Firefox Extension Development Tutorial :: Preferences

Table of contents

1. Overview

In many cases your extension will need to be able to store data or simple user preferences for future use. In the Home Page Scheduler, for example, the Default Home Page URL and the scheduled Home Pages all need to be saved somewhere so that each time the user opens Firefox the data is available.

Firefox provides you with the Preferences System for these tasks. To see all of the currently stored preferences in your Firefox, type 'about:config' into the location bar and press return. This page will show you a listing of all the current preferences and their values. For example, browser.startup.homepage tells the browser what page to load when the user wishes to visit their home page.

A preference can be an int, bool, or string.

2. Setting Defaults

The preference strings that you create for your extension should follow a simple naming convention. Here is the preference that holds the Default Home Page for our extension:

```
extensions.hpsched.defaultHomePage
```

Begin your strings with 'extensions', followed by your package name, and then the rest can be whatever makes the most sense to you. This just makes sure no two extensions want to create the same preference string for two different reasons.

In most cases you will want to set what should be the default setting for a preference. For example, in the Home Page Scheduler www.google.com is the default startupHomePage if the user does not provide one. You may create these default preferences in a file within your defaults/preferences directory (you created this directory earlier). All .js files within this directory will be loaded by the Preference System when necessary, so the name of your file does not matter. We chose to use our package name: hpsched.js.

Within this file you may add lines like this:

```
pref("extensions.hpsched.defaultHomePage", "www.google.com");
pref("extensions.hpsched.schedules", "");
```

Notice that, by default, no scheduled events are created, so we just initialized this to an empty string.

3. Referencing from JavaScript

The steps necessary to reference the preferences system from within the JavaScript code could have been made simpler, but it is not too bad. First, your file must have access to the Preference Manager component. To do that, you must add the following line of code to your file:

```
            var prefManager =
Components.classes["@mozilla.org/preferences-service;1"]
.getService(Components.interfaces.nsIPrefBranch);
```

Now that you have access to the preference manager you can get and set preferences using getIntPref()/getBoolPref()/getCharPref() and setIntPref()/setBoolPref()/setCharPref(). These functions are very straight forward and have been used in the code samples above.

4. Creating a Preferences Pane

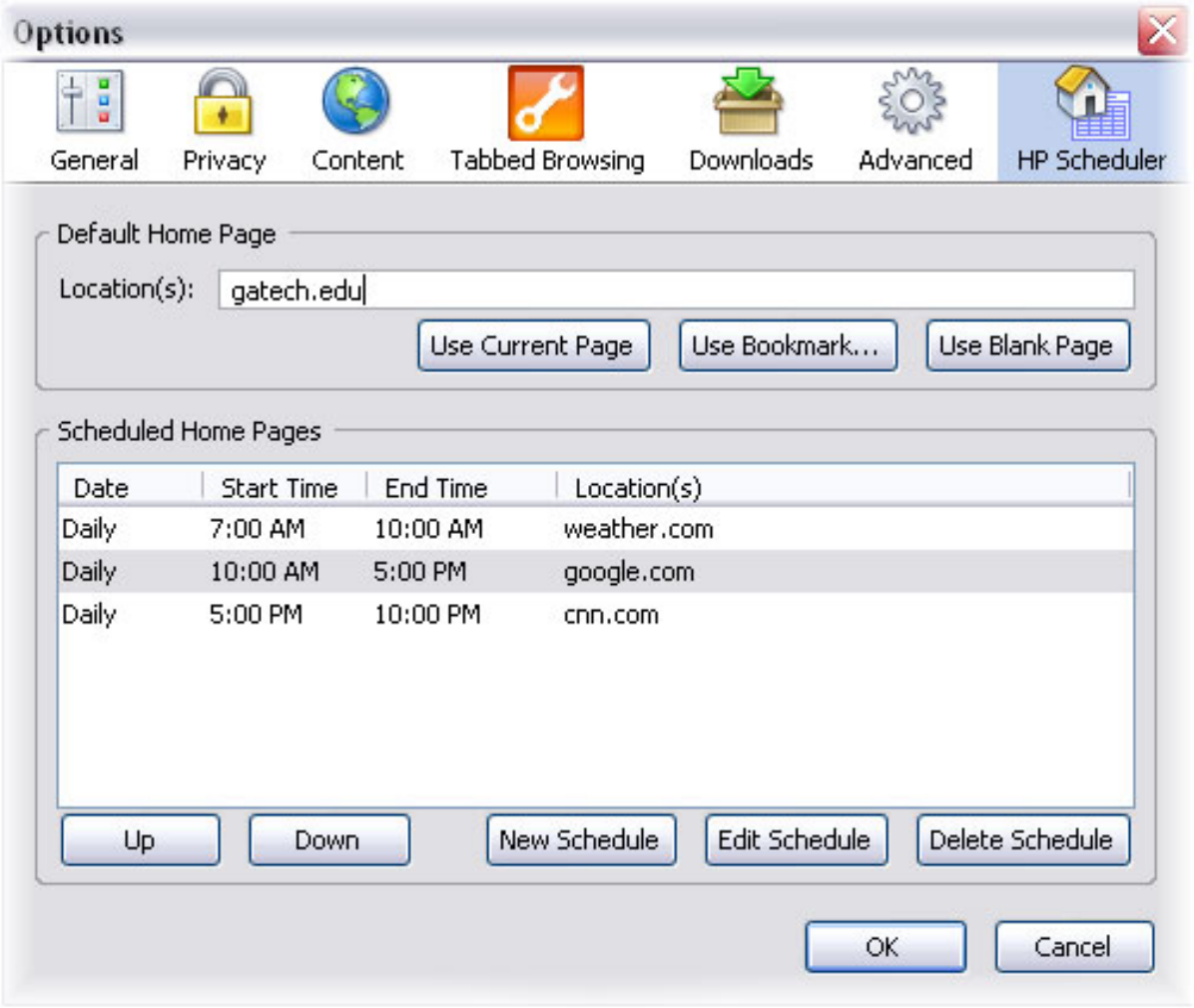
Before we begin using our preference strings to store data, let's learn how to create a Preferences pane. Select Options from the Tools menu in Firefox to open up your preferences. (Note: options and preferences means the same thing in Firefox). You should see General, Privacy, Content, Download, Advanced, and maybe other tabs. If you would like to have a tab for your extension preferences, you must create an XUL overlay onto the preferences.xul file that contains your pane information.

The code below shows how to create an XUL overlay for your pane:

```
          <?xml version="1.0"?>

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

            <!-- Merge with the BrowserPreferences Window -->
            <prefwindow id="BrowserPreferences">

              <!-- Create a new pane (tab) for HP Scheduler. -->
              <prefpane id="hpschedPane" label="&prefpane.label;"
                        onpaneload="populateSchedulesList()"
image="chrome://hpsched/content/images/hpsched32.png">

                <!-- Intermediary between GUI and preferences system -->
                <preferences>
                   <!-- see the next section for more information -->
                </preferences>

                <!-- GUI Elements... -->
              </prefpane>
            </prefwindow>
          </overlay>
```

Notice that the merge point is <prefwindow id="BrowserPreferences">. If you did not know what you want to merge with, the DOM Inspector is your best tool for looking through browser code and finding element IDs. The <prefpane> tag requires an id, label, and image attributes. This image will show up in the Options window. The onpaneload attribute allows you to run code when the options pane is first loaded.

Here is a screenshot of the Home Page Scheduler Preference Pane. Notice that buttons, textboxes, and all other GUI elements work just fine in the Options window.



Preference Window

5. Making XUL Elements bind to Preferences

There are two steps to making your XUL element interface with a preference string. We will use the Default Home Page textbox as an example.

First, within the <preferences> tag shown in the sample above, you will need to define the preference that will be used:

```
            <preferences>
              <preference id="defaultStartupHomepage"
                          name="extensions.hpsched.defaultHomepage"
                          type="string" />
            </preferences>
```

This line creates an element with the defaultStartupHomepage DOM ID, and associates itself with the extensions.hpsched.defaultHomepage preference string. It also lets the preference system know that the data should be stored as a string (not an int or bool). We can now use this element's ID in the XUL GUI element:

```
            <textbox preference="defaultStartupHomepage"
id="defaultStartupHomepageField" />
```

For simple elements, such as checkbox, colorpicker, radiogroup, textbox, listitem, listbox, and menulist, all you have to do is add the preference attribute to the element. The Preference System will automatically update the preference string anytime this element is modified! This was a major Firefox 1.5 improvement.

6. More Difficult Preferences interfaces

In some cases your GUI elements will not have a one-to-one correspondence with the preferences strings. In the Home Page Scheduler, for example, the tree element that holds our scheduled home pages cannot automatically store its data as an int, string, or bool - it is much too complicated. The Firefox Preference Manager provides a way for you to interface with more complicated GUI elements. There are three basic steps:

1. *onpaneload*: Add this attribute to your prefpane tag so that a function can be called to initialize your GUI elements. At this point in the code you can access the preferences strings manually and set the GUI elements as desired.
2. *onsynctopreference="return myFunc();"*: This attribute must also be inserted into the element that will be altering the preference's value. Anytime the user changes the element, the Preference System will call this function to update the preference string. This function must return the new value of the preference string, which will then be stored immediately.
3. *preference-editable="true"*: You must add this attribute to the XUL elements that should cause the preference string to be modified. That way, the Preference System can call your update code as soon as the user modifies the element.

Let's look at the onpaneload code for the Home Page Scheduler. You will see that it manually retrieves the preference and builds the tree's child nodes to match the information:

```
            /*
```

```
             * Called when the preference window loads.  Populates the
Schedules List
             * based on the "extensions.hpsched.schedules" preference.
             */
            function populateSchedulesList() {
              var prefString =
prefManager.getCharPref("extensions.hpsched.schedules");
              if(prefString == "")
                return;

              var schedules = prefString.split(".NEXT.");
              var schedulesList = document.getElementById("schedulesList");

              for (var i=0; i < schedules.length; i++) {
                var pieces = schedules[i].split(".ITEM.");

                var newItem = document.createElement("treeitem");
                var newRow = document.createElement("treerow");
                newItem.appendChild(newRow);

                var date = document.createElement("treecell");
                date.setAttribute("label", pieces[0]);
                newRow.appendChild(date);

                var start_time = document.createElement("treecell");
                start_time.setAttribute("label", pieces[1]);
                newRow.appendChild(start_time);

                var end_time = document.createElement("treecell");
                end_time.setAttribute("label", pieces[2]);
                newRow.appendChild(end_time);

                var url = document.createElement("treecell");
                url.setAttribute("label", pieces[3]);
                newRow.appendChild(url);

                schedulesList.appendChild(newItem);
              }
            }
```

Here is our tree element in the XUL file:

```
            <tree id="schedulesTree" preference-editable="true"
preference="schedules"
                        onsynctopreference="return saveSchedulesList();">
```

Last we must provide the function that converts the information from the tree into a string suitable for preference storage:

```
            /*
             * Save the Schedules List to the
```

```
"extensions.hpsched.schedules" preference.
             * This is called by the pref's system when the GUI element is
altered.
             */
            function saveSchedulesList() {
              var schedulesList =
document.getElementById("schedulesList").childNodes;
              var prefString = "";

              for (var i=0; i < schedulesList.length; i++) {
                var columns = schedulesList[i].childNodes[0].childNodes;
                var str = columns[0].getAttribute("label") + ".ITEM."
                        + columns[1].getAttribute("label") + ".ITEM."
                        + columns[2].getAttribute("label") + ".ITEM."
                        + columns[3].getAttribute("label");
                if(prefString == "")
                  prefString = str;
                else
                  prefString += ".NEXT." + str;
              }

              /* return the new prefString to be stored by pref system */
              return prefString;
            }
```

7. Further Reading

Preferences Overview: http://developer.mozilla.org/en/docs/Preferences_System:preference