CS297 Report
Total Recall for Ajax Applications

Smita Periyapatna

smita.periyapatna@gmail.com

Advisor: Dr. Chris Pollett

Department of Computer Science

San José State University

Spring 2008

Table of Contents

List of Figures

1. Introduction

In AJAX, most of the action takes place inside a single page. When an AJAX page is loaded, new instance of JavaScript objects are created. When you leave that page and go to some other page like say Yahoo, the JavaScript objects are completely wiped out. When you hit the back button, the page actually reloads completely. All the objects are lost and this can be pain.

First, it is something that not all programmers are aware of, which can lead to errors, so it is important to know about. Second, users see their state completely wiped out; when they go back to their AJAX application with the back button, they see the original state of their program, not the last place they left it. Third, this can affect performance, since the AJAX application has to re-retrieve everything from the server rather than use its local state.

My project aims to solve the above-mentioned problem. I propose to build an extension for Mozilla Firefox browser, which extends the functionality of the browser to save the states of an Ajax page in something similar to storing normal history items. This will allow users to go back to the last place they left it, instead of going to the original state of their AJAX application. It would also check if some states were already saved in the history item, if they were, then those states would not be saved. My extension will allow users to set a time interval at which they want to save the states of a page.

This report describes the work that was accomplished so far in preparation for CS298. The work in CS297 was submitted in the form of deliverables. The deliverables helped in gathering relevant information and knowledge to implement the final product. The remainder of the report talks about all the deliverables in detail. The report concludes with the future work remaining to implement the final product.

2. Technology Involved

For my extension, I have used eXtensible User Interface Language (XUL) for creating widgets and written Javascript functions to bind user actions. XUL is a XML grammar to add/modify user interface widgets of the browser. I have also used mozilla’s XPCOM interfaces.

2.1. XUL

XUL is an xml based user interface markup language developed by Mozilla. XUL provides a rich set of UI components. XUL can be used to build feature rich cross platform applications. XUL also allows the use of existing web standards and technologies like CSS, JavaScript and DOM.

2.2. Javascript

Javascript is a scripting language used mostly for client side web development. It is the core scripting language in Mozilla. Javascript is used in various levels in Mozilla. A user interface level which manipulates content through the DOM. A client layer which calls on the services provided by XPCOM. An application layer is available in which Javascript is used to create an XPCOM component.

3. Deliverable 1 – Hello World Extension

The first deliverable was to create a test extension to Firefox. I created a simple extension by adding a new menu item to the existing menu, a new menu to the menubar and displayed a string on the status bar panel. The following figure shows a menu item "New History" added to the History menu. The menubar also has a new menu "New Item" added. Also the status bar panel has string "Hello World" displayed.

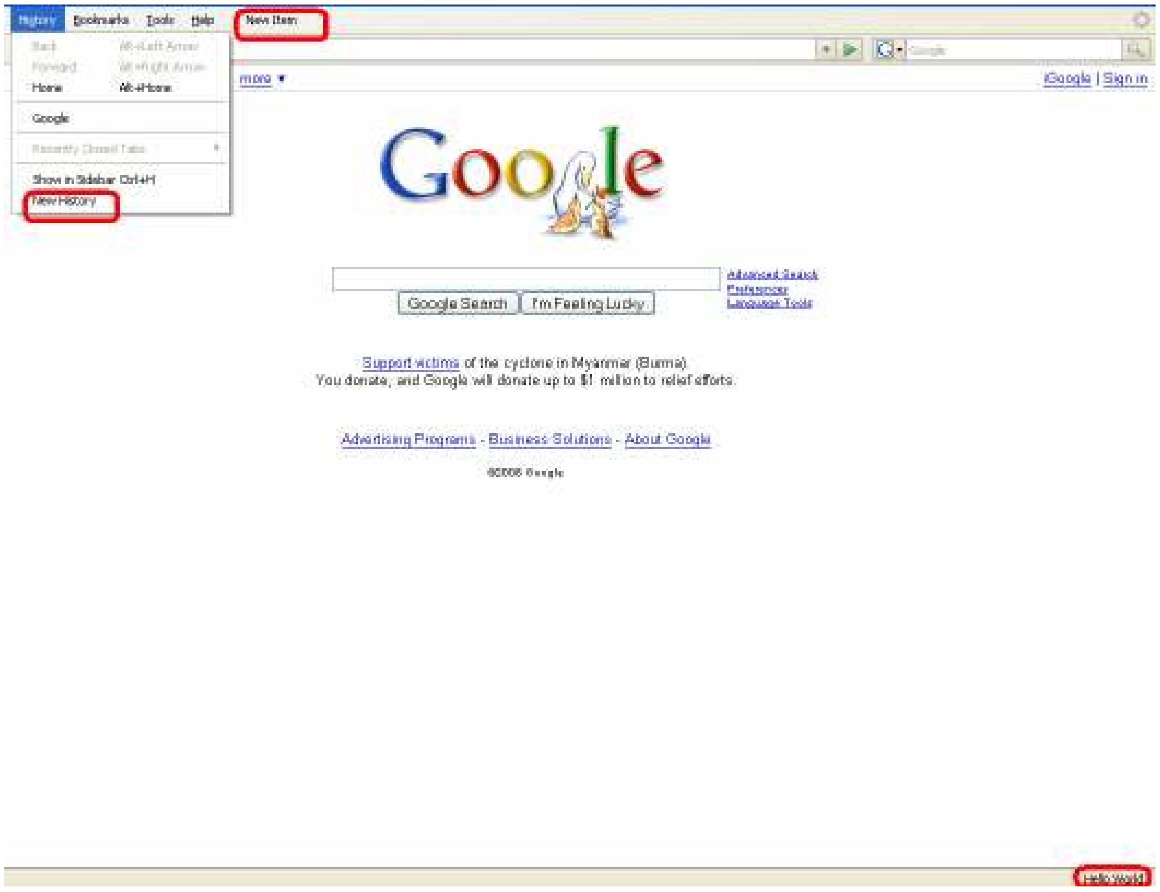


Figure 1: Sample Extension

Following are the steps to create a sample Firefox extension

3.1. *Creating Directory Structure*

Before writing the extension, it is necessary to have the right directory structure

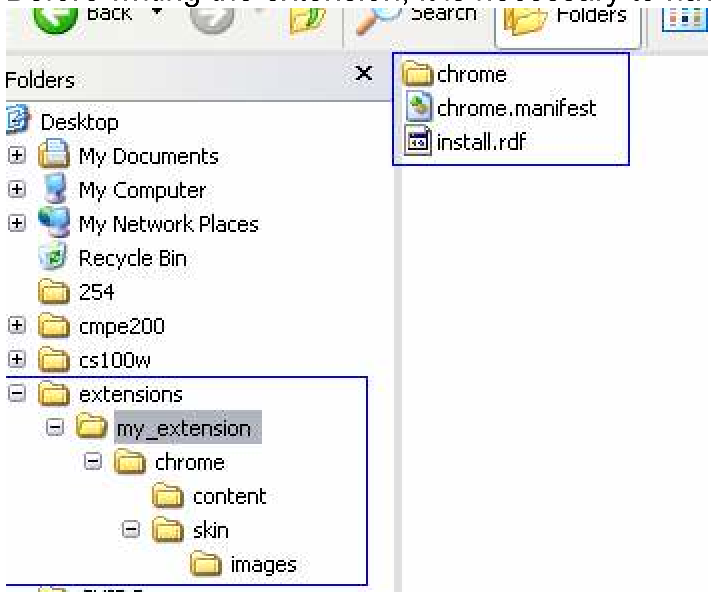


Figure 2: Directory Structure

3.2. *Creating install.rdf*

Install.rdf is used to determine information about an extension as it is being installed. It contains metadata identifying the extension, providing information about who created it, version, etc. Following is the sample install.rdf file:

```
<?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">
    <!-- Required Items -->
    <em:id>test@mail.com</em:id>
    <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>Tester</em:creator>
    <em:description>a sample extension. </em:description>
    <em:homepageURL>http://www.mozilla.org/</em:homepageURL>
    <em:iconURL>chrome://sample/skin/images/pic.jpg</em:iconURL>
    <em:optionsURL>chrome://sample/content/prefSample.xul</em:optionsURL>
  </Description>
</RDF>
```

Figure 3: install.rdf

3.3. Creating chrome.manifest

The chrome.manifest tells Firefox the location of the chrome packages files and overlays. Overlays attach other UI widgets to XUL documents at run time.
The chrome.manifest files also contains the location of the content directory which has the XUL and JavaScript files, Skin which has the images and CSS files and Locale which has the DTD and .properties files.

Following is the sample chrome.manifest file:

```
content sample chrome/content/
overlay chrome://browser/content/browser.xul chrome://sample/content/sample.xul
style chrome://global/content/customizeToolbar.xul chrome://sample/skin/test.css
skin sample classic/1.0 chrome/skin/
```

Figure 4: chrome.manifest

3.4. The XUL file

Following is the XUL file describes the menu items and string displayed on the status bar panel.

```
<?xml version = "1.0"?>
<overlay id = "sample" xmlns = "http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

<!--This displays the "hello world" string on the status bar panel-->
<statusbar id = "status-bar">
   <statusbarpanel id = "my-panel" label = "Hello World"/>
</statusbar>

<!--This creates the new menu "New Item"-->
<menubar id = "main-menubar">
   <menu id = "new-menu">
      <menuitem label = "New Item"/>
      <menupopup id = "new-item">
         <menuitem label = "Add Item" />
         <menuitem label = "Clip Item" />
      </menupopup>
   </menu>
</menubar>

<!--This creates the menu item "New History" under History menu -->
<menupopup id = "goPopup">
  <menuitem label = "New History"/>
</menupopup>

</overlay>
```

Figure 5: XUL File

3.5. Testing extension

Create a text file and put the path of your extension folder. Save the file with the id of the extension given in your install.rdf file. Save the file in your Profile directory. Start Firefox.

4. Deliverable 2 – Capture Events

The second deliverable was about capturing various events generated by mouse clicks. Following figure shows three new menus, “New Menu”, “Clipped Items” and “MenuOnFly”.

It also shows a new icon, “Add Bookmark”, added to the toolbar. Clicking on this icon will display the dialog box, which gets displayed when “Bookmark This Page…” menu item under “Bookmarks” menu is clicked.

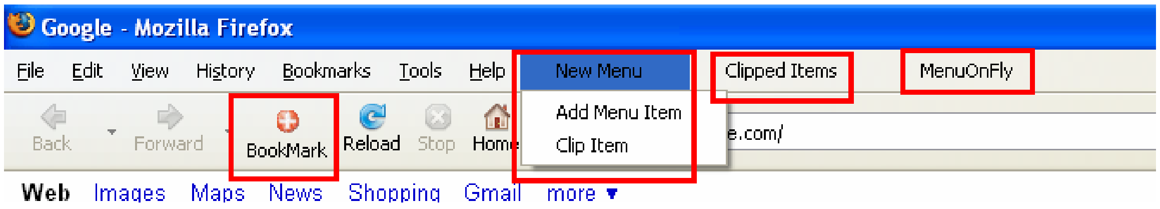

Google - Mozilla Firefox
File Edit View History Bookmarks Tools Help New Menu Clipped Items MenuOnFly
Add Menu Item
Clip Item
Back Forward BookMark Reload Stop Home
Web Images Maps News Shopping Gmail more

Figure 6: New Toolbar and Menubar

Clicking on the “Add Menu Item” menu item will add a new menu item under “MenuOnFly” menu. Clicking on the “menuitem1” will generate an alert box.

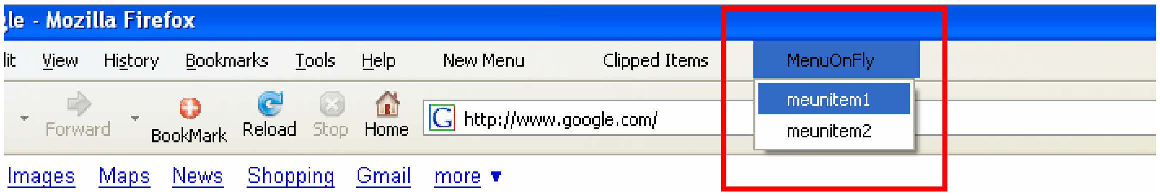

Mozilla Firefox
View History Bookmarks Tools Help New Menu Clipped Items MenuOnFly
meunitem1
meunitem2
Forward BookMark Reload Stop Home
http://www.google.com/
Images Maps News Shopping Gmail more

Figure 7: Adding New Menu Items

Similarly highlighting text on the web page and selecting the “Clip Item” menu will clip the text and add it as menu item under “Clipped Items” menu. Selecting the menu item under “Clipped Items” will allow you to copy and paste that text in any editor.

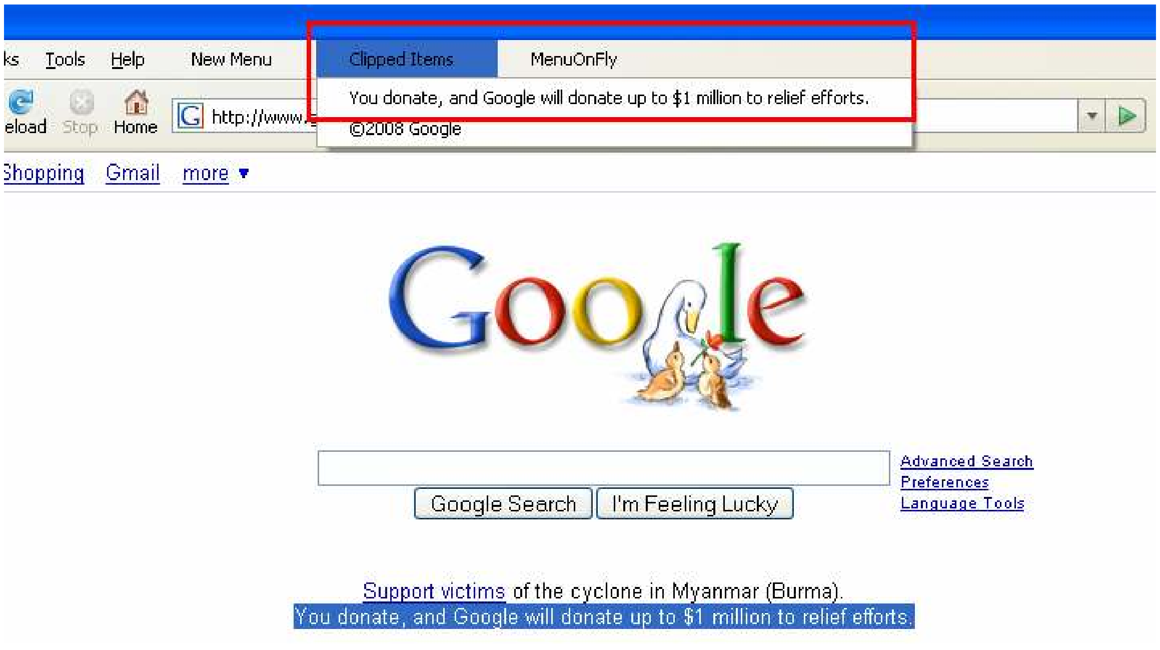

Tools Help New Menu Clipped Items MenuOnFly
You donate, and Google will donate up to $1 million to relief efforts.
©2008 Google
eload Stop Home
Shopping Gmail more
Google
Advanced Search
Preferences
Language Tools
Google Search
I'm Feeling Lucky
Support victims of the cyclone in Myanmar (Burma).
You donate, and Google will donate up to $1 million to relief efforts.

Figure 8: Clipping Text

5. Deliverable 3 – Preferences

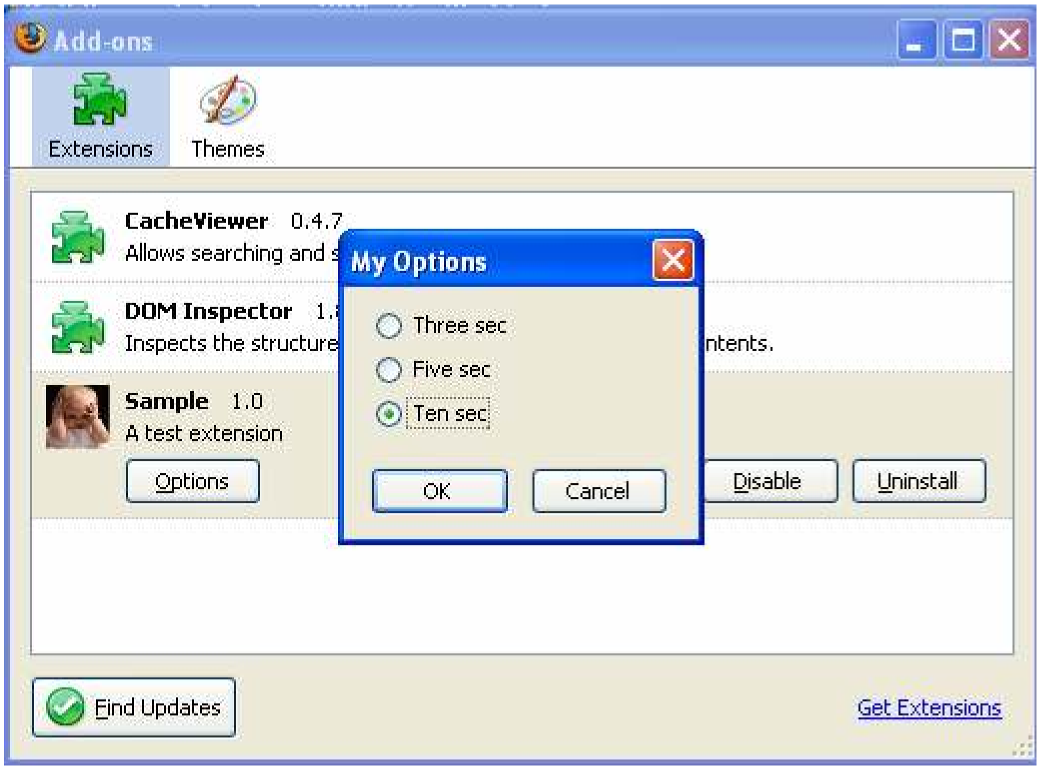
The third deliverable was intended to create a preference system. A preference is any value or defined behavior that can be set by the user. Preference changes via user interface, usually a preference dialog, takes effect immediately. Figure 9 shows the preference file used to create the preference dialog. The figure below that shows the preference dialog. The path to the preference XUL file should be given in the install.manifest file.

This deliverable will allow users to cache pages according to the time interval set by them. After the time interval lapses, the page is cache. Deliverable 4 explains how pages are cached.

```
<?xml version = "1.0"?>
<?xml-stylesheet href = "chrome://global/skin/" type = "text/css"?>
<prefwindow id = "myExtensionOptions" type = "prefwindow"
            xmlns = "http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul"
            buttons = "accept, cancel" title = "My Options">

    <prefpane id = "myPane" flex = "1" >
            <preferences id = "tabsPreferences">
                 <preference id = "extensions.my_extension.myOptions"
                              name = "extensions.my_extension.myOptions" type = "int"/>
            </preferences>
            <script type = "application/x-javascript" src = "chrome://sample/content/pref.js"/>
            <radiogroup id = "myOpt" preference = "extensions.my_extension.myOptions">
                 <radio label = "Three sec" value = "2" />
                 <radio label = "Five sec" value = "3"/>
                 <radio label = "Ten sec" value = "4"/>
            </radiogroup>
    </prefpane>
</prefwindow>
```

Figure 9: Preference XUL File



Three sec [✓ Ten sec] Five sec

Figure 10: Preference Dialog

When the user selects any option, an observer receives the change notifications and a callback function then executes so that the changes takes effect immediately.

XPCOM interface, nsIPrefService, is used for implementing the preference system.

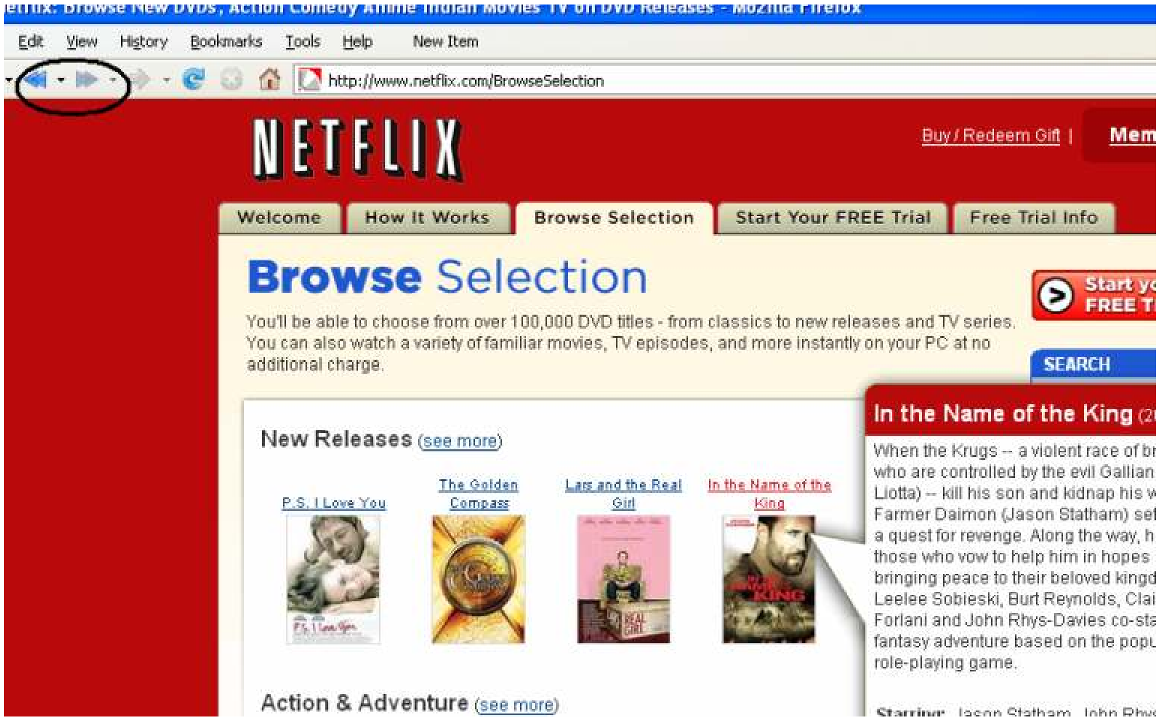
6. Deliverable 4 – Caching Pages

The fourth deliverable was about caching pages according to the user selected preference. The following figure shows two new icons in the toolbar that serves as back and forward buttons to view the cached pages.

The below three figures are from netflix website, where in users can browse movies of different genre.

In figure 11, observe that the details about a particular movie are displayed, as a tool tip only when the user moves the mouse on that movie image/poster. This state (tool tip) of the page is never stored as a history item, when users are browsing that page.

With the help of my extension, these states can be cached. Figure 12 and 13 shows those states of the page loaded from cache, when the user clicks the cacheBackButton and cacheForwardButton respectively.



You'll be able to choose from over 100,000 DVD titles - from classics to new releases and TV series. You can also watch a variety of familiar movies, TV episodes, and more instantly on your PC at no additional charge.

In the Name of the King

When the Krugs -- a violent race of br who are controlled by the evil Gallian Liotta) -- kill his son and kidnap his w Farmer Daimon (Jason Statham) set a quest for revenge. Along the way, h those who vow to help him in hopes bringing peace to their beloved kingd Leelee Sobieski, Burt Reynolds, Clai Forlani and John Rhys-Davies co-sta fantasy adventure based on the popu role-playing game.

Figure 11: Caching Begins

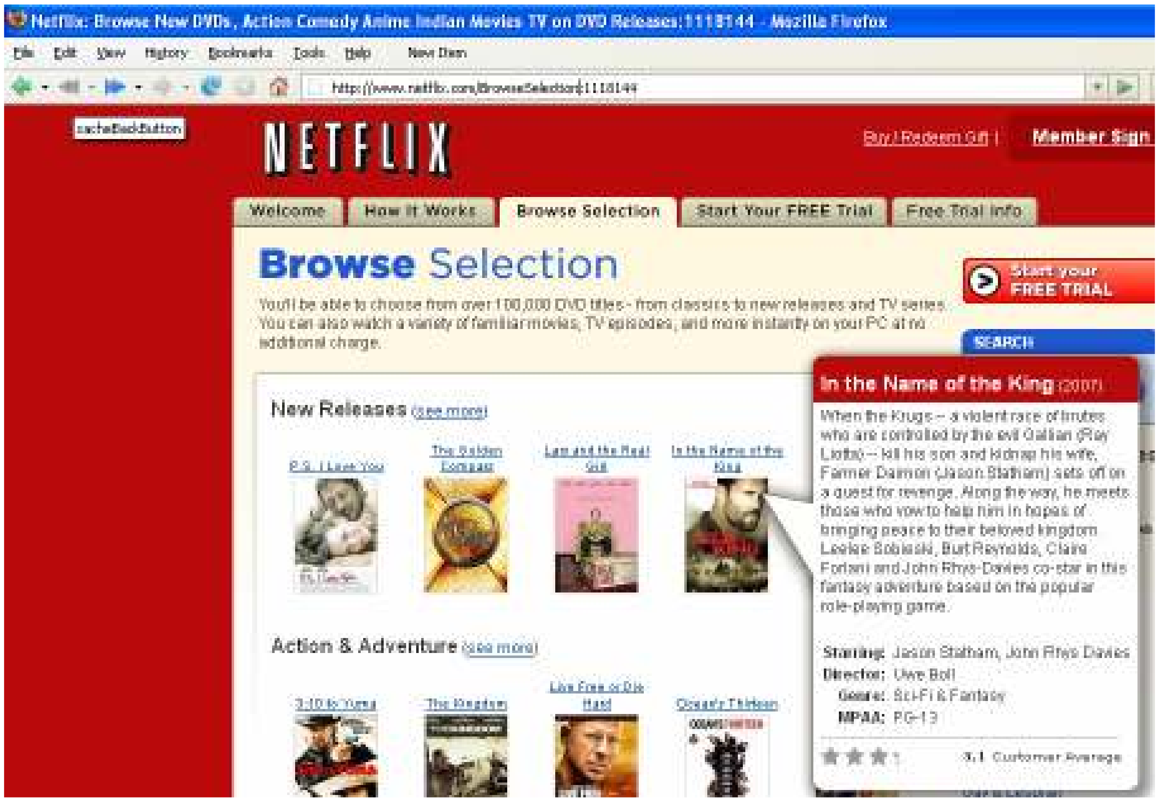
For this delieverable, the page's DOM tree is captured using the body.innerHTML property, which is then written to the disk. The URL of the page is used as a key for writing to and reading from the cache. The URL is appended with the timestamp at which the page is cached, before writing to the cache. I used RDF as an intermediate API between the user interface and cache API.

6.1. Resource Description Framework

Resource Description Framework (RDF) is a simple, cross-platform database for small data stores. Bookmarks, global history in mozilla use RDF. Before writing to cache, the content, the URL and the title are stored as RDF resources. Before writing any new content to the cache the content is compared with all the previously written content to check if it is equal. This is done with the help of RDF.

When the content is cached a new menu item is added to the cacheBackButton, with the title of the page as its label. When the cacheBackButton is clicked the most recent menu item gets appended to the cacheForwardButton menu while it gets deleted from the cacheBackButton menu.

When the cacheBackButton is clicked the document content is loaded from the cache. The following figure shows the button clicked and the page that was cached few seconds back gets displayed. Also the URL in the address bar shows the URL appended with the time stamp, the time at which the page was cached.



Browse Selection

You'll be able to choose from over 100,000 DVD titles - from classics to new releases and TV series. You can also watch a variety of familiar movies, TV episodes, and more instantly on your PC at no additional charge.

New Releases (see more)

Action & Adventure (see more)

In the Name of the King (2007)

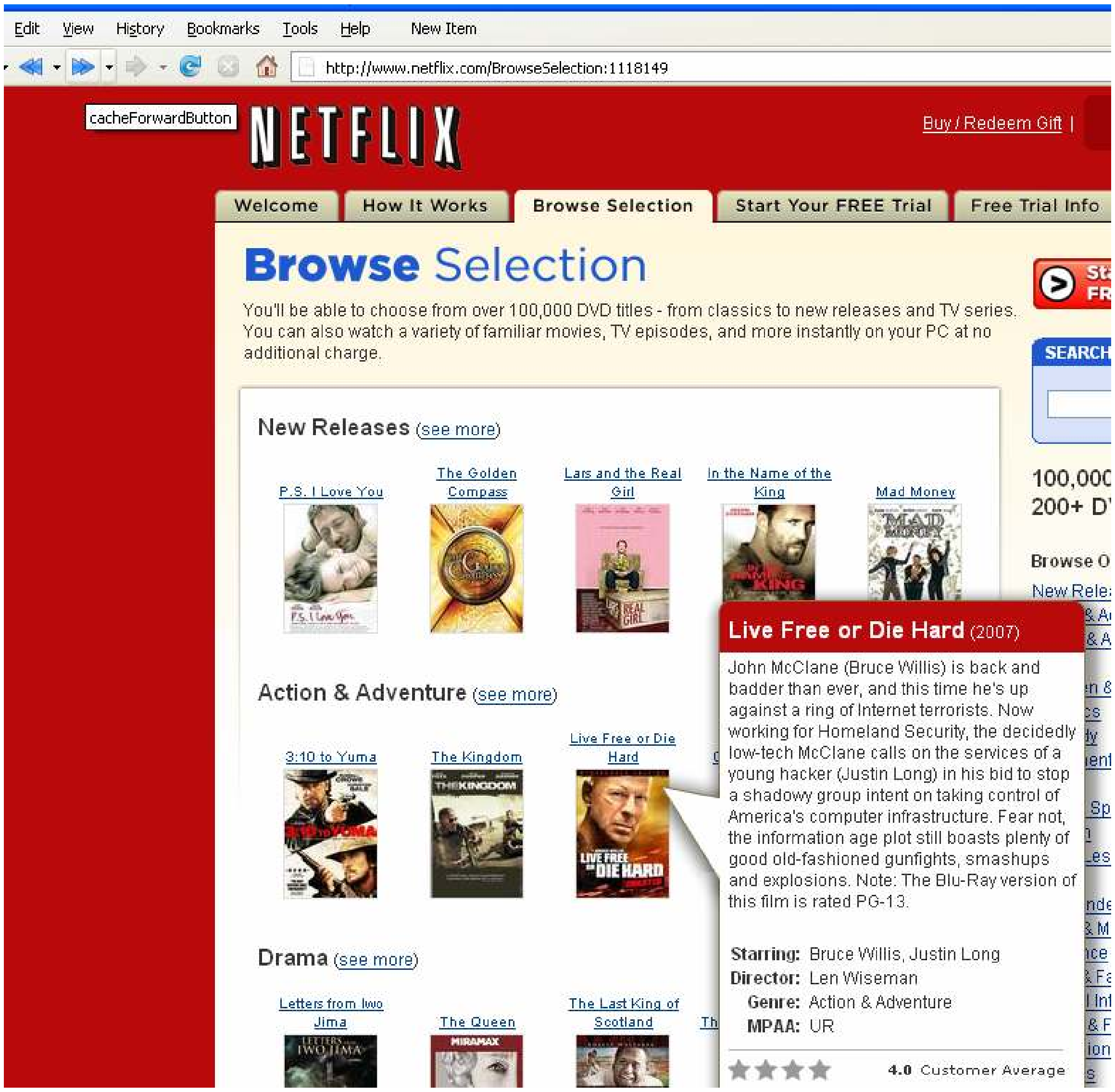
When the Krugs -- a violent race of brutes who are controlled by the evil Gallian (Ray Liotta) -- kill his son and kidnap his wife, Farmer Daimon (Jason Statham) sets off on a quest for revenge. Along the way, he meets those who vow to help him in hopes of bringing peace to their beloved kingdom. Leelee Sobieski, Burt Reynolds, Claire Forlani and John Rhys-Davies co-star in this fantasy adventure based on the popular role-playing game.

Starring: Jason Statham, John Rhys Davies
Director: Uwe Boll
Genre: Sci-Fi & Fantasy
MPAA: PG-13

3.1 Customer Average

Figure 12: Back Button

When the cacheForwardButton is clicked the document content is loaded from the cache. The following figure shows the forward button clicked and the page that was cached few seconds back gets displayed. Also the URL in the address bar shows the URL appended with the time stamp, the time at which the page was cached.



Browse Selection

You'll be able to choose from over 100,000 DVD titles - from classics to new releases and TV series. You can also watch a variety of familiar movies, TV episodes, and more instantly on your PC at no additional charge.

Live Free or Die Hard (2007)

John McClane (Bruce Willis) is back and badder than ever, and this time he's up against a ring of Internet terrorists. Now working for Homeland Security, the decidedly low-tech McClane calls on the services of a young hacker (Justin Long) in his bid to stop a shadowy group intent on taking control of America's computer infrastructure. Fear not, the information age plot still boasts plenty of good old-fashioned gunfights, smashups and explosions. Note: The Blu-Ray version of this film is rated PG-13.

Starring: Bruce Willis, Justin Long
Director: Len Wiseman
Genre: Action & Adventure
MPAA: UR

4.0 Customer Average

Figure 13: Forward Button

XPCOM interfaces like nsICacheService and nsIRDFDataSource were used for implementing writing to and reading from the cache and RDF respectively.

7. Future Work and Conclusion

In CS298, I will extend deliverable four to tabbed browser. This will involve capturing the tab open event and tab close event. This feature will work in a way similar to the way the Back/Forward controls work, with tabbed browser. Also, caching will be done for pages which are implemented using flash, frames etc.

More information has to be gathered on how to capture the changing state of the pages which are implemented using flash.

CS298 will also involve improving and integrating deliverable 3 and 4 only, as deliverable 1 and deliverable 2 was done to know the steps involved in writing an extension to Mozilla. Most of the groundwork for implementing the final project is laid in the form of deliverable 4. This deliverable has couple of things to fix, like when the cacheBackButton/cacheForwardButton controls are used, the pages that are loaded from cache do not display images sometimes. This will be fixed in CS298.

The work done in this semester in the form of deliverables will definitely help in implementing the final project.

8. References

[1] Official page of Mozilla.
"http://developer.mozilla.org/en/docs/Building_an_Extension"

[2] Creating Applications with Mozilla. David Boswell. O'Reilly. 2002.

[3] XUL Tutorial and XPCOM Reference.
"http://www.xulplanet.com/"

[4] JavaScript Reference.
"http://www.w3schools.com/jsref/default.asp"

[5] RDF reference
"http://developer.mozilla.org/en/docs/RDF_in_Fifty_Words_or_Less"