SIMPLE WEB BROWSER

Industrial programming

F20SC
Hans-Wolfgang Loidl

Stuart Marples – SM81
H00156296

CONTENTS

Introduction

Throughout this report I will be talking about the simple web browser that has been created and the process that went into creating it. Furthermore, I will be describing how to use this browser for new users and supplying a developer's guide for any developers that may want to understand and develop more to the program with a testing section to show how effectively the program works and what bugs and problems it may run into. The web browser contains features like storing the user's history, storing the user's favourites with a name and being able to choose their own home page.

Requirements Checklist

Requirement	Completed?	Comments
Sending HTTP request messages	Yes	
Receiving HTTP response messages	Yes	
Home Page – Create Home Page URL	Yes	
Home Page – Edit Home Page URL	Yes	
Home Page – Home Page loads on browsers start up.	Yes	
Favourites – Add a Favourite URL	Yes	
Favourites – Add name for Favourite URL	Yes	
Favourites – Modification of Favourite	Yes	
Favourites – Deletion of Favourite	Yes	
Favourite – Open Favourite URL by clicking its name	Yes	
History – Loads list of previously visited URLs	Yes	
History – Navigate to previous and next pages	Yes	
History – Click links to go to page	Yes	
Multi-Threading	No	This has yet to be implemented.

Design Consideration

Code Design:

I stored the details for the favourites, history and home page in different files which get loaded into various variables throughout the code.

To store the Favourites of the user I used a 2D array, which would store the name with the URL. This allowed easy access between the name and the URL which allowed me to program the click on the name of a favourite to open it requirement. The maximum number of favourites that a user can have at once is 50 because a user's favourites are stored only for the current session and any more would be excessive.

To store the history of the user I used an array which stores 50 elements. I used an array because it would allow easy access to specific locations and made it easier to implement the label click function which would open the link once the user clicked the label.

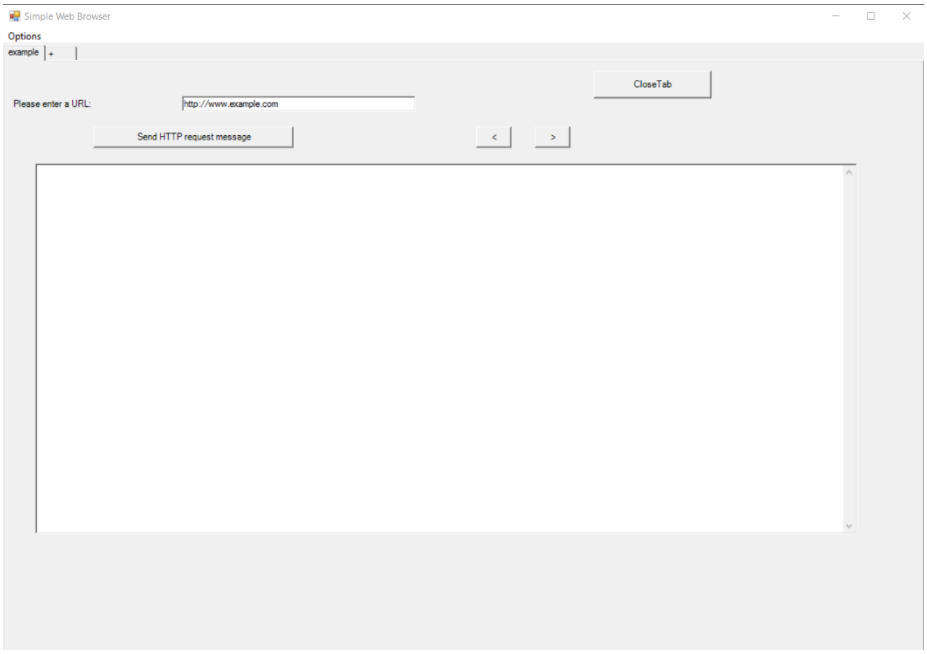
To display the history and the favourites I created a set of array labels for each page that would be generated when the form is created that are blank and in the right position to be filled when a user adds a favourite or has visited a page. I did this to allow the user to click the individual labels to open the links in a new tab.

To implement the Next and Previous buttons I created 2 array stacks, one for the URLs previously visited by the user and one for the URLs the user was on before they used the previous button. This made it easy to keep the history for each tab separate and still maintain functionality.

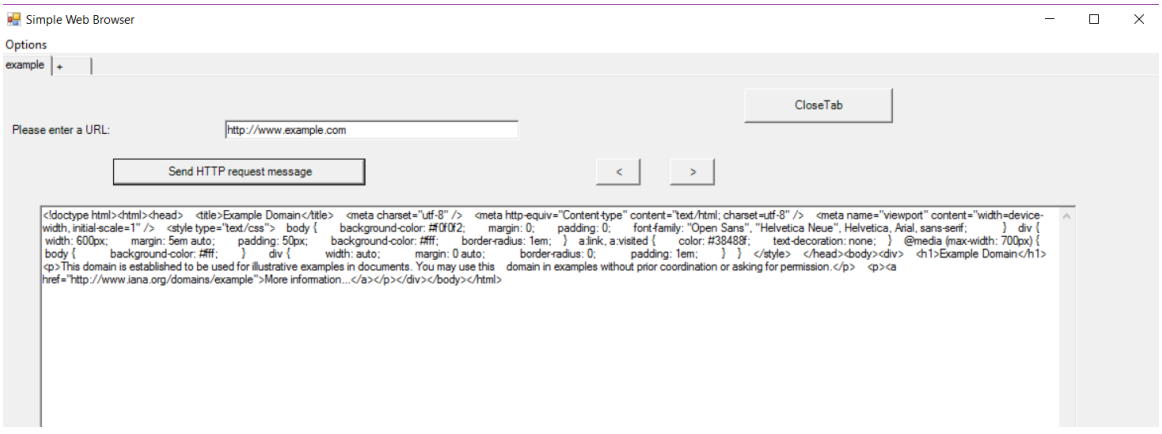
For the Graphical User Interface, I decided to keep the colour scheme a basic grey with the typical layout the same as a browser with the tabs at the top of the form and the options in the top left corner.

User Guide

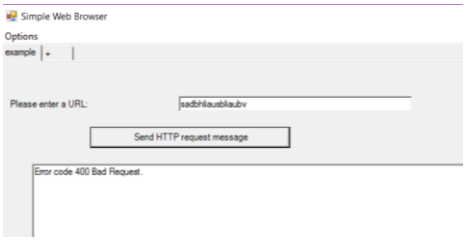
When the user starts the program, they come to the home page which is initially set to http://www.example.com but can be changed by the user.



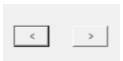
The user can then load the HTML by pressing the send HTTP request message button. This will insert the HTML code into the textbox as shown below.



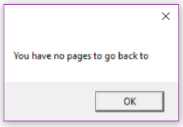
If the user has an error while trying to load the HTML a message will appear in the textbox that explains why the error has occurred.



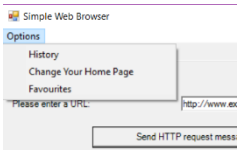
There are previous and next buttons which are the two arrow buttons which will take you to the page previously visited before your current page, or a page



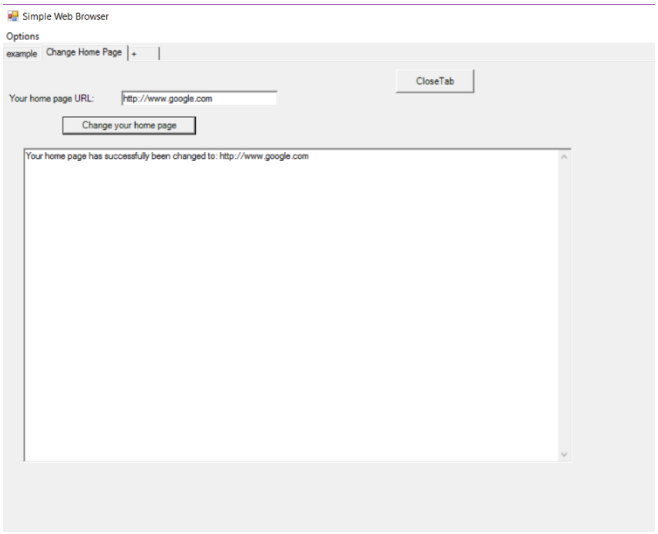
visited after your current page. If there is not a previous page or a next page you will have a pop up message which will alert you to this.



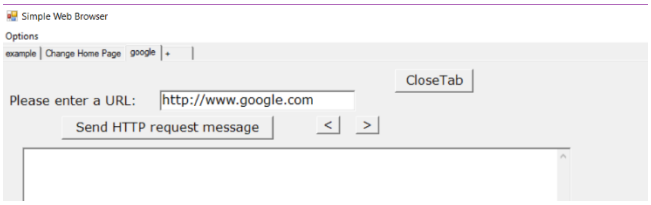
If the user wishes to change their home page they can click on the options menu at the top corner which will give out a list of options.



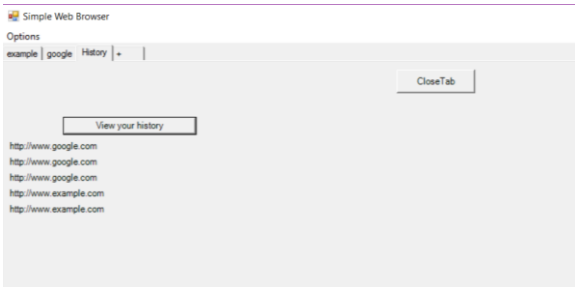
Once you click Change your home page there will be a new tab that opens which will allow you to enter a new URL which will change the page that opens when you open a new tab.



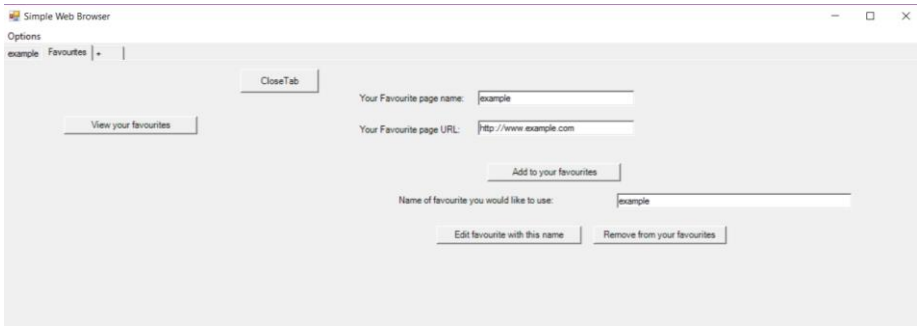
You can open a new tab by clicking the + tab at the top of the page which will now open a tab to the newly created home page URL.



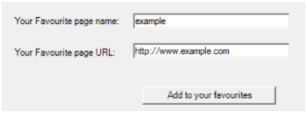
To close this tab, you press the close tab button in the top right of the form and the current tab will be removed from the list of tabs. To view your history, you click the options in the top left and select the history option. This will open a new tab where you can click the view your history button to display up to the last 50 things you have visited. You can then click on any of these links to open them in a new tab.



To open your favourites, you click the options menu in the top left corner and select the favourites option which will open the page in a new tab.



You can add a favourite by typing into the two textboxes the name and the URL of the desired favourite and then clicking the add to your favourites button.

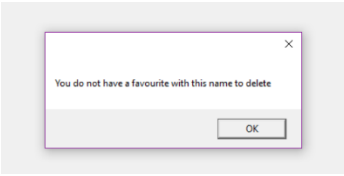


To view your favourites, you press the view your favourites button which will display a list of the names of your favourites and the URLs accompanying them. You can then click on any of the labels shown and they will open the link in a new tab.

To edit a favourite, you enter the favourites name in the lower textbox on the right and click the edit favourite with this name button. This will then load the details of the required favourite into the textboxes above for editing. You then click the add to your favourites button for the changes to be saved.

To remove a favourite, you enter the name of the favourite in the lower right hand textbox and click the remove from your favourites button.

An error can occur for various reasons such as trying to enter a favourite with the same name as another, trying to view favourites when none exist and trying to remove or edit a favourite with a name that doesn't match any favourite saved. This will result in an error message being shown in a pop up box.



Developer Guide

The first third of the programs code is made up from creating the original page for loading HTML, the history tab, the favourites tab and the home page tab. This includes segments of code which set the location, name, size and text for the various labels, textboxes and buttons.

```
// URLLabel
URLLabel.AutoSize = true;
URLLabel.Location = new Point(5, 40);
URLLabel.Name = "URLLabel";
URLLabel.Size = new Size(78, 13);
URLLabel.Text = "Please enter a URL:    ";
```

There is also a large set of loops from lines 100-159 which will iterate through an array of labels, setting the location for each one and making each individual label clickable for opening URLs in new tabs. To Find the correct location I generated variables called “x” and “y” that would hold the x and y coordinate of each label temporarily. These would change after each label was generated so that the name and the URL labels would be next to each other. It would display the first 25 favourites one after the other, and then would display the rest further to the right in the same manner. To make these labels clickable I created 2 functions, one which would read the second label to find the URL and then open it if clicked, and another which would read the label that contained the name if that was the label clicked and find the URL associated with that name to open it.

```
y = 100;
while (i < favlabels.Length)
{
    favlabels[i] = new Label();
    favlabels[i].Location = new Point(x, y);
    if (i != 0 && i % 2 != 0)
    {
        x = x - 100;
        y = y + 20;
    }
    else
    {
        if (i == 0)
        {
            x = x + 100;
        }else
        {
            x = x + 100;
        }
    }
    if (i == (favlabels.Length / 2) - 1)
    {
        x = 300;
        y = 100;
    }
    //Adds the labels to the Favourites Tab
    FavTab.Controls.Add(favlabels[i]);
    if (i % 2 == 0 || i == 0)
    {
        //Opens link in new tab if they click on the name of a favourite
        favlabels[i].Click += new EventHandler(fav_Click);
    }
    else
```

Then between lines 343-358 I create the menu “options” which allows the user to open the home page, history and favourite tabs. This section of code creates the options and adds the appropriate text and the appropriate clickFunction which opens the tab the user requested.

From lines 360 – 385 I assigned values to the tab control for size, location, name and the initial pages to be loaded to it along with an EventHandler which will run when the user selects the “+” tab.

Lines 400 - 452 consists of 3 functions which run when the user wants to open either the favourites, the home page or the history tab. The function for each one will just add the required buttons, labels, textboxes and add the page to the list of available tabs at the top. Each function contains a line similar to: “`dynamicTabControl.TabPages.Insert(dynamicTabControl.TabPages.Count - 1, HomeTab);`”
which inserts the tab into the list of tabs to keep the “+” tab at the far right of the list.

The next function buttonClose_Click() is the function which will close the tab that the user wants closed. It does this by reading the name of the button that called the function which is easy for the main 3 pages, history, home page and favourites. But to find which tab

pressed the button I use the code: "`int.Parse(temp.Name.Split('n')[1])`" where temp is the details from the button that was pressed. This code will find the number at the end of the button name which is incremented or decremented with every tab that opens or closes so that the number can correspond with the tabs. It also decrements the counters for the various other items for each tab such as btcount which is the button count. This gets decremented by 3 because there are 3 buttons in every tab that gets opened.

The next 2 functions from lines 454 – 516 are the functions for when a label is clicked in the history or favourites tabs. An easy way to do this was to temporarily change the home page URL to the URL that the user had clicked and then open a new tab which would automatically load the home page. The boolean variable "favTab" would only get set to true to allow the buttonNew_SelectedIndexChanged() to create a new tab without clicking the + button. This would later get changed back to false in the other function. The fav_Click worked very similarly to the label_Click, but would read the name of a favourite and then find the URL in the array before changing the homeURL and opening the new tab.

```
{
    string temp = homeURL;
    homeURL = l.Text;
    favTab = true;
    buttonNew_SelectedIndexChanged(null, null);
    homeURL = temp;
}
```

HTML_Click is the function which will take a URL from the user and load the HTML into the appropriate textbox. Throughout this function I use the if statement "`if (temp.Name == "HTMLButton" || temp.Name == "BackButton" || temp.Name == "ForwardButton")`" to check if the intial page is the one that is being used. The only difference this makes is the corresponding textbox that needs to be read in from and the textbox to display the HTML. Finding the textbox to read from used the code "`textboxes[(((int.Parse(temp.Name.Split('n')[1])) / 3) * 2) + 1].Text`" where temp is the details from the button that was pressed. The code finds the number of the button that was pressed using (temp.Name.Split('n')[1]) and since each tab has 3 buttons, we can find the number of the tab - 1 by dividing the button number by 3. Since there is 2 textboxes in each tab we then need to multiply the number by 2 and then we add 1 to compensate for the -1 we had for the tab number.

The function then uses the URL to send a web request which if succeeds then adds the URL to the stacks for going backwards and forwards with the "<" and ">" buttons. This also uses a similar mathematical function to find which stack applies to the tab that pressed either of these buttons. It then collects the response from the request and adds the URL to the history array. It finally attempts to print the HTML in the next textbox while still displaying the URL in the first textbox. I added the line "`HTMLTab.Text = cPage.Split('.')[1];`" which will change the text displayed in the tab to the name of the URL being loaded, so if the URL was "http://www.google.com" then it would display "google" at the top. If the request or response fails, an exception can be caught which will find the status code of the failure and display the error message and error code to the user.

The next function Home_Click takes a users input and writes it into the file, then reloads the home file into the homeURL variable and outputs a success message.

The function History_Click loads the data from the file into the history array and then displays each element of the array into different labels. It starts from the end of the array to display the most recent elements first.

The function ListFav_Click is very similar to History_Click but with an additional set of labels to display the name and the URL next to each other.

The function EnterFav_Click starts by looping through the array to see if a favourite with that name already exists and if so gives an error message. There is a boolean variable full which will be set to false if there is anything in the array and if there is not then it will be used to display an error message saying the user has too many favourites and some need to be deleted. Otherwise it will write the details of the new favourite into the file and load the file into the array. If the user is editing their favourites then the variable faved gets set to true, which will then change the details of their favourites in the array. It then carries on to rewrite the array into the file to save these changes permanantely.

The function DeleteFav_Click will search for the favourite to delete, if it does not exist it will output an error and if it does exist then it deletes it from the array. It will then write the array into the file to save the deletion. There is also the variable deleteCount which will increase when something is deleted, it will then set the labels for however many items were deleted to null so that they don't display something that has been deleted or an old duplicate.

The function EditFav_Click finds the name of the favourite to edit and loads the details of the favourite into two textboxes ready to be edited by the user. It then sets faved to be true so that when the user is done editing and adds it to his favourites, it achieves the appropriate response.

The next two functions are BackButton_Click and ForwardButton_Click which work in a very similar way. For both buttons it will take the name of the button that pressed it to find which tab is being used. It then checks that there is something to either move forward or backwards to and if not outputs an error message. If there is something to load, the back button will pop the value and push it onto the forward stack. It then sets the textbox to be the previous URL ready to be loaded. The forward button is similar but done in reverse where the textbox will be the value that is popped off the forward stack, and that value is pushed into the backwards stack.

The next 3 functions are for loading the history, favourites and home page from files. These all work very similarly where they check that the file exists and if it doesn't then it creates the file and then carries on. It then reads from the files and inserts the data into various variables which are then ready to be used in the other functions. The loadhistory function has a couple of checks which makes sure it displays the most recent 50 items from your history and then if the file has 100 enteries or more it will clear the file to stop the file getting too big.

The final function is the buttonNew_SelectedIndexChanged which adds a new tab when the user opens the “+” tab or if the user is opening a favourite or an item from their history. If there is 50 tabs open it will not allow the user to open more and ouput an error to let the user know. Otherwise it will generate a new tab with new buttons, textboxes and labels with names, locations and text and click events and then adds them to the tab. The code then adds the tab to the tab control and increases the counters for the tab, buttons and textboxes.

Testing

Testing the load HTML function.

Test	Expected output	Actual Output	Comments
http://www.google.com	The expected HTML.	The expected HTML	
www.google.com	Error code 400	Error code 400	
www2.macs.hw.ac.uk/sm861	Error code 404	Error code 404	
http://www2.macs.hw.ac.uk/~djg30/WebBrowserTesting/test_forbidden	Error code 403	Error code 403	
iykjvkuygvkuyvouvyouyv	Error code 400	Error code 400	
#!"£$%^&U*IO	Error code 400	Error code 400	

Adding/editing/ Favourites

Test	Expected output	Actual Output	Comments
Adding “example” “http://www.example.com”	The favourite to be added	The Favourite was added	
Adding “example” “http://www.example.com”	Error stating name for favourite already exists	Error stating name for favourite already exists	
Adding “google” “www.google.com”	The favourite to be added	The favourite to be added	When opened, this will lead to a 400-error due to the missing the “http://”
Edit “google” to “google” “http://www.google.com”	The favourite to be successfully edited	The favourite to be successfully edited	
Edit “facebook” to “facebook” “http://www.facebook.com”	Error because there is not a favourite with the name “facebook”	Error because there is not a favourite with the name “facebook”	
Edit “example” to “google” “http://www.example.com”	Error due to name being the same as another favourite.	The favourite was successfully edited	This is because there was only an error check in the code for adding

			and not for editing.

Removing Favourites

Test	Expected output	Actual Output	Comments
Remove “google”	The favourite to be deleted	The favourite was deleted	
Remove “facebook”	Error message saying there isn’t a favourite with that name	Error message saying there isn’t a favourite with that name	

Changing the home page

Test	Expected output	Actual Output	Comments
Change to “http://www.google.com”	The home page has been changed	The home page has been changed	
Change to “www.google.com”	The home page has been changed	The home page has been changed	This will successfully change the user’s homepage but will load a 400 error when the home page is loaded.

Conclusions

In conclusion, I like the way that the browser looks and works with loading the HTML and displaying the name of the page in the tab text. With more time, I could have finished the multi-threading for each tab and would have liked to have figured out a way to have an unlimited number of tabs instead of having to set the maximum a user could use.