService);
    var chnl = iosvc.newChannelFromURI(iosvc.newFileURI(file));
    var helper = Components.classes["@mozilla.org/uriloader/external-helper-app-
service;1"].
            getService(Components.interfaces.nsIExternalHelperAppService);
    var stream = helper.doContent(mime,chnl,this);
    chnl.asyncOpen(stream,this);

}

function getSelectedText()
{
    if( window.content.getSelection) {
        return window.content.getSelection().toString();
    }
}

function addContact() {

    var txt = getSelectedText();
    if(txt == null || txt.length == 0) return;

    var statezipre =
/\b(AL|AK|AS|AZ|AR|CA|CO|CT|DC|DE|FM|FL|GA|GU|HI|ID|IL|IN|IA|KS|KY|LA|ME|MH|MD|MA|MI|MN|M
S|MO|MT|NE|NV|NH|NJ|NM|NY|NC|ND|MP|OH|OK|OR|PW|PA|PR|RI|SC|SD|TN|TX|VI|UT|VT|VA|WA|WV|WI|
WY|AA|AE|AP)\b \b[0-9]{5}(-[0-9]{4})?\b/;
    var myarr = txt.split(/\r\n/);

    for(m = 0 ; m < myarr.length; m++) {
        if(myarr[m].length == 0)
        {
            myarr.splice(m,1);
        }
    }
    var obj = {};
    if(myarr.length > 1)  {
        for(i = 0; i < myarr.length; i++) {
            var arr = statezipre.exec(myarr[i]);
            if(arr) {
                obj = collectData(myarr, arr[0], i, myarr.length);
                break;
            }
        }
    } else {
        //data is in one line. apply state zip pattern to check
        var arr = myre.exec(txt);
        if(arr) {
            for(j=0; j < arr.length; j++) {
                alert("j is " + arr[j]);
            }
        }

    }
    var fileData = buildContact(obj);
    send('text/x-vcard',obj.filename,fileData);
}

function addEvent() {
    var dataobj={};
    dataobj.what="";
    dataobj.when="";
    dataobj.where="";
    dataobj.summary="";
    var txt = getSelectedText();
    txt = txt.replace(/\r\n/,' ');
    dataobj.what = txt;
    var re1 = /((0?[1-9])|(1[012]))[/](0?[1-9]|[12][0-
9]|3[01])[/](19|20)?\d\d\s\d\d?(:\d\d)?\s(AM|PM)(\s(to|TO)\s(((0?[1-
9])|(1[012]))[/](0?[1-9]|[12][0-9]|3[01])[/](19|20)?\d\d\s)?\d\d?(:\d\d)?\s(AM|PM))?/mi
    var re2 = /((0?[1-9])|(1[012]))[/](0?[1-9]|[12][0-9]|3[01])[/](19|20)?\d\d\s((All
```

```
Day)|(\d\d?(:\d\d)?\s(AM|PM)?))?/mi
    var fullmthre =
/(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)\s\d\d?\s\d{4}\s\d\d?(:\d\d?)?(AM|PM)/m
i
    var arr = re1.exec(txt);
    var whenidx = -1;
    if(!arr) {
        arr = re2.exec(txt);
    }
    //LOG(txt);
    if(!arr) {
        arr = fullmthre.exec(txt);
    }
    //LOG("fullmthre is " + arr);
    if(arr)
        whenidx = arr.index;
    var summ = "";
    var splitlinesarr = txt.split(/\r\n/);
    if(whenidx == 0) {
        for(i = 1; i < splitlinesarr.length; i++) {
            summ = splitlinesarr[i];
            summ = splitlinesarr[i].trim();
            if(summ && summ.length > 0) break;
        }
    }
    dataobj.summary = summ;
    //LOG('whenidx ' + whenidx);
    var obj = parseDateString(arr[0]);
    dataobj.when = obj;
    dataobj.where = parseAddress(txt);
    //LOG(dataobj);
    generateCalendarEvent(dataobj);
}

function generateCalendarEvent(eventobj) {
    var mask = "yyyymmdd'T'HHMMss";
    obj = eventobj.when;
    LOG('IN generateCalendarEvent ' + obj.startdate + " - " + obj.enddate);
    var str = "BEGIN:VCALENDAR\n";
    str += "BEGIN:VEVENT\n";
    str += "DTSTART:"+obj.startdate.format(mask)+"\n";
    str += "DTEND:"+obj.enddate.format(mask)+"\n";
    str += "LOCATION:"+eventobj.where+"\n";
    str += "DESCRIPTION:" + eventobj.what.replace(/\r\n/g,'\\n')+"\n";
    str += "SUMMARY:"+eventobj.summary+"\n";
    str += "BEGIN:VALARM\n";
    str += "TRIGGER:-PT15M\n";
    str += "ACTION:DISPLAY\n";
    str += "DESCRIPTION:REMINDER\n";
    str += "END:VALARM\n";
    str += "END:VEVENT\n";
    str +="END:VCALENDAR\n";
    LOG(str);
    var filename = eventobj.summary.trim()+'.ics';
    //Get rid of special characters
    filename = filename.replace(/[:\\/]/g,'-');
    send('text/calendar',filename,str);
}

function parseAddress(str) {
    var stre =
/\d+\s[\w\s]+\b(AVENUE|AV|AVE|WY|WAY|TE|DRIVE|DR|COURT|CT|Boulevard|Blvd|Road|RD|Street|S
T|LANE|LN|TL|EL CAMINO REAL|Highway|HWY)\b/i
    var statezipre =
/\b(AL|AK|AS|AZ|AR|CA|CO|CT|DC|DE|FM|FL|GA|GU|HI|ID|IL|IN|IA|KS|KY|LA|ME|MH|MD|MA|MI|MN|M
S|MO|MT|NE|NV|NH|NJ|NM|NY|NC|ND|MP|OH|OK|OR|PW|PA|PR|RI|SC|SD|TN|TX|VI|UT|VT|VA|WA|WV|WI|
WY|AA|AE|AP)\b \b[0-9]{5}(-[0-9]{4})?\b/
    var addr = stre.exec(str);
    var address,street,streetcity, statezip = "";
    var streetidx,statezipidx = 0;
    if(addr) {
```

```
        street = addr[0];
        streetidx = addr.index;
    }
    var stateziparr = statezipre.exec(str);
    if(stateziparr) {
        statezip = stateziparr[0];
        statezipidx = stateziparr.index;
    }
    LOG('street ' + street);
    LOG('statezip ' + statezip);
    LOG ('Diff ' + str.substring(streetidx,statezipidx) );
    streetcity = str.substring(streetidx,statezipidx);
    streetcity = streetcity.replace(/\r\n/g,' ');
    //LOG('statezip.index ' + statezip.index);
    streetcity = streetcity.trim();
    address = streetcity + ' ' +statezip;
    LOG('address ' + address);
    return address;
}

function parseDateString(str) {
    LOG(' IN parseDateString ' + str);
    var simpledtfrmt = /\d?\d[/]\d?\d[/](19|20)?\d\d/
    var simpletmfrmt = /\d?\d(:\d\d)?\s?(AM|PM)/
    var ampmre = /(AM|PM)/i
    var obj = {};
    obj.startdate="";
    obj.enddate="";
    //The following formats are possible
    //mm?/dd?/yy(yy)?
    //mm?/dd?/yy(yy)?\s All Day
    //mm?/dd?/yy(yy)? \d\d?(:\d\d?)? (AM|PM)
    //mm?/dd?/yy(yy)? \d\d?(:\d\d?)? (AM|PM) to \d\d?(:\d\d?)? (AM|PM)
    //mm?/dd?/yy(yy)? \d\d?(:\d\d?)? (AM|PM) to mm?/dd?/yy(yy)? \d\d?(:\d\d?)?(AM|PM)
    if(str) {
        var frmprt = "";
        var toprt = "";
        var frmdt,frmtm,todt,totm = "";
        var toidx = str.indexOf("to");
        LOG('toidx ' + toidx);
        if(toidx != -1) {
            frmprt = str.substring(0,toidx);
            var dtarr = simpledtfrmt.exec(frmprt);
            if(dtarr[0])
                frmdt = CleanDate(dtarr[0]);
            var tmarr = simpletmfrmt.exec(frmprt);
            var splitarr = tmarr[0].split(' ');
            if(splitarr[0].indexOf(':') == -1)
                splitarr[0] += ":00";
            frmtm = splitarr.join(' ');
            LOG(frmdt + ' ' + frmtm);
            toprt = str.substring(toidx+2,str.length);
            var todtarr = simpledtfrmt.exec(toprt.trim());
            if(todtarr && todtarr.length > 0) {
                todt = CleanDate(todtarr[0]);
            } else {
                //to part does not have a date
                todt = frmdt;
            }
            var totmarr = simpletmfrmt.exec(toprt);
            var tosplitarr = totmarr[0].split(' ');
            if(tosplitarr[0].indexOf(':') == -1 && tosplitarr.length > 1) //format 7PM
would have length 1
                tosplitarr[0] += ":00";
            totm = tosplitarr.join(' ');
            obj.startdate = new Date(Date.parse(frmdt + ' ' + frmtm));
            obj.enddate = new Date(Date.parse(todt + ' ' +totm));
        } else {
            //HERE IF THERE IS NO "TO" DATE TIME. So Could be either date and time or
date and "All Day"
            //end to index
```

```
            var alldayidx = str.search(/All Day/i);
            if(alldayidx != -1) {
                var frmdtmarr = simpledtfrmt.exec(str);
                LOG('frmdtmarr ' + frmdtmarr);
                if(frmdtmarr) {
                    frmdt = CleanDate(frmdtmarr[0]);
                    todt = frmdt;
                    frmdt += ' 9:00 AM';
                    todt += ' 5:00 PM';
                    LOG('frmdt ' + frmdt);
                    obj.startdate = new Date(Date.parse(frmdt));
                    obj.enddate = new Date(Date.parse(todt));
                }
            } else {
                var frmdtmarr = simpledtfrmt.exec(str);
                LOG('frmdtmarr ' + frmdtmarr);
                if(frmdtmarr) {
                    frmdt = CleanDate(frmdtmarr[0]);
                    var tmarr = simpletmfrmt.exec(str);
                    var splitarr = tmarr[0].split(' ');
                    if(splitarr[0].indexOf(':') == -1)
                        splitarr[0] += ":00";
                    frmtm = splitarr.join(' ');
                    LOG(frmdt + ' ' + frmtm);
                    todt = frmdt;
                    frmdt += ' ' + frmtm;
                    todt += ' 5:00 PM';
                    LOG('frmdt ' + frmdt);
                    obj.startdate = new Date(Date.parse(frmdt));
                    obj.enddate = new Date(Date.parse(todt));
                }
            }
        }
        LOG('obj.startdate ' + obj.startdate);
        LOG('obj.enddate ' + obj.enddate);
    }
    return obj;
}

function showAddress() {

    var thtags = content.document.getElementsByTagName("th");
    var headers = [];
    var street_address_idx,city_address_idx = 0;

    for(i = 0; i < thtags.length; i++) {
        var thdata = text(thtags[i]);
        if(thdata.length > 0) {
            headers[i] = thdata;
        }
    }

    headers.reverse();

    var addre = /street|address/i;
    var cityre = /city/i;

    for(tt = 0; tt < headers.length; tt++) {
        if(headers[tt] && headers[tt].length > 0){
            headers[tt] = headers[tt].replace(/^\s+|\s+$/g, '');
            if(addre.exec(headers[tt]) != null) {
                street_address_idx =  tt;
            }
            if(cityre.exec(headers[tt]) != null) {
                city_address_idx =  tt ;
            }
            LOG("header tt length " + headers[tt].length);
        }
    }
    var trnode = parent(thtags[0]);//TODO if no headers exist fails here
```

```
    var stre =
/\d+\s[\w\s]+\b(AVENUE|AV|AVE|WY|WAY|TE|DRIVE|DR|COURT|CT|Boulevard|Blvd|Road|RD|Street|S
T|LANE|LN|TL|EL CAMINO REAL|Highway|HWY)\b/i
    var rw = 0;
    var data = new Array();
    while(next(trnode)) {
        trnode = next(trnode);
        var innerdata = new Array();
        var alltds = trnode.childNodes;
        var tdelems = [];
        var t = 0;
        for(g = 0; g < alltds.length; g++) {
            if(alltds[g].nodeType == 1) {
                tdelems[t] = alltds[g];
                t++;
            }
        }
        alltds = tdelems;
        alltds.reverse();
        var arr = stre.exec(text(trnode));
        var col = 0;
        if(arr && arr.length > 0) {
            for(k = 0; k < alltds.length; k++) {
                innerdata[col] = text(alltds[k]);
                col++;
            }
        }
        if(innerdata.length > 0) {
            data[rw] = innerdata;
        }
        rw++;
    }
    var querystr = "";
    var rcnt = 0;
    for(a = 0; a < data.length; a++) {
        var innerarr = data[a];
        var otherdata = "";
        var addrdata = "";
        if(innerarr) {
            var city = "";
            for(b = 0; b < innerarr.length; b++) {
                if(b == street_address_idx) {
                    var matches = stre.exec(innerarr[b]);
                    if(matches && matches.length > 0)
                        addrdata += headers[b] + "::"+ matches[0] + " " +city + "||";
                    else
                        otherdata += headers[b] + "::" + innerarr[b] + "||";
                } else if (b == city_address_idx) {
                    city = innerarr[b];
                } else {
                    if(headers[b] && headers[b].length > 0 && innerarr[b] &&
innerarr[b].length > 0) {
                        var matches = stre.exec(innerarr[b]);
                        if(matches && matches.length > 0) {
                            addrdata += matches[0];
                        }
                        otherdata += headers[b] + "::" + innerarr[b] + "||";
                    }
                }
            }
            otherdata = addrdata + otherdata;
            otherdata = otherdata.replace(/^\s+|\s+$/g, '');
            addrdata = addrdata.replace(/^\s+|\s+$/g, '');
            if(addrdata && addrdata.length > 0) {
                querystr += "&record"+rcnt+"="
                querystr += addrdata;
                rcnt++;
            }
        }
    }
    LOG( querystr);
```

```
    var url = "http://localhost:8080/test/index1.jsp?";
    url+= querystr;
    var win = window.open(url, "BOO", "");
}

function collectData(origData, matchedData, idx, total) {
    //if idx is 3 then we have a standard pattern
    //alert("idx is " + idx);
    //alert("origData is " + origData);
    //idx is the point where STATE ZIP exists
    var obj = {};
    obj.name = origData[0];
    obj.state = matchedData.substring(0,2);
    obj.zip = matchedData.substring(3);

    switch (idx) {
        case 1:
            obj.name = "unknown";
            obj.street1 = origData[0];
            obj.city =
origData[1].substring(0,origData[1].indexOf(matchedData.substring(0,2)));
            break;
        case 2:
            obj.street1 = origData[1];
            obj.city =
origData[2].substring(0,origData[2].indexOf(matchedData.substring(0,2)));
            break;
        case 3:
            if(origData[3].indexOf(matchedData) == 0) {
                obj.street1 = origData[1];
                obj.city = origData[2];
            } else {
                //this means city is before state
                obj.title = origData[1];
                obj.street1 = origData[2];
                obj.city =
origData[3].substring(0,origData[3].indexOf(matchedData.substring(0,2)));
            }
            break;
        case 4:
            if(origData[4].indexOf(matchedData) == 0) {
                obj.street1 = origData[2];
                obj.city = origData[3];
            } else {
                //this means city is before state
                obj.title = origData[1];
                obj.street1 = origData[2];
                obj.street2 = origData[3];
                obj.city =
origData[4].substring(0,origData[4].indexOf(matchedData.substring(0,2)));
            }
            break;
    }
    obj.filename = obj.name+".vcf";
    var emailre = /\b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}\b/;
    var telre = /\b\(?\d{3}\)?[-\s.]?\d{3}[-\s.]?\d{4}\b/;
    if(idx < total) {
        //look for phone or email
        for(i = idx+1; i < total; i++) {
            var data = origData[i];
            var emailarr = emailre.exec(data);
            if(emailarr && emailarr.length > 0){
                obj.email = emailarr[0];
            }
            var phonearr = telre.exec(data);
            if(phonearr && phonearr.length > 0) {
                if(obj.phone)
                    obj.fax = phonearr[0];
                else
                    obj.phone = phonearr[0];
            }
```

```
        }
    }
    return obj;
}

function send(mime,filename, data) {
    var file = Components.classes["@mozilla.org/file/directory_service;1"]
                         .getService(Components.interfaces.nsIProperties)
                         .get("ProfD", Components.interfaces.nsIFile);
    file.append("temp");
    LOG('filename ' + filename);
    file.append(filename);
    file.createUnique(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);
    var outputStream = Components.classes["@mozilla.org/network/file-output-stream;1"]
        .createInstance( Components.interfaces.nsIFileOutputStream );
    outputStream.init( file, 0x04 | 0x08 | 0x20, 420, 0 );
    var result = outputStream.write( data, data.length );
    outputStream.close();
    read(mime,file.path);
}

function deleteFile(file) {
    file.remove(false);
}

function buildContact(obj) {
//alert(obj.street1 + obj.street2+obj.city + obj.state+obj.zip);
    var str = "BEGIN:VCARD" +"\n";
    str += "VERSION:2.1" + "\n";
    str += "N:"+obj.name+"\n";
    str += "FN:"+obj.name+"\n";
    if(obj.title)
        str += "TITLE:"+obj.title+"\n";
    if(obj.phone)
        str += "TEL;WORK;VOICE:"+obj.phone+"\n";
    if(obj.fax)
        str += "TEL;FAX:"+obj.fax+"\n";

    if(obj.street1) {
        str += "ADR;WORK:;;"+obj.street1+";";
        if(obj.street2) {
            str += obj.street2+";";
        }
        str += obj.city+";"+obj.state+";"+obj.zip+"\n";
    }
    if(obj.email) {
        obj.email = obj.email.toLowerCase();
        str += "EMAIL;PREF;INTERNET:" + obj.email +"\n";
    }
    str += "END:VCARD";
    return str;
}

function LOG(msg) {
  var consoleService = Components.classes["@mozilla.org/consoleservice;1"]
                                 .getService(Components.interfaces.nsIConsoleService);
  consoleService.logStringMessage(msg);
}
```

Listing 13 – firefoxmash.js

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<%@ page import="java.net.*,java.io.*,java.util.*"%>

<head>
<script type="text/javascript"
```

```
src="http://www.google.com/jsapi?key=ABQIAAAAFdyjfyZbMg2e1toC0Wn6HRRhbiXqwVLAV3idsTmm6kfJ
JgQf2RQj106BTmITBa8CfN9P0PfktZcjaQ"></script>
<script>
google.load("maps", "2.x");
google.setOnLoadCallback(init);
var side_bar_html = "<ol>";
var mapobj = {};

var gmarkers = [];
var q = 0;

function init() {
    mapobj = new google.maps.Map2(document.getElementById("map"));
    mapobj.setCenter(new GLatLng(37.344241, -121.892266), 15);
    mapobj.addControl(new GMapTypeControl());
    mapobj.addControl(new GLargeMapControl());
    plotAddress(mapobj);
}

function plotAddress(map) {
    var lastpoint = "";
    for(b = 0; b < addressarr.length; b++){
        // add the points
        var pt = addressarr[b];
        var point={};
        if(pt.lat == 0 && pt.lng ==0) {
            point = null;
        } else {
            point = new GLatLng(pt.lat,pt.lng);
            lastpoint = point;
        }
        var marker = createMarker(point,pt.address,pt.address);
        if(marker)
            map.addOverlay(marker);
    }
    map.panTo(lastpoint);
  // put the assembled side_bar_html contents into the side_bar div
  document.getElementById("data").innerHTML = side_bar_html+"</ol>";

}

// This function picks up the click and opens the corresponding info window
function myclick(q) {
    GEvent.trigger(gmarkers[q], "click");
}

// A function to create the marker and set up the event window
function createMarker(point,name,html) {
    var marker = null;
    if(point) {
        marker = new GMarker(point);
        GEvent.addListener(marker, "click", function() {
        marker.openInfoWindowHtml(html);
        });
        // save the info we need to use later for the side_bar
        gmarkers[q] = marker;
        // add a line to the side_bar html
        side_bar_html += '<li><a href="javascript:myclick(' + q + ')">' + name +
'</a></li>';
        q++;
    } else {
        side_bar_html += '<li>' + name + '</li>';
    }
    return marker;
}

var lndmrkcnt = 0;

function addLandMark() {
    var obj = document.getElementById("landmarkdata");
    //alert(obj.value);
```

```
    if(obj.value && obj.value.length > 1) {
    var baseIcon = new GIcon();
    baseIcon.shadow = "http://localhost:8080/test/images/landmark_shadow.png";
    baseIcon.iconSize = new GSize(30, 34);
    baseIcon.shadowSize = new GSize(30, 34);
    baseIcon.iconAnchor = new GPoint(9, 34);
    var landmarkIcon = new GIcon(baseIcon);
    //alert("lndmrkcnt is "+lndmrkcnt);
    switch (lndmrkcnt) {
        case 0:
            landmarkIcon.image = "images/A1.png";
            lndmrkcnt++;
            break;
        case 1:
            landmarkIcon.image = "images/B1.png";
            lndmrkcnt++;
            break;
        case 2:
            landmarkIcon.image = "images/C1.png";
            lndmrkcnt++;
            break;
        case 3:
            landmarkIcon.image = "images/D1.png";
            lndmrkcnt++;
            break;
        case 4:
            landmarkIcon.image = "images/E1.png";
            lndmrkcnt++;
            break;
        default:
            landmarkIcon.image = "images/default.png";
            lndmrkcnt++;
            break;

    }
    var markerOptions = { icon:landmarkIcon };
    var geocoder = new GClientGeocoder();
      geocoder.getLatLng(
        obj.value,
        function(point) {
          if (!point) {
            alert(obj.value + " not found");
            lndmrkcnt--;
          } else {
            mapobj.setCenter(point, 15);
            var marker = new GMarker(point,markerOptions);
            mapobj.addOverlay(marker);
            var lobj = document.getElementById("landmarklist");
            lobj.innerHTML += "<li>"+obj.value+"</li>";
            obj.value="";
          }
        }
      );
  } else {
    alert("Please enter a valid address");
  }
}

</script>
<style>
    * {
        font-family:"Trebuchet MS", sans-serif;
        font-size:14px;
    }

    #canvas {
        width:1200px;
        height:580px;
        /*border:1px solid red;*/
    }
```

```
    #top {
        width:1200px;
        height:50px;
        /*border:1px solid black;*/
    }

    #container {
        padding-top:10px;
        margin-left:400px;
        /*border:1px solid black;*/
    }

    #info {
        width:22%;
        height:100%;
        float:left;
        /*border:1px solid gray;*/
    }

    #title {
        padding-left:25%;
        padding-top:5px;
        padding-bottom:5px;
        border:1px solid gray;
        font-weight:bold;
        background-color: gray;
    }

    #ltitle {
        padding-left:15%;
        padding-top:5px;
        padding-bottom:5px;
        border:1px solid gray;
        font-weight:bold;
        background-color: gray;
    }


    #data {
        width:100%;
        height:100%;
        overflow:auto;
        margin-top:2px;
    }

    #landmarks {
        width:17%;
        float:left;
        margin-left:2px;
        /*border:1px solid red;*/
    }

    #map {
        width:60%;
        height:104%;
        float:left;
        margin-left:2px;
        border:1px solid gray;
    }

    li {
        padding-top:6px;
    }
</style>
<title>Firefox Mashup Builder</title>
</head>
<script>
var addressarr=[];
var i = 0;
</script>
<body onunload="GUnload()">
```

```
<div id="canvas">
    <div id = "top">
        <div id="container">
            <input type="text" name="landmark" id ="landmarkdata" value="Drop another
address!" size="50"/> <input type="button" value="Add a Landmark"
onclick="addLandMark()"/>
        </div>
    </div>
    <div id="info">
        <div id="title">
            ADDRESSES
        </div>
        <div id="data">
<%
    int cnt = 0;
    HashMap cache = null;
    cache = (HashMap) request.getSession().getAttribute("cache");
    System.out.println("Querystring  "+request.getQueryString() );
    Enumeration map = request.getParameterNames();
    while(map.hasMoreElements()) {
        String key = (String) map.nextElement();
        System.out.println("Key is " + key);
    }
    if(cache == null)
        cache = new HashMap();
    while(true) {
        String address = request.getParameter("record"+cnt);
        if(address == null)
            break;

        if(address != null) {
            address = address.substring(address.indexOf("::") + 2,address.length());
            address = address.substring(0,address.length() - 2);

            try {

                String [] latlng = new String[2];
                String errorcode = "";
                if(cache.get(address) != null) {
                    latlng = (String[]) cache.get(address);
                    System.out.println(" Got from cache... " + latlng);
                } else {
                    String url = "http://maps.google.com/maps/geo?";
                    String address1 = URLEncoder.encode(address,"UTF-8")            ;
                    url = url +
"q="+address1+"&output=csv&key=ABQIAAAAFdyjfyZbMg2e1toC0Wn6HRRhbiXqwVLAV3idsTmm6kfJJgQf2R
Qj106BTmITBa8CfN9P0PfktZcjaQ";
                    URL g = new URL(url);
                    //System.out.println("URL inside is " + url);
                    BufferedReader in = new BufferedReader(new
InputStreamReader(g.openStream()));
                    String inputLine;
                    while ((inputLine = in.readLine()) != null) {
                        System.out.println(" FETCHING FROM GOOGLE... " + inputLine);
                        String [] data = inputLine.split(",");
                        if(data.length == 4) {
                            errorcode = data[0];
                            latlng[0] = data[2];
                            latlng[1] = data[3];
                            cache.put(address,latlng);
                            break;
                        }
                    }
                    in.close();
                    Thread.sleep(1000);
                }//end Else query Google
                if(errorcode.equalsIgnoreCase("602")) {
%>
<script>
    var obj = {};
    obj.address = "<%=address%>";
```

```
    obj.lat = 0;
    obj.lng = 0;
    addressarr[i] = obj;
    i++;
</script>

<%
                } else {
%>
<script>
    var obj = {};
    obj.address = "<%=address%>";
    obj.lat = "<%=latlng[0]%>";
    obj.lng = "<%=latlng[1]%>";
    addressarr[i] = obj;
    i++;
</script>
<%
                }
            } catch (Exception e) {
                e.printStackTrace();
                System.out.println("ERROR!"+e.getMessage());
            }
        }//end of if address = null
        cnt++;
    }
    request.getSession().setAttribute("cache",cache);
    //TODO put it in session here request.get
%>
    </div>
    </div>
    <div id="map">
    </div>
    <div id="landmarks">
        <div id="ltitle">
            LANDMARKS
        </div>
        <div"lmarkdata">
            <ul id="landmarklist">
            </ul>
        </div>
    </div>
</div>
</body>
</html>
```

Listing 14 – index1.jsp