F20SC: INDUSTRIAL PROGRAMMING

CW1: Web Browser

Callum Forsyth
cf51@hw.ac.uk

Introduction

The purpose of this coursework was to develop a simple web browser in C#, windows form application was used to create a simple GUI for the project. During this coursework I improved my programming knowledge while using advanced programming techniques, I have improved my object oriented programming skills vastly and I now have a better understanding of reading documentation and selecting the correct libraries.

Assumptions

I made a number of assumptions throughout my development of the web browser.

1. There are certain URL's that return encoded html or incorrectly formatted html, this is likely due to my web browser having no support for cookies or other formatting techniques such as JavaScript and CSS. I have made the assumption that we will be graded on the ability to send and receive HTTP requests/response and not the actual content that is returned, therefore I have not included any additional features to deal with correct formatting of HTML code.
2. I have assumed that the user has some previous knowledge of using a web browser, therefore I have not included error handling at every instance the user has the ability to enter a string, for instance when creating a home page, if the user enters an incorrect URL this will not be caught until the URL is loaded in the browser.
3. I have also assumed that it is better to prioritise meeting requirements over the usability of the project, therefore I have used a very simple GUI with minimal additional features to aid with useability and styling. I have included necessary buttons and textboxes in order for the application to be functional, but I have not focused on the usability of the GUI.

Additional Features

Delete History Feature – I implemented a button on the history menu that allows the user to delete all of the history, it does this by over writing the history file with blank text.

Close Menu Button – I implemented a button on each menu that allows the user to close the menu if they have accidently opened it, all of the menus auto close when an option is selected but this allows the user to close the menu if they open it and then decide not to use that menu.

No URL prompt – I implemented a feature to alert the user that they have clicked the 'GO' button with entering a URL, it prompts the user to input a URL using a message box.

Requirements Checklist

Requirement	Achieved	Note
Sending HTTP request messages	YES	HTTP requests are sent in the HttpClass.cs class using System.Net.WebRequest.
Receiving HTTP response messages	YES	HTTP responses are handled in the HttpClass.cs class using System.Net.HttpWebResponse.
404 Bad Request – Error Code	YES	Error codes are caught using an exception and displayed to the user.
403 Forbidden – Error Code	YES	Error codes are caught using an exception and displayed to the user.
404 Not Found – Error Code	YES	Error codes are caught using an exception and displayed to the user.
Display	YES	The user is able to reload the current page as well as view the HTML/Error codes.
Home Page	YES	Users are able to create and edit a Home Page URL, the home page is set to my university home page and is loaded into the browser on start up.
Favourites	Partially	Users are able to add a URL to a list of favourites and associate a name with this URL.
History	YES	The browser maintains a list of URL's, the user is able to navigate to the previous/next page and jump to a page by clicking the link in the history list.
Bulk Download	NO	This requirement was not met.

Design Considerations

Class Design

When choosing my class design I decided to have separate classes for each requirement in the project, I then split those classes into separate methods and tried my best to have each method only performing one task, this helped to avoid code duplication as much as possible as I was able to make generic methods than I can then call with different parameters when needed. My Form1.cs Class is my largest class as I was unable to call certain methods from elsewhere as this Form class deals with the elements and event handlers for the GUI content.

Data structures

History List:

```
List<string> historyFileList = new List<string>();
```

I used a list to store the users URL's for the History requirement, this list is populated by reading the lines of the text file and entering the elements into the list, I chose a list as it allows me to store the URL's in the order they are searched for, it also allows me to search an index position which I used when implementing the forward and back buttons. I chose to reverse the list using `historyFileList.Reverse();` as this returns the URL's in the order they are searched for, with the most recent URL at index position 0.

Favourites List:

```
List<string> favFileList = new List<string>();
```

This list is similar to the history list, it is populated by reading in the lines of the text file, the Name is entered into Index 0 with the URL in index 1. I did this to allow me to search the list for the name using, `.FindIndex(ea.StartsWith);` and then return the data at that index position + 1. This will always return the URL which I can then use for searching the favourites.

GUI Design

When designing the GUI I chose a minimalist design, my form contains buttons, text boxes and panels. I tried to keep the buttons/text boxes to a minimum and only implement the required functionality in order to make the GUI as usable as possible. I followed the common format for a web browser with the back/forward buttons, home button symbol and refresh button symbol.

Advanced Language Constructs

I made use of exceptions in my HTTP class, I used try catch statements to catch any exceptions that arise during the http request, for example if a user enters a url that creates a 403 forbidden or 404 not found exception the program does not crash it instead catches the exception and displays the error code to the user and prompts them to enter a new url.

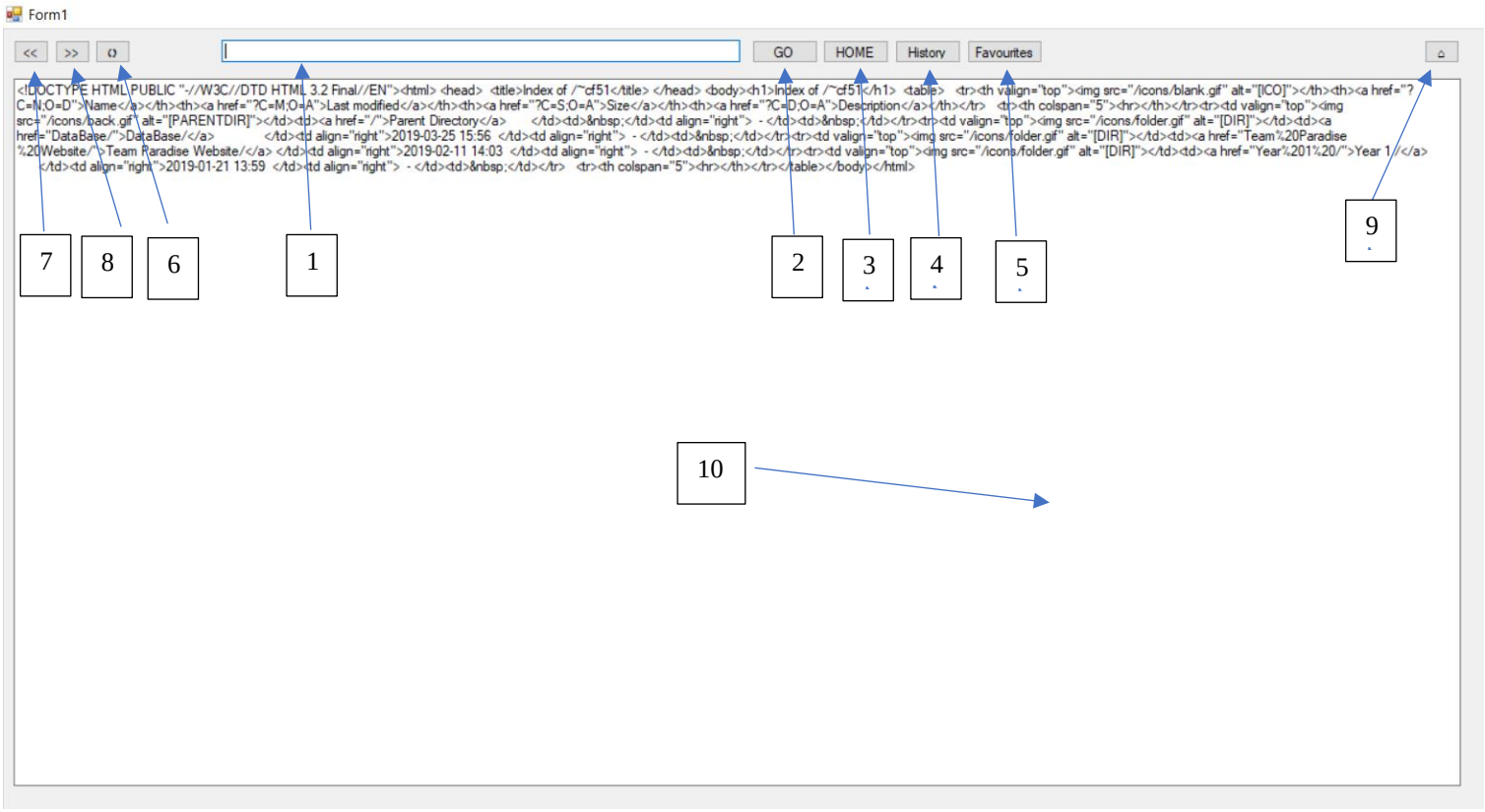
Performance related considerations

Auto Close – I implemented a feature to automatically close a menu when the information is filled out correctly, I did this to reduce the amount of time the user needs to send on clicking buttons and it also avoids multiple events for each button click when the menu closes itself, reducing code thus making the application faster and more useable.

Pre-Completed – I implemented some of my textboxes to already contain the common strings that they will need to enter, for instance the ‘http://www.’ Is already pre completed on the set favourites menu. This reduces the time the user spends entering text.

Auto-Complete – I implemented a feature to automatically change the url in the search bar when the user clicks the back and forward button, this shows the url changing as the html changes and make it faster to adjust the url if the user wanted to add information to the url to navigate to another page such as adjusting ‘http://www.google.come’ to ‘http://www.google.com/gmail/’

User Guide



1 – Search Bar: This is used to enter your URL in the format ‘http://www.google.com’.

2 – Go Button: This is used to complete your search once a correct URL is entered in the search bar.

3 – Home Button: This is used to navigate back to your Home Page.

4 – History: This is used to display the History Menu.

5 – Favourites: This is used to display the Favourites Menu.

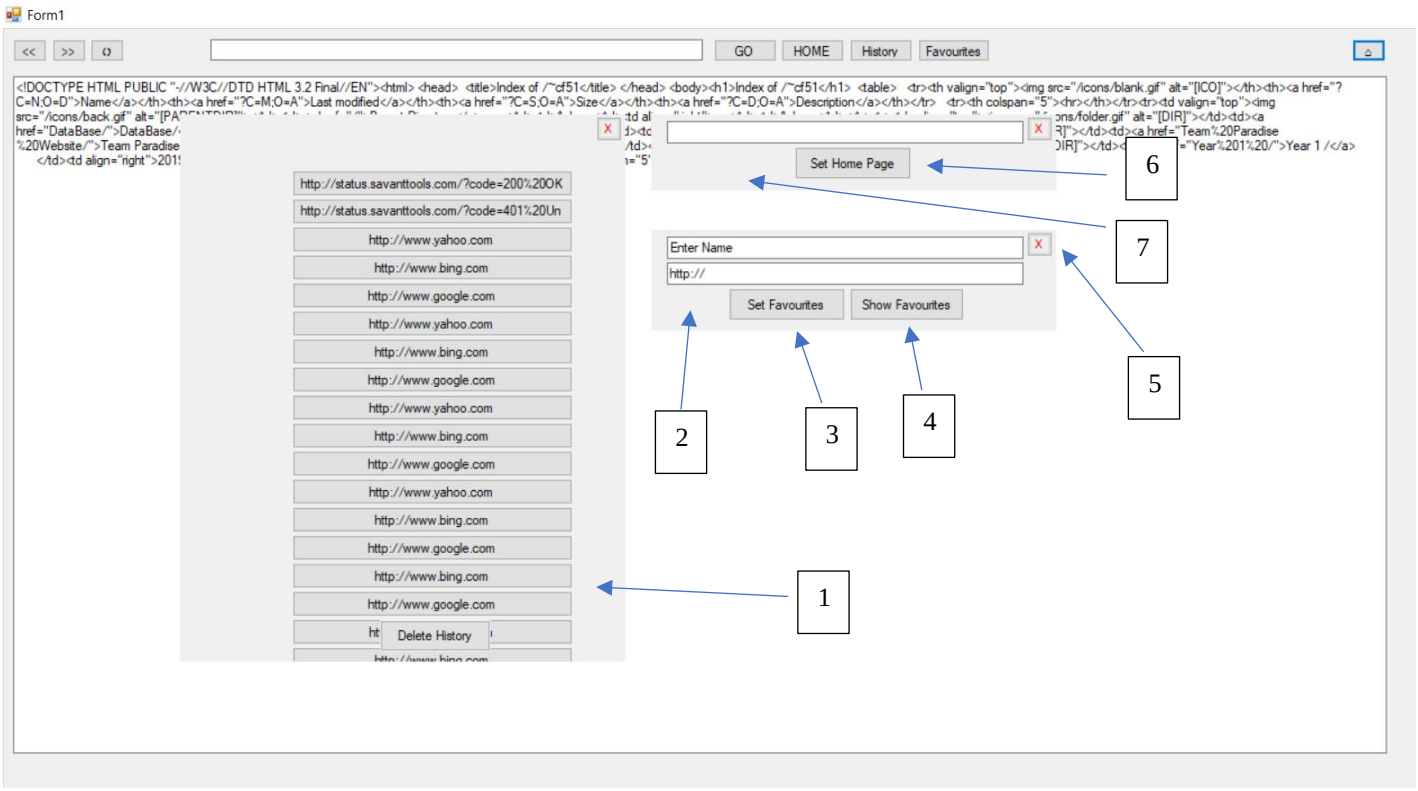
6 – Refresh Button: This is used to refresh the current page that you are on.

7 – Back Button: This is used to navigate back to the previous page you were on.

8 – Forward Button: This is used to navigate forward to the previous page you were on.

9 – Home Menu Button: This is used to display the Home Page Menu.

10 – Display Box: This is used to display the HTML code.



1 – History Menu: This displays your history, you can click on the link to navigate to your desired page.

2 – Favourites Menu: This is used to set your Favourites.

3 – Set Favourites Button: This is used to set your Favourites once the above fields have been completed

4 – Show Favourites Button: This is used to show your Favourites.

5 – Close Button: This is used to close the current menu.

6 – Set Home Page Button: This is used to set your Home Page once the above field has been completed.

7 – Home Page Menu: This is used to set your Home Page:

Developers Guide

HTTPClass.cs

- This calls deals with the HTTP requests and responses. It contains two methods, firstly the 'GetResponseWebRequest()' method creates a HTTP request with the url in the search box, it also contains error handling to display an error message to the user if there is a 403, 404, 400 … error, next the 'GetResponse' method awaits for a url and creates a http response message, it then returns the response content.

History.cs

- This class deals with the History requirements, it contains four methods 'createHistory()', this method creates a file using FileStream on the condition that a file does not already exist. The next method is 'deleteHistory()', this uses a StreamWriter to over-write the current contents of the file with string.Empty. Next is the 'setHistory()' method which uses a StreamWriter to append the text with the URL's from the users searches. Finally the 'getHistory()' method, this firstly opens the file then reads all of the lines to the historyFileList, it then reverses the list and returns it.

Favourites.cs

- This class deals with the favourites requirement, it contains three methods. Firstly the 'createFavourites()' method creates a new favourites file using FileStream on the condition that a current favourites file does not exist. Next the 'setFavourites()' method uses a StreamWriter to append the text file with the name and corresponding URL for each favourites entry, I chose to use .AppendText rather than .WriteLine as the favourites text file is only creates once and the text is then changed each time, there is no option to delete the favourites file through the GUI. Finally the 'getFavourites()' method opens the file and reads all of the lines to a list using .ReadAllLines, it then reverses the list to show the most recent favourite is index position 0 and returns favourites list.

Form1.cs

- This is the largest class in my project, it contains the event handlers for the useable buttons and text boxes, I will highlight methods that need explaining as some of the methods are very simple plain English code.
- goBtn_Click()
    - This method takes a string from the search box which will be a URL, it does a check to ensure the search box is not empty using string.Empty, if it is empty is displays a message box. It then gets a web request with the url as the parameter, calls the HTTP Class and displays the content to the display box.
- CW1_Load()
    - This method is actioned when the form loads, it hides the panels (menus) that are meant to be hidden and sets the home page url as my heriot watt page, it then calls the HTTP class to create a web request with the home page url as the parameter and displays this content to the display box.

- buttonClick()
    - This method is an event hander for the button clicks in the history and favourites menu, it creates an event if the button is clicked and then performs a web request with the content of the button, for history this is the name of the button. This allows the user to click the URL on the history menu and navigate to that web page.
- historyButton_Click()
    - This method is actioned when the history button is clicked in the GUI, it calls the getHistory method of the history class and creates a list with its content, it then uses a for loop to dynamically create buttons up to the number of index's in the list, it sets the name of the button to the index of the list which is a URL.
- backBtn_Click()
    - This class is actioned when the user clicks the back button, it calls the getHistory() method of the History class, it then uses a foreach loop to iterate over the elements of the list searching for the url of the current web page using .FindIndex and .StartsWith, when it finds the url it return its index, it then sets the next url to the index + 1, meaning the previously visited url, finally it created a web request using that new URL, which will navigate the user to the previous web page.

HomePage.cs

- This class deals with the Home Page requirement, it contains two methods. The 'setHomepage()' method creates a new file using FileStream and then uses a StreamWriter to write the URL taken from the Form1 class to the text file, I used .WriteLine rather than append as there is only every one home page, thus I don't need to append the text, I am simply writing a new URL. The 'getHomePage()' method uses a StreamReader to read the lines of the file provided it is not empty and then set the homePageUrl equal to the line on the text file, it then returns the homePageUrl to be used in a HTTP request.

Program.cs

- This is the main entry point for the program, it contains a main method that starts the application.

Testing

Test	Expected	Actual	Pass
Perform HTTP request and display a 400 error	The remote server returned an error: (400) Bad Request.	The remote server returned an error: (400) Bad Request.	YES
Perform HTTP request and display a 403 error	The remote server returned an error: (403) Forbidden.	The remote server returned an error: (403) Forbidden.	YES
Perform HTTP request and display a 404 error	The remote server returned an error: (404) Not Found.	The remote server returned an error: (404) Not Found.	YES
Refresh Page	A new HTTP Request should be sent, and the page should be refreshed.	A new HTTP Request is sent, and the page is refreshed.	YES
Add Home Page	A new home page should be saved and clicking the home button should display that page.	A new home page is added, clicking the home button then displays this page.	YES
Change Home Page	A new home page should over-write the old home page and clicking the home button should display that new page.	A new home page is created, clicking the home button displays the new page.	YES
Perform HTTP request with empty search bar.	An error should be shown prompting the user to enter a URL.	Message box shows “Please enter a URL in the search box”	YES
Click Back button	A new HTTP request should be sent with the previous back URL.	The previous page is displayed.	YES
Click Forward Button	A new HTTP request should be sent with the previous forward URL.	The previous page is displayed.	YES
Click Close Button	The Menu containing the close button should be set to Visible = False.	The menu disappears	YES
Delete History	The history file should be over-written to be blank and the history file should no longer contain any elements.	The history file is now blank.	YES

Conclusions

I am very proud of the work I have produced during this project, I have improved my programming knowledge more in this coursework than I have in any task I have completed thus far in university, I am very glad I switched to this course in week 3. I have learned I need to improve my time management skills as unfortunately I underestimated the number of hours I would spend on certain tasks and thus did not get to implement the bulk download feature; I also feel I could of spent more time making my code more efficient and implementing some more advanced language techniques. Overall, I am very happy with the work I have produced and more importantly I knowledge I have now gained.

References

1 – Microsoft documentation: https://docs.microsoft.com/en-us/

2 – Han Wolfgang Loidl Lecture Material