DEVELOPING A SIMPLE WEB BROWSER

Assessed Coursework 1 – Industrial Programming (F21SC)

Abstract

The aim of this coursework was to develop a simple web browser, a windows based application using the object-oriented languages C#.

Georgios Karagiannis
gk13@hw.ac.uk
H00226912

Table of Contents

1. Introduction

The aim of this coursework was to develop a simple web browser, a Windows based application using the object-oriented languages C#.

In order to implement this coursework I used the Visual Studio 2015. My application is in form and can be installed and executed on any Windows 7 or above system.

The web browser should send HTTP request messages for URLs typed by the user and also should receiving HTTP response messages and display the answers (the HTML code) of the messages on the interface. Furthermore, if the response status is not 200 (everything are OK) the web browser should provide the error code and a comment about the error (400: Bad Request – 403: Forbidden – 404: Not Found).

Also, the user should be able to setup a homepage and when the browser open should open that page. The user should save a URL as favorite and he should be able to edit it, to change the favorite's name and if he does not want it any more he should be able to delete it and also to use it in order to visit the page.

The browser must keep the website's URL that the user visited, the history, and the user should be able to navigate into the history's URL, he should be able to go BACK in order to visit the previews website or to go FRONT in order to visit again the same website. Finally, clicking directly in a history's URL he should visit that website.

And finally the web browser should be designed in such a way that the communication is separated from the GUI and by supporting tab and all that using different threads.

The web browser has to have graphic user interface and it has to be friendly to users, I had to use forms, menus and button to increase accessibility.

MS Visual Studio for C# provide us with so much useful stuff that's well-implemented and easy to use. Writing in C# gives us access to all the .NET Framework class libraries, which are quite extensive. I order to implement the web browser I used some them.

Firstly I will discuss which of the requirements I have delivered and which I have manage to implement. After that I will provide the basic design of my code. Furthermore I will provide a user guide and a developer guide. Finally I will show the results of the test I have done and how C# features helped me to implement the application.

2. Requirements' checklist

1	Send HTTP request messages	Works
2	Receive HTTP response messages	Works
3	display the contents of the messages on the interface	Works
4	display HTTP response status error codes	Works
5	User creates home page	Works
6	User edits home page	Works
7	Home page loaded on the browser's start up	Works
8	User adds a URL to a list of favorites	Works
9	User associates a name with each favorite URL	Works
10	User can modify a favorite item	Works
11	User can delete a favorite item	Works
12	User is be able to request a favorite web page by clicking its name on the Favorites list	Works
13	On the browser's start up, the Favorites list loaded to the browser	Works
14	The browser maintains a list of URLs, corresponding to the web pages requested by the user	Works
15	The user should is able to navigate to previous and next pages, and jump to a page by clicking on the links in the History list	Does not work
16	On the browser's start up, the History list is loaded to the browser	Works
17	The web browser is designed in such a way that the browser-server communication is separated from the GUI support	Works
18	The web browser supports separate Tabs	Works
19	I provide one thread for each Tab	Does not work
20	Each Tab contains information about its local state	Works

My program is able to sent HTTP request messages and to receive the responses, it can handle the most common errors (404, 403, 400) and can display the content (the HTML code of the website). The browser is simple and easy to use because has a GUI in order to perform all the operation, it has button to increase accessibility.

The user can set a homepage and to change it any time he wants, also he can visit the home page anytime he wants and finally the homepage is loaded on the browser's start up.

The browser has a list of favorites and the user is able to add a URL into this list, he is also able to associate a name with each favorite and he can change the name in an earlier time and also is able to delete the URL from the list. Of curse he can visit any URL is in favorite list. Every time the browser starts, the favorite list loads to the browser.

The browser keeps the URL the user visited in a list an the user is able to see that list. Unfortunately the user can not navigate to previous and next pages or to jump to a page by clicking on a URL in the history list

The browser-server communication is separated from the GUI, this communication is provided every time the user wants to visit a website and is called from an other class.

The browser supports Tabs and the user is able to have several websites open. Every user's Tab is a separate Task and every Tab has the website's code.

3. Design Considerations

I will explain how I chose to design the code, how I chose to store my program's data and which classes I build. I will also illustrate a UML diagram of my program

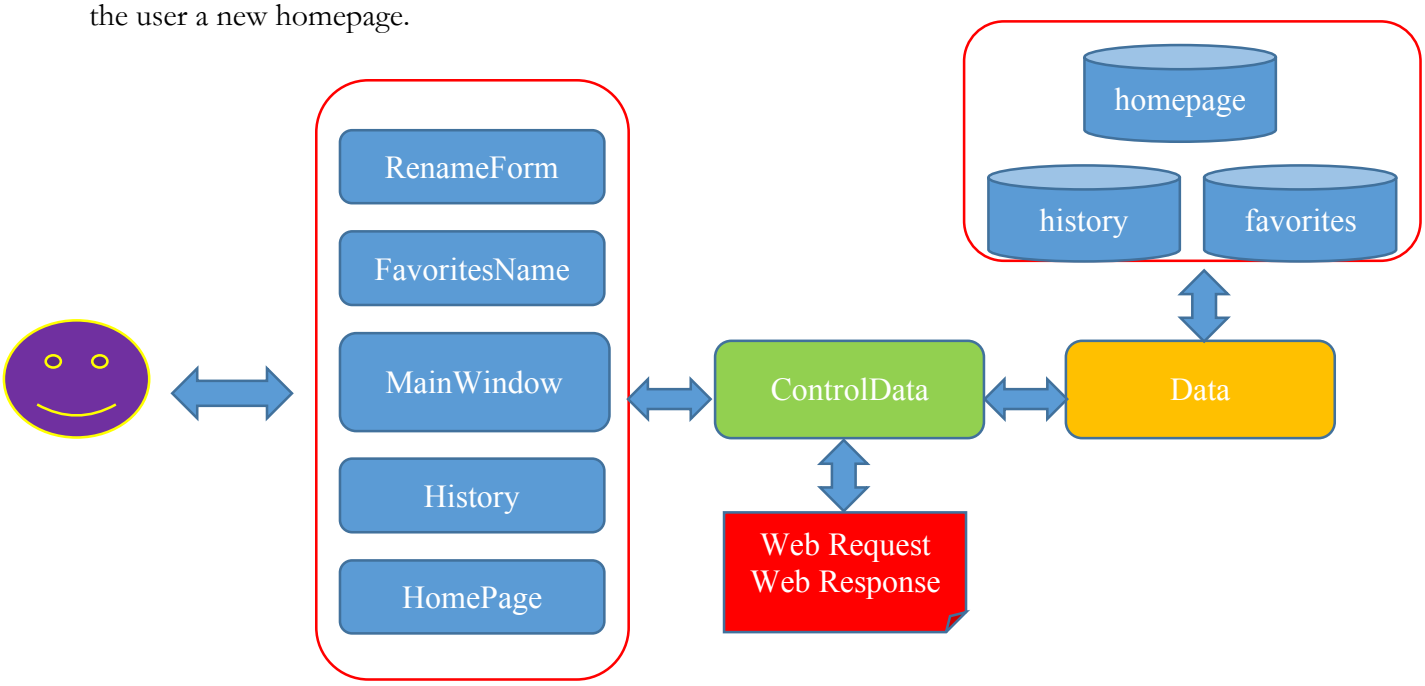
For the implementation take some ideas from Model–view–controller (MVC) architectural pattern.

I have divide the application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

I decide to implement two main classes, the class Data and the class ControlData

- My Model is the Class Data. In this class I have all my methods in order to store my Favorites, my History and my HomePage .
- My Controller is the class ControlData which manipulates my Data.
- My View is the class MainWindow which generates an output presentation to the user based on Model

I deside to implemet 5 forms. The main form is the MainWindow, is the one which opens when the user starts the application. The FavoritesName is the form in which the user writes the name of the site he deside to add to his favorites. The RenameForm is the form that opens when the user will deside to change the name of a favorite site. The History form is the form that shows the list with the URL the user visited and finally the HomePage form which opens in order to set the user a new homepage.



Furthermore, I decide to store my data in three files. The first one is the file homepage.txt, is a plain text file because I had to store only a URL and one .txt file is suitable from that. I store my program's favorites in a XML file, for any favorite have to keep the URL and the name that the user chose. I used one XML file in order to have any favorite as a node. Furthermore I should be able to search the XML file in order to delete a node (one website from my favorites) or to edit a node (to change the name). Finally I store the list of the URLs the user has visit in the past in a .txt file. By browser does not support navigation to the site I have previews visit so I chose a .txt file for that.

Writing crucial data to the disk as text is sometimes dangerous. Any anonymous user can open the text file and easily read my data. With Object Serialization, you I could reduce this danger to a certain extent. I could have a Dictionary <string> <string> in order to store my favorites, an ArrayList <string> for my history and just a string for the homepage. All that could be in the class Data and I could write the object directly to a filestream without converting values of individual properties into a text.

.NET framework provide us to do Serialization of Custom-build classes with my own properties and methods. And then comes inheritance and all others. Serialization is the process of converting complex objects into stream of bytes for storage. Deserialization is its reverse process, that is unpacking stream of bytes to their original form. The namespace which is used to read and write files is System.IO.

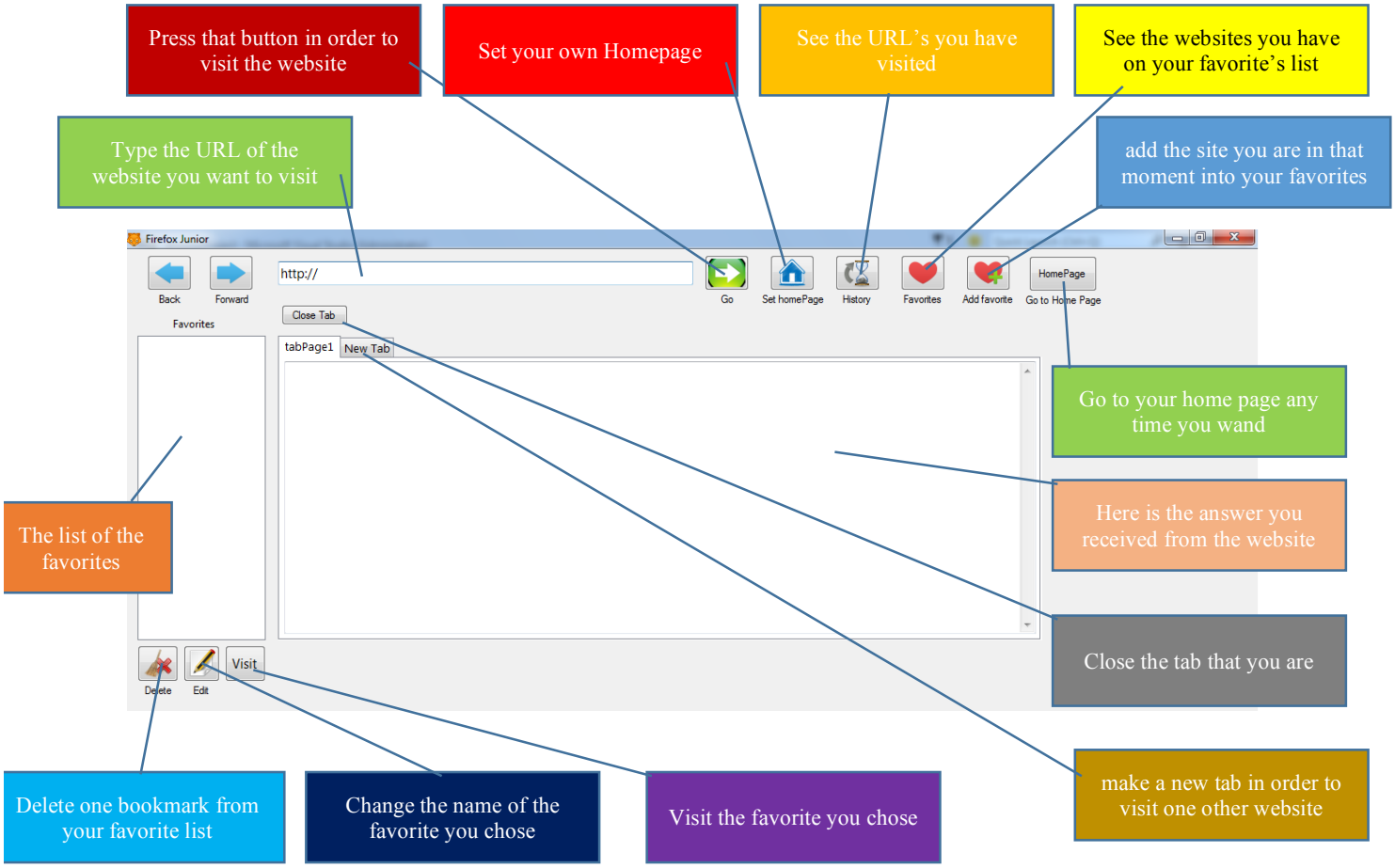
I tried to make an easy to use GUI in order to use the application any typical user. The application has only button for every operation.

I used many classes from .NET Framework in order to implement the application. BrowserRequests and HttpWebRequest are the most important for my application's communication.

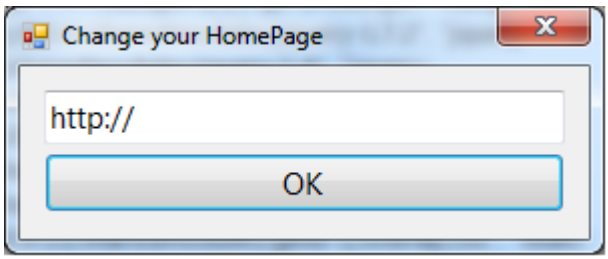
4. User Guide

I will provide a user guide which helps users to operates my application.

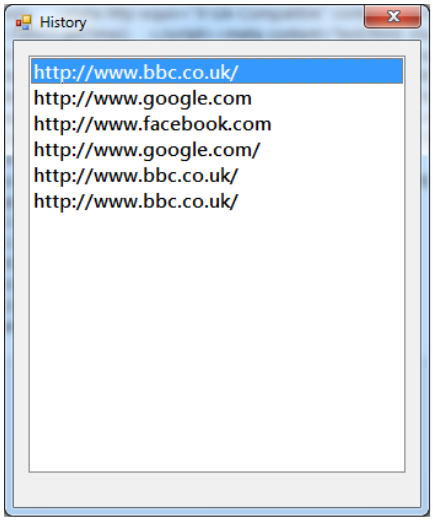
That is the main windows of my application



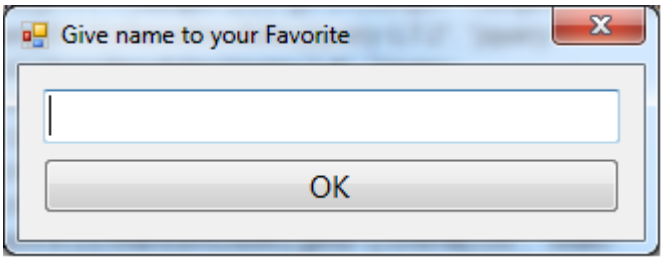
- If the user wants to visit a website should write the URL in the address bar and to press the **GO button**.
- In order to choose a homepage the user should press the **Set homepage** button. The following window will appear and the user should type the URL of the home page.

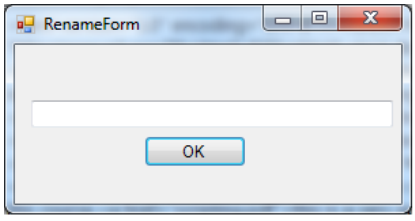


- If the user wants to see the browser's history he should press the button **History.** The following window will appear with the history.

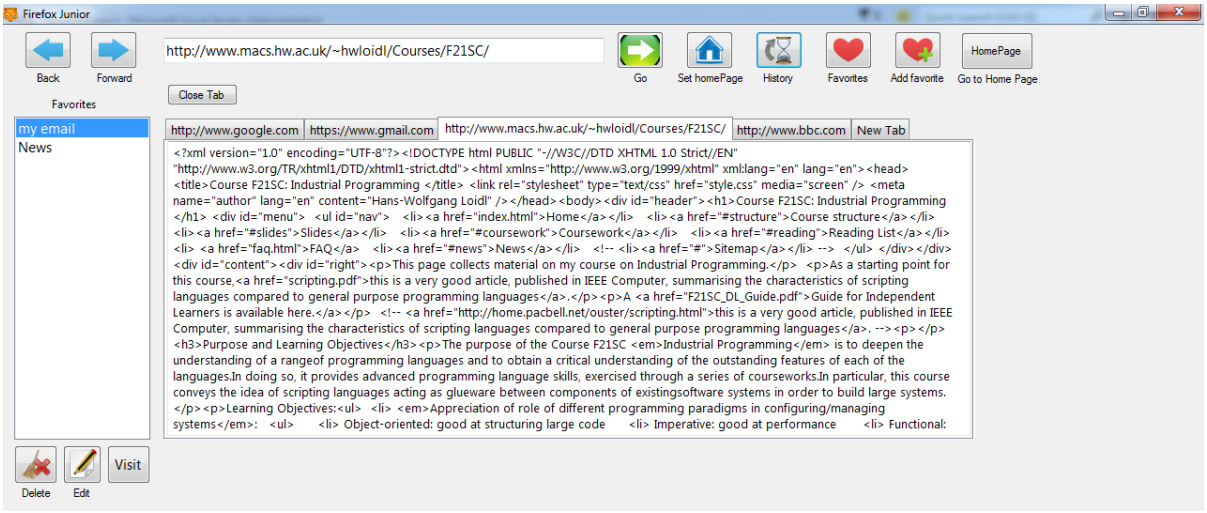


- If the user wants to see the list with his favorites websites he should press the button **Favorites** and in the left box he will see the names of his favorites websites.
- If the user wants to add a URL to his favorites should press the button **Add to Favorite**. The following window will appear and the user should type the name of the favorite.



- When the user wants to visit his home page should press the button **homepage**
- If the user wants to delete a favorite website firstly he should press the button **Favorites** in order to see the favorites. After that he should select it from the list and press the button **delete**.
- If the user wants to visit a favorite website firstly he should press the button **Favorites** in order to see the favorites. After that he should select it from the list and press the button **Visit**.
- If the user wants to rename a favorite website firstly he should press the button **Favorites** in order to see the favorites. After that he should select it from the list and press the button **Edit**. The following window will appear and the user should type the new name of the favorite
- 

This is the main windows of web browser when the someone uses it



5. Developer Guide

Firstly I will describe my application design and I will explain every class I have build, every method I implement and how my methods works. Secondly I will explain my forms code and how the forms uses my classes' methods.

Class Data

The "Favorites" part

The class Data has a method AddFavorite which takes as arguments two strings, 1) the name that the user chose to give for the website and 2) the URL that is in the URL address bar in that time. I used the class XmlDocument which represents an XML document.
When the user wants to store a URL into to his favorites, this method creates a new element in the XML file, the attribute is the URL and the text is the name that he chose.

The XML file is like that:

<favorites>

<favorit url="**http://www.in.gr/**">News</favorit>

<favorit url="**https://www.gmail.com**">My email</favorit>

</favorites>

If there in not other websites in the favorites, I have to set the root in the XML file (<favorites>) and after the root I append the child (<favorit url="**http://www.xxxx.xxx/**">XXXXXX</favorit>)

If there is already the file Favorites.xml I just append the element as child.

At the end I save the XML file.

The method RemoveFavourite takes as argument the string that the user chose to select in the list of the favorites, basically takes as argument the name of the website that user chose to delete from the favorites. After that I have to search every node in the XML file that I keep the favorites until to find the element which has the same name that user select and I delete the elements. In the end I save the XML file.

The method RemoveFavourite returns a list with the name of the favorites in order to show that list in the main window. I read very node of the XML file and I store every value in a list of strings and I return that list.

The method RenameFavorite takes as argument two strings, the name of the favorite that the user chose to rename and the new name the user wrote into the form RenameForm. It search every node in the XML file in order to find the node which the user chose and I replace that name with the new name.

The method SearchFavorite takes as argument a string, the name of the site in the list of favorites that the user chose to visit, I search every node in the favorites XML file in order to find the node with the value that user chose and I return the attribute of that node which is the URL.

The "History" part

I store the web browser's history into a ".txt " file, I serialize the history using the StreamWriter class. When the user asks to see the browser's history I have the GetHistory method in which I declare a new List<string>, I read the file history.txt and I add every line from the file as a element in the list.

The "home page" part

I store the home page into a plain text file, I serialize it using the StreamWriter class and I return the homepage on the browser load using the method GetHomePage(). This method reads the file home.txt, it makes it a string and return the string

Class ControlData

Only this class communicates with the main web browser's windows and basically with the user and with the GUI. First of all, I have this class in order to control my data, to handle my stored data (history, favorites, homepage), basically in order have access to my class Data methods and to send the results to GUI.

First of all I make a new private object Date type in order to have access to Data's methods.

I have the method AddFavoriteControl in order to store a website to my favorites. That method just uses the Data's method AddFavorite.

The method RemoveFavouriteControl also uses the Data's method RemoveFavourite.

The method SearchFavoriteControl used the Data's method SearchFavorite.

Same with the methods SetHistoryControl and GetHistoryControl, they just use the Data's methods.

Moving on to the home page control part, I also have two methods which uses the Data's methods in order to set a new homepage and to return the home page for the program's load.

The most important method in this class is the method BrowserRequests. The .NET Framework uses specific classes to provide the three pieces of information required to access Internet resources through a request/response model: the Uri class, which contains the URI of the Internet resource you are seeking; the HttpWebRequest class, which contains a request for the resource; and the HttpWebResponse class, which provides a container for the incoming response. System.Net namespace provides HttpWebRequest and HttpWebResponse classes, Sytem.IO namespace provides classes to send request and receive response in streams.

I create a new object HttpWebRequest type which takes the the web request using the WebRequest class and the GET method. After that I use the HttpWebResponse class in order to take the response and I store the response into a string variable using StreamReader and I return that string which is basically the HTML code of the website. If the is a problem with the request I check the exception and especially the status code of the error. In that case I return to the user a message based on the error code.

The last part of the class ControlData is the Task part, I use Task<string> in order to have multiple tabs and basically many websites opened at the same time, every calls my BrowserRequests method in order to keep the string with the response as I analyzed previously and using the method Page I return the HTML or the error code with the my message.

Form MainWindow

That form is the main browser's window, my main form.

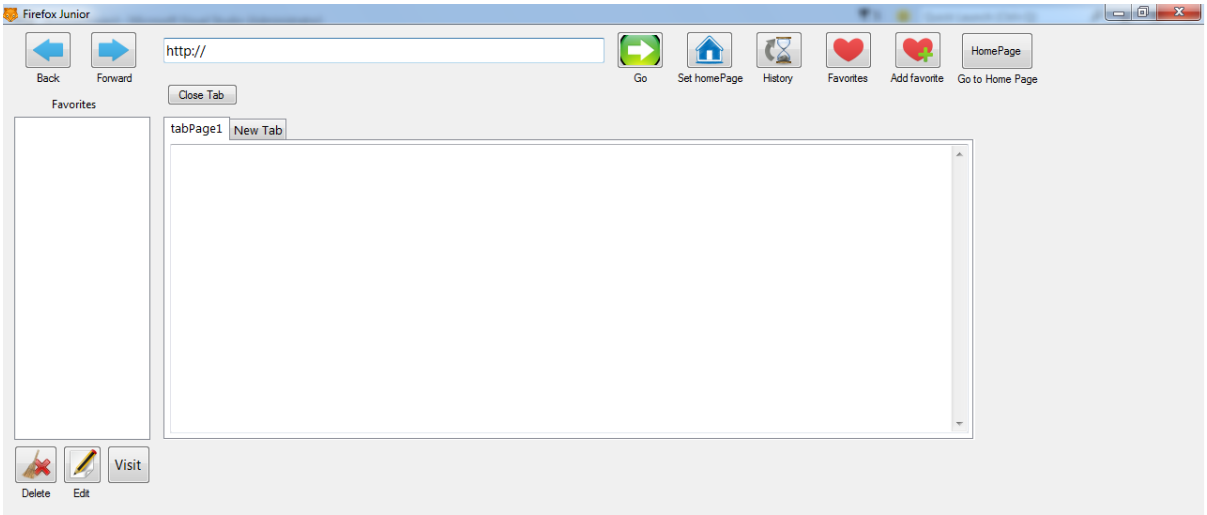
Starting I declare a news object from the class ControlData in order to have access to my data and to sent requests. After that I declare the other forms.

Constructor

When the form initialize constructor makes a new TabPage object and a new multiline TextBox with scroll bar object in order to provide in this tab the web server response.

After that I check if the homepage.txt file exists in order to call the ControlData's method to open the file and to return the URL. I put that URL in the address bar and I call the browser to

sent a request. If the history.txt exist I keep that URL in the browser's history and I load the list of favorites if the .xml file exist.



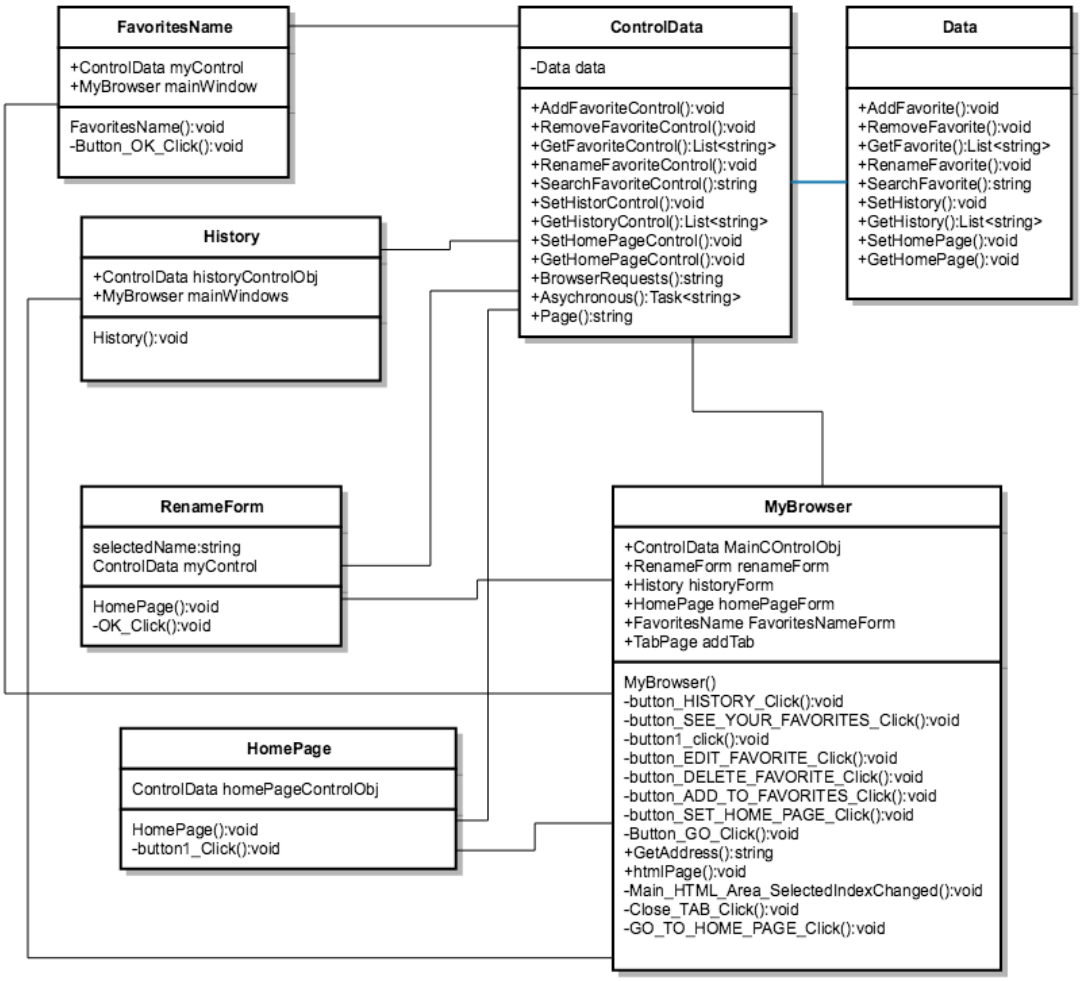
The buttons

- The buttons **Back** and **Forward** does not work.
- The button **History** calls and opens the ***form History***. This form communicates with the class ControlData and illustrates in a ListBox the full web browser's history. Basically, uses the GetHistoryControl method in order to illustrate the history. I explained previously how the GetHistoryControl works.
- The button **Favorites** calls the GetFavouriteControl in order to illustrates in the ListBox "Favorites".
- The button **add favorite** calls and opens the ***form FavoritesName***. This form communicates with the class ControlData and uses the method AddFavoriteControl in order to store the URL which is in the address bar (takes the URL using the main's form method GetAddress which just returns the address bar text) and the text that the user writes in the form.
- The button **Set homepage** calls and opens the ***form HomePage.*** This form communicates with the class ControlData and uses the method SetHomePageControl in order to store in the homepage.txt the URL that the user wrote in the textbox.
- The button **Delete** uses the method RemoveFavouriteControl in order to delete the favorite that the user chose. After that reloads the favorites list using the method RemoveFavouriteControl. If there is nothing to delete then shows a message.
- The button **Edit** calls and opens the ***form RenameForm*** in order to change the name of the favorite the user chose. In this form I call the ControlData's method RenameFavoriteControl and the text that the user will write in the textbox will replace the favorite's name.
- The buttons who sends requests
    - The button **GO** uses the text from the address bar, that string is transformed to Uri object using the .NET Uri class and makes a new Task. Uses the method BrowserRequests takes the response and using the method htmlPage illustrates into selected tab the response as string. Furthermore, the text in the tab is the URL and the URL stored into the history file.
    - The button **HomePage** (go to home page) makes the same as the button GO but as URL takes the string which is in the homepage file.

- The button **Visit** makes the same as the button GO but in the address bar is the URL of the favorite the user chose. In order to implement that I use the method SearchFavoriteControl which return to the address textbox the URL.

- The button **Close Tab** remove the tab page the user chose using the RemoveAt method with argument the tab the user is.

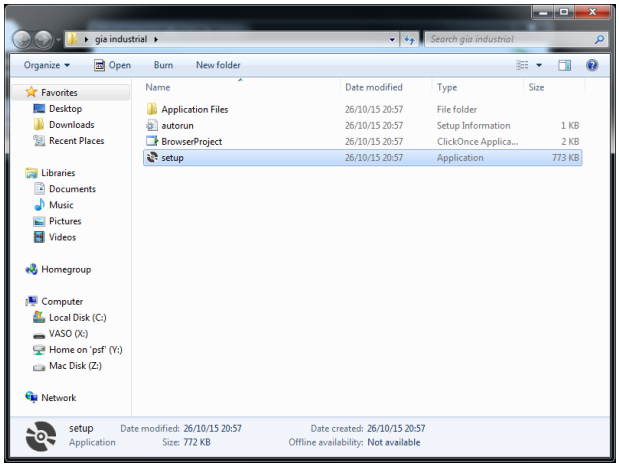
UML Class Diagram

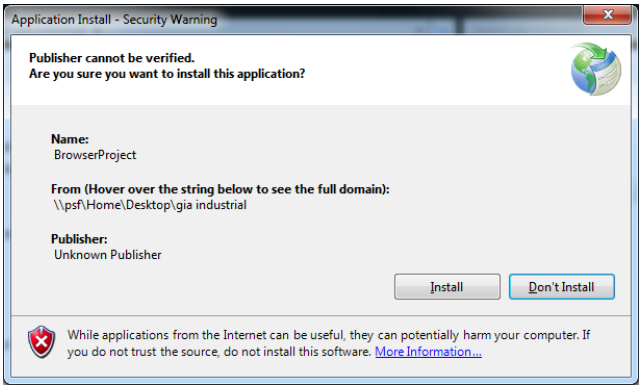


6. Testing

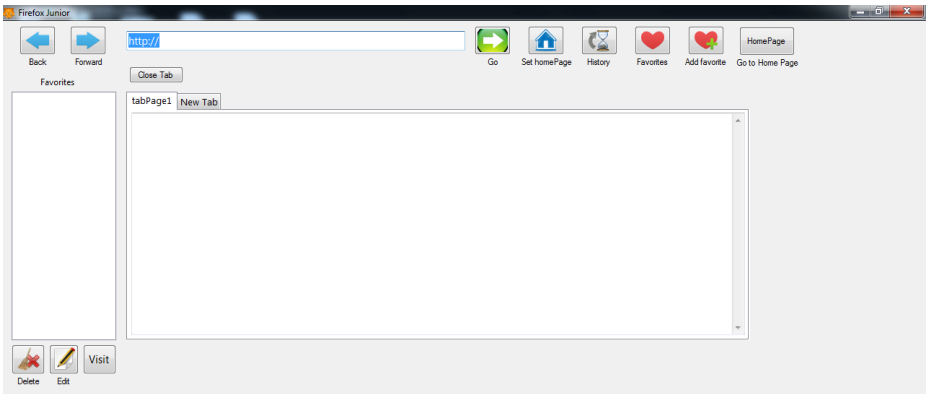
I will show the results of my testing.

- First off all I exported my application making an .exe file in order to install it in my computer and I start the installing

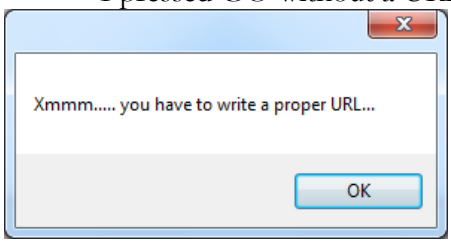




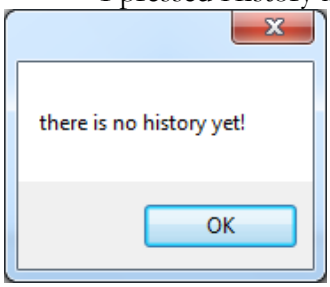
- After that I saw the main window empty because my browser had not home page and favorites.



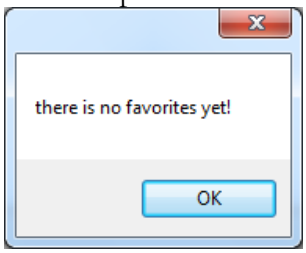
- I pressed GO without a URL in the address bar and I received the message



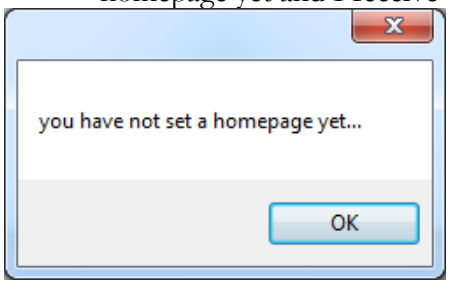
- I pressed History knowing that there is no history yet and I received the message



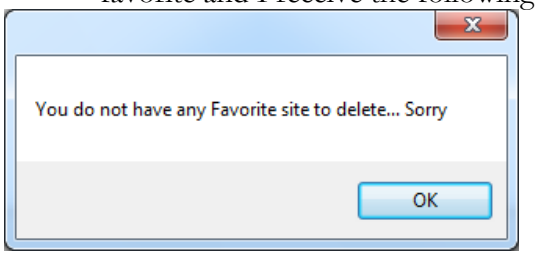
- I pressed Favorites knowing that there is no favorites yet and I received the message



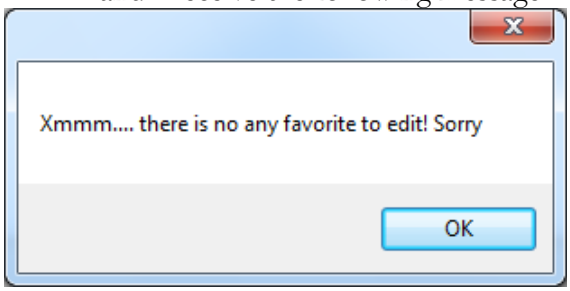
- I pressed HomePage in order to go to my home page knowing that no one had set a homepage yet and I receive the following message



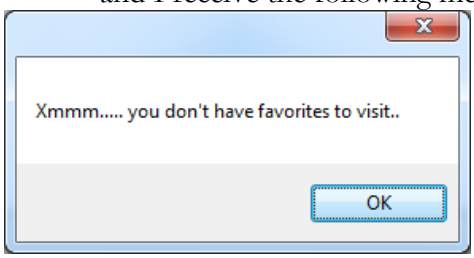
- I press Delete in order to delete something from my favorites without having any favorite and I receive the following message



- I press Edit in order to rename something from my favorites without having any favorite and I receive the following message



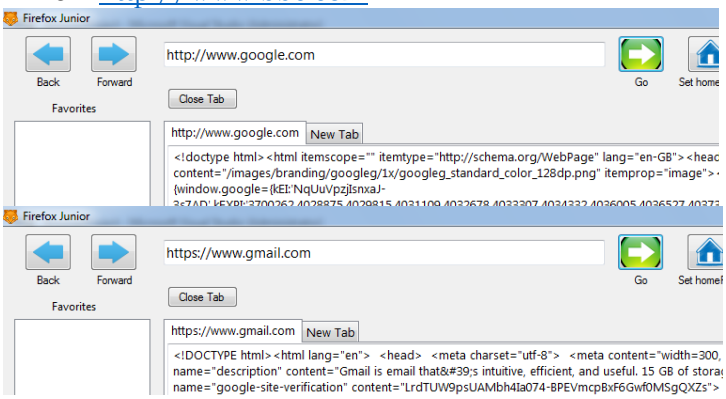
- I press Visit in order to visit a something from my favorites without having any favorite and I receive the following message



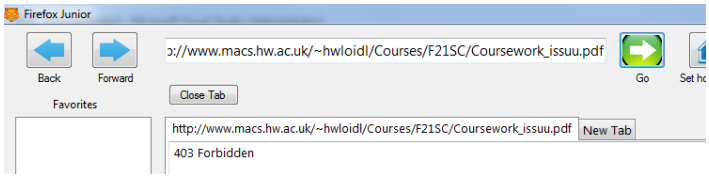
- I already have mention that the buttons Back and Forward does not work

HTTP requests

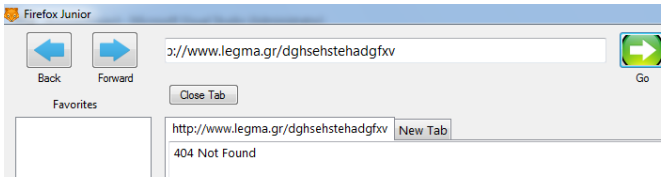
- I check the http requests using plenty of URL's, I also check if a take response using https URL.
    - http://www.google.com
    - https://www.gmail.com
    - http://www.bbc.com



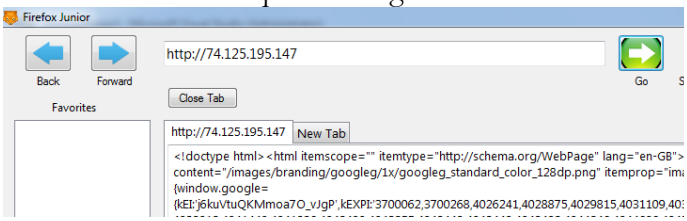
- I check about the status error codes
    - The url: http://www.macs.hw.ac.uk/~hwloidl/Courses/F21SC/Coursework_issuu.pdf Illustrates me the text "403 Forbidden"



    - The url: http://www.legma.gr/fdghsthethadrgaetg illustrates me the text "404 Not Found".

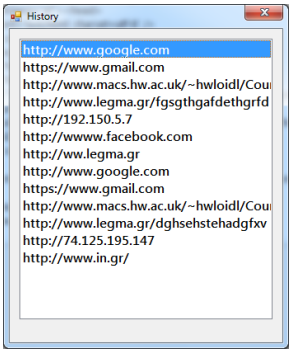


    - I could not check the status error 400.
- I check if it sends requests using IP address

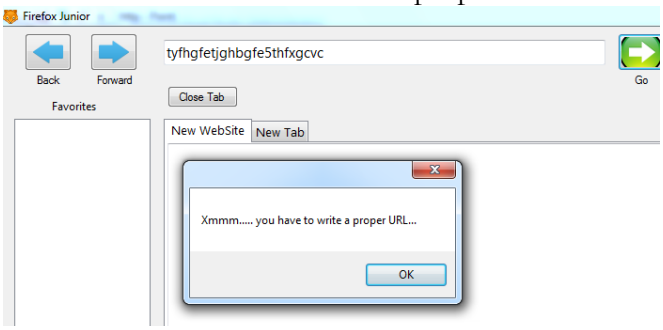


- I check if loads the home page I have set at the program's load

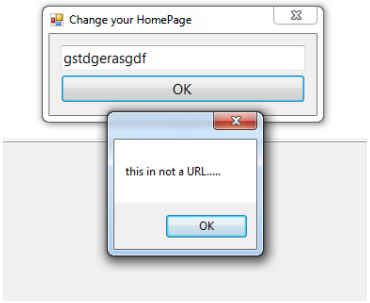
- I check if I can see the history



- I check if I can save a URL to favorites, if I can delete it from the favorites, if I can rename it and if I can visit a URL from the favorites.
- I check if I can have multiple tabs and if I can close tabs.
- I check if a user does not write a proper URL address



- I check if a user writes a not proper URL in order to set a homepage



Errors I found out

- You are able to save to favorites just the http://
- You are able to set as homepage something like that http://dgergerggafweed

7. Reflection on programming language and implementation

The C# language is intended to be a simple, modern, general-purpose, object-oriented programming language.

The language, and implementations thereof, should provide support for software engineering principles such as strong type checking, array bounds checking, detection of attempts to use uninitialized variables, and automatic garbage collection. Software robustness, durability, and programmer productivity are important. The IDE that I used for the implementation was the Microsoft Visual Studio 2015 which helped me a lot. The use of the IDE enables the developer to create a highly functional and efficient GUIs and class structures as the IDEs provide implemented libraries that can be reused.

C# would still would be an excellent choice for the development of this project because it facilitates the use of highly comprehensive libraries of codes. The use of visual studio also creates a rich development environment for the implementation of the project.

Is found extremely helpful the MSDN, I found many things about the .NET Framework, the classes it provides and the usage of them.

In order to implement the application:

- I used the object oriented features of C#, I had to design the and to implement my own classes and my lows methods. I also used a lot of the .NET libraries and C#
- I used the XmlDocument and the XmlElement, XmlTextReader, XmlNodeType in order to handle XML files
- I used “foreach” in order to search in XML files
- I used List<string> in order to store data
- I used Stream and StreamWriter, StreamReader in order to handle files
- I used the class Uri which handls URLs
- I used HttpWebRequest, HttpWebResponse and WebException in order to have communication
- I used forms to make the GUI
- In the forms I used textbox, buttons, TabPage,
- I used Exception to handle possible errors

I had not a all experience with C#, I think that it is not a very easy to learn programming language although it has a lot of common with Java and C++ and many convenience features.

8. Conclusion

C# is a modern and innovative programming language that carefully incorporates features found in the most common industry and research languages.

To summarize, in this project the most important thing for me was the design of the application. The design helped me to implement the application focusing in the C# and .NET features. I tried to have a good class structure and data.

It was the first application I have ever implement using Visual Studio and C#. It was a challenge for me and I had the opportunity to learn more about the technological advances .NET provides.

I had to learn how to use features like:

- Arrays
- Constructors and Destructors
- Indexers
- Properties
- Passing Parameters
- XML Documentation
- GUI
- Serialization

It would be better if the BrowsersRequests method would be implemented as an independent class. The reason of this is because with this way, each of the method could use a different thread and that thread will be not part of the GUI. The multithreading would be independent from all main classes.

This feature would have offered the user the ability to simultaneously load a slow loading process and use the other functionalities which is provided within the web browser. These functionalities would include opening a new tab, loading a new website while the slow loading process is running.

The button Go, Forward and the navigation into the history does not work, I could keep my history in an ArrayList in a class and I could serialize the class when the program closes and deseriazide the class when the program opens. Using that I could have navigation.

For the data storage I could have a Dictionary <string> <string> in order to store my favorites, an ArrayList <string> for my history and just a string for the homepage. All that could be in the class Data and I could write the object directly to a filestream without converting values of individual properties into a text.

I tried to make an easy to use GUI in order to use the application any typical user. The application has only button for every operation.

I tried to make the application as good as possible and also I tried to understand in deep the C#.

Appendix

The following website helped me to implement the coursework

- www.macs.hw.ac.uk/~hwloidl/Courses/F21SC/index.html
- msdn.microsoft.com
- stackoverflow.com