

SYSTEMS PROGRAMMING AND SCRIPTING

FLORIAN BERGMANN
PERSON ID: H00020398

Assessment One: Stock Manager

CONTENTS

I DEVELOPMENT OF A BASIC WEB BROWSER	1
1 INTRODUCTION	2
1.1 Document overview	2
1.2 Remit	2
2 REQUIREMENT'S CHECKLIST	4
3 DESIGN CONSIDERATIONS	5
3.1 Architectural overview	5
3.2 Separate change	7
3.3 Loose coupling	7
3.4 Threading	7
4 USER GUIDE	9
4.1 Requesting a web-page	9
5 DEVELOPER GUIDE	12
5.1 Implementation of the view	12
5.2 Implementation of the requirements	12
6 TESTING	13
7 CONCLUSIONS	14
II APPENDIX	15
A APPENDIX: SOURCE CODE	16
BIBLIOGRAPHY	17

LIST OF FIGURES

Figure 1	model view presenter (MVP)-Pattern	5
Figure 2	Classes of the logic project.	6
Figure 3	The application's main window.	9
Figure 4	Error message due to invalid url.	10
Figure 5	Browser displaying the current web-page.	10

LIST OF TABLES

LISTINGS

ACRONYMS

AR	Additional-Requirements
FR	Favourite-Requirements
GR	GUI-Requirements
GUI	graphical user interface
HPR	Homepage-Requirements
HR	History-Requirements
HTML	HyperText Markup Language

HTTP	Hypertext Transfer Protocol
MVP	model view presenter
PR	Printing-Requirements
URL	Uniform Resource Locator
WR	Web-Requirements

Part I

DEVELOPMENT OF A BASIC WEB BROWSER

INTRODUCTION

This chapter will provide an overview over the document and recapture the specified requirements that were requested.

1.1 DOCUMENT OVERVIEW

The document is divided into seven chapters that will describe different aspects of the developed program:

[Chapter 1](#) provides an overview over the document and the specified requirements, alongside certain assumptions that were made during the development.

[Chapter 2](#) will give a short outline of the requirements that were fulfilled, as well as mention added functionality that was not requested.

[Chapter 3](#) will provide a high level overview over the system's architecture and its sub modules.

[Chapter 4](#) is a short user guide that will describe the usage of the program by leading the reader through a selected choice of use cases to accomplish certain tasks.

[Chapter 5](#) will be based upon [Chapter 3](#) and provide an in-depth explanation of certain details of the implementation.

[Chapter 6](#) will outline how testing was performed and which cases have been covered.

[Chapter 7](#) will provide a reflection of the development process and the program and highlight areas of interest from the developer's point-of-view.

1.2 REMIT

The remit will summarize the requirements provided in the document *Systems Programming & Scripting (2010/2011) Assessment Two* and list all assumptions made in respect to a certain requirement.

For later reference in [Chapter 2](#) the requirements will be divided into seven groups: Web-Requirements ([WR](#)), Homepage-Requirements ([HPR](#)), Favourite-Requirements ([FR](#)), History-Requirements ([HR](#)), Printing-Requirements ([PR](#)), GUI-Requirements ([GR](#)), Additional-Requirements ([AR](#))

WR01: Send Hypertext Transfer Protocol ([HTTP](#)) request message for typed Uniform Resource Locator ([URL](#))s.

- WR02: Receive HTTP responses for send requests.
- WR03: Display received HyperText Markup Language (HTML)-code.
- WR04: Display HTML for 200, 400, 403 and 404 headers.
- HR01: Allow creation and editing of a homepage-URL.
- HR02: Load homepage on application start-up.
- FR01: Allow adding, deleting and editing of a URL to a list of favourites.
- FR02: Allow the specification of a name for a favourite.
- FR03: Request a favourite's URL when the favourite is activated.
- FR04: Load all favourites on application start-up.
- HR01: All pages that are requested shall be saved in a history.
- HR02: Request a URL when an entry in the history is activated.
- HR03: Load history on application start-up.
- PR01: Allow printing of displayed web page.
- GR01: Provide a graphical user interface (GUI) for the requirements.
- GR02: Use menus, buttons, short cuts to increase *accessibility*¹.
- AR01: Utilize multi-threading to keep application responsive.
- AR02: Allow requesting of multiple web pages simultaneously.

¹ Assumption: accessibility can be enhanced by using a standard layout common in Windows environments.

REQUIREMENT'S CHECKLIST

The following list shall provide an overview over the fulfilled requirements. The numbers correspond to those used in [Chapter 1](#).

WR01: Fulfilled by utilizing the [HTTP](#)WebRequest-classes of .NET.

WR02: Fulfilled by utilizing the [HTTP](#)WebResponse-classes of .NET.

WR03: Fulfilled in the [GUI](#).

WR04: Fulfilled via error-handling.

HR01: Fulfilled as application setting ¹.

HR02: Fulfilled.

FR01: Fulfilled ²

FR02: Fulfilled.

FR03: Fulfilled. [URL](#) will open in current window.

FR04: Fulfilled.

HR01: Fulfilled.

HR02: Fulfilled. [URL](#) will open in current window.

HR03: Fulfilled ³.

PR01: Fulfilled. User can print the [HTML](#)-code of the current window.

GR01: Fulfilled. See [Chapter 4](#).

GR02: Fulfilled. See [Chapter 4](#).

AR01: Fulfilled. See [Chapter 3](#) and [Chapter 5](#) for further information.

AR02: Fulfilled.

¹ A Windows conform standard path based on the user account is chosen to save the homepage and can not be adjusted.

² A Windows conform standard path based on the user account is chosen to save the favourites and can not be adjusted.

³ A Windows conform standard path based on the user account is chosen to save the history and can not be adjusted.

DESIGN CONSIDERATIONS

This chapter provides a general overview of the application's design without explaining implementation details.

Therefore it will provide a general overview of the application's architecture and point out areas of interest in the design: these areas will describe the reasons the current approach has been chosen. If appropriate, the design patterns that were employed will be mentioned and described. Concrete implementation details will later be described in [Chapter 5](#).

3.1 ARCHITECTURAL OVERVIEW

The application was built with the [MVP](#) pattern as a basis for the architecture¹.

This way the program logic can be separated from the display logic. To achieve this, the view passes all calls to the presenter. The presenter routes them to the model if necessary.

All return values will be passed back to the presenter. The presenter will then decide what action the view needs to take according to the received values. These action will be communicated via an interface.

This leads to a **decoupling** between the view and the model and allows the presenter to be reusable across multiple views.

A schematic visualisation of this pattern looks like this:

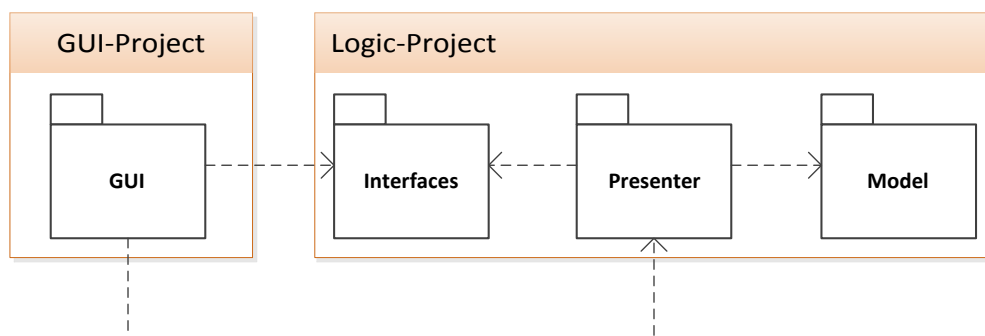


Figure 1: [MVP](#)-Pattern

¹ Further information can be found at [MVP-Pattern \(http://msdn.microsoft.com/en-us/magazine/cc188690.aspx\)](http://msdn.microsoft.com/en-us/magazine/cc188690.aspx)

This already show the division into two distinct projects:

GUI: Hosts the **GUI** implemented in WinForms. This includes all display-related logic: perform actions based on events, display and hide controls, respond to clicks.

LOGIC: Hosts the *business*-logic, the interfaces and the presenters.

As can already be inferred from the short description, most of the application's logic is based in the logic project, which (excluding the view interfaces) encompasses the following classes:

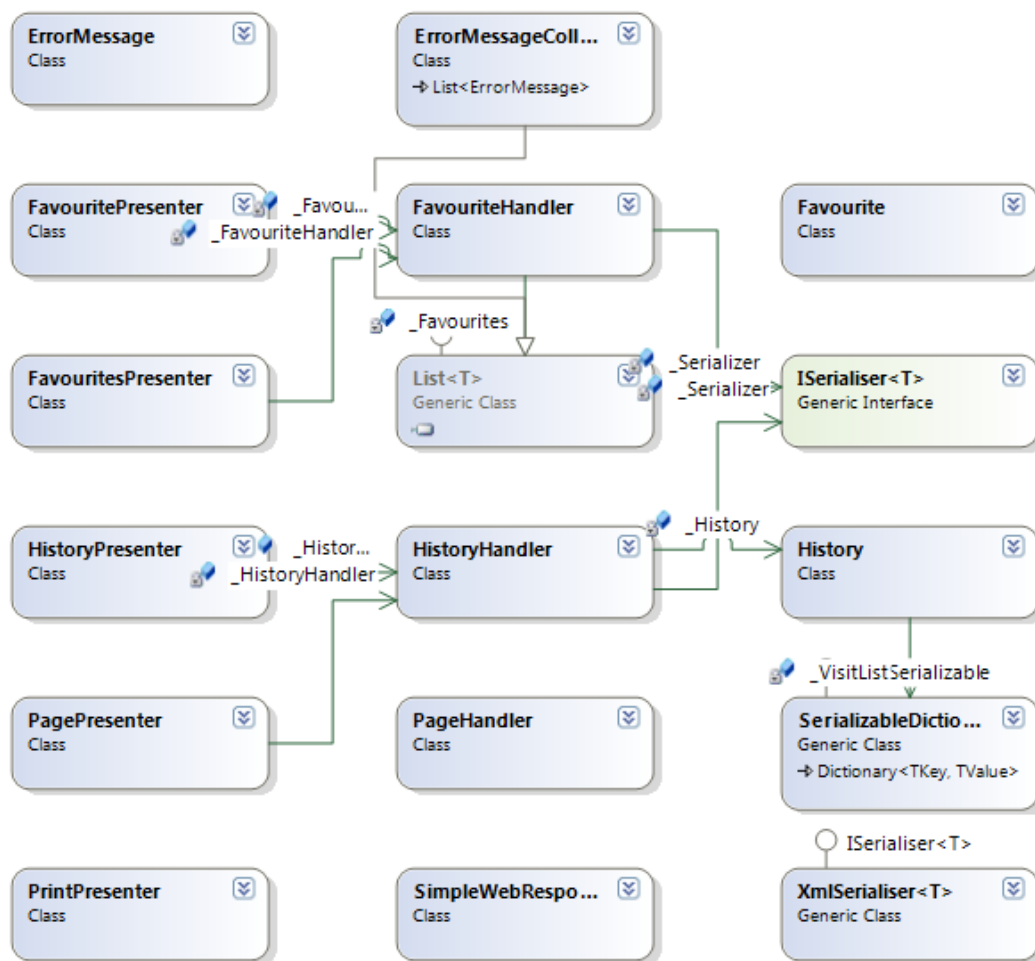


Figure 2: Classes of the logic project.

The following design decisions were made while implementing the application:

3.2 SEPARATE CHANGE

Special attention should be paid to the classes `FavouriteHandler` and `HistoryHandler`. From a design point-of-view, they provide the presenters that are utilizing them with a facade to the data-classes `History` and `Favourite`.

To centralize changes and prevent data loss due to multiple handlers, both objects were implemented via the `Singleton` pattern, ensuring only one object is present at any given time. This way all presenters will modify the same data.

Moreover the handler classes implement an `Observer` pattern to notify the relevant presenter about any changes that occur in the data. This way the presenter can make certain that the view is always displaying accurate data. This was necessary as the `PagePresenter` and the `HistoryPresenter` both operate on the history-data: the `PagePresenter` adds every visited page to the history, whereas the `HistoryPresenter` ensures that the `History` is correctly displayed in the view.

The same principle applies in the case of the `FavouritePresenter` and the `FavouritesPresenter`. The first one adds and edits one single `Favourite`, whereas the second presenter handles the display of multiple `Favourites` and the deletion logic.

3.3 LOOSE COUPLING

To provide a loosely coupled application that may be enhanced with little effort at a later time, the communication between distinct logical groups in the application was implemented via interfaces.

The logical groups (that are not denoted separately in the projects) are:

- View: Realized in a separate project.
- Logic: Realized in the logic project.
- Persistence: Realized in the logic project in the classes `ISerialiser` and the concrete implementations of the interface.

As already shown in the [MVP](#) pattern the presenter \longleftrightarrow view communication is decoupled via interfaces. Moreover the logic \longleftrightarrow persistence communication is decoupled via the `ISerialiser` interface as well.

3.4 THREADING

To fulfil the requirements AR01 and AR02 multi threading was introduced in the application when a web-page is requested.

Therefore the request will be carried out in an asynchronous thread.

As the view should not be concerned about concrete implementation choices, the decision was made, to introduce multi-threading in the **logic!** (logic!) project. It is implemented in the TextPage-presenter. This way the application will not lose one of its core-functions should the customer decide to exchange the GUI.

However, certain changes had to be implemented in the GUI to support multi-threading in a WinForms application (see [Section 5.1](#) for the details): should the GUI be exchanged these problems might need to be taken into account again.

USER GUIDE

The application provides five functions:

- Requesting and displaying web-pages.
- Managing the history of requested web-pages.
- Managing user-defined favourites.
- Managing a user-defined home-page.
- Allow printing of the currently displayed web-page.

These functions shall be explained in this chapter.

4.1 REQUESTING A WEB-PAGE

Upon starting the application the user will see the following window:

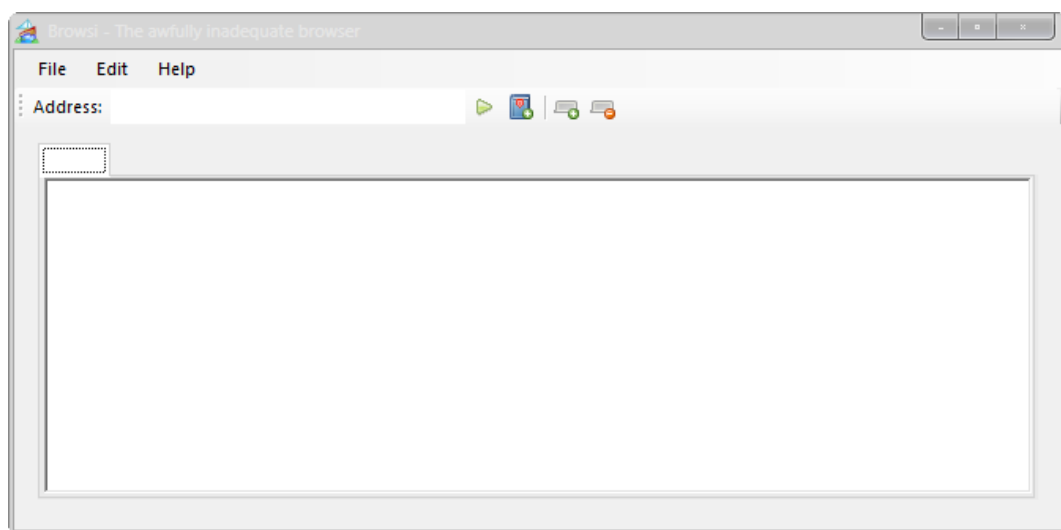



Figure 3: The application's main window.

To request a web-page the user must enter the desired address in the `Address` text-box and then press the Go-button ().

The format of the entered address will be verified. It must match the following pattern¹:

1 `http://[www.]address.domain[/url-path]`

Should the entered address not match the given format, the program will notify the user about the error by displaying a message-box:

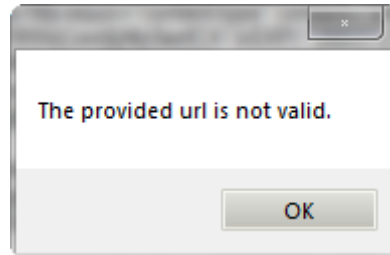


Figure 4: Error message due to invalid url.

When the url is valid the browser will display the [HTML](#)-code in the current tab:

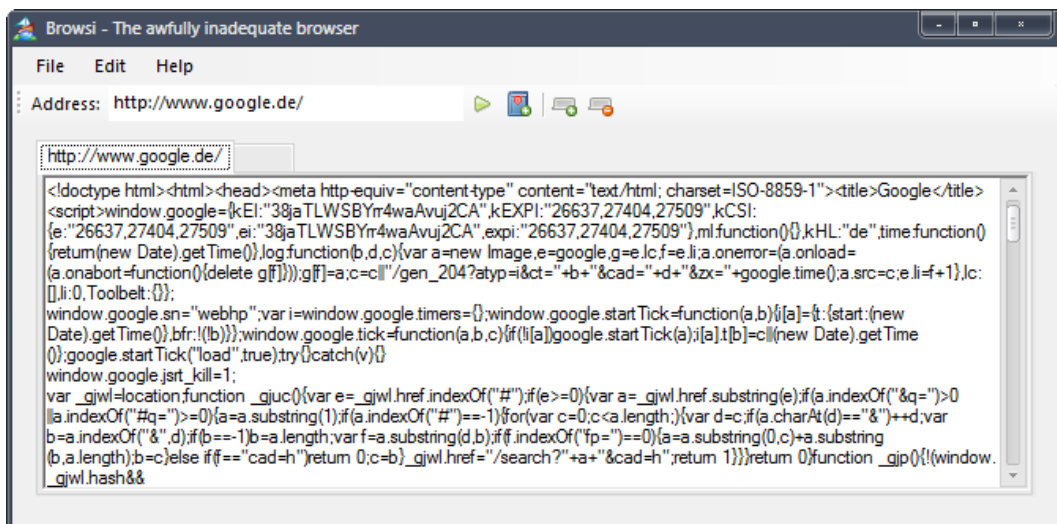



Figure 5: Browser displaying the current web-page.

To request multiple pages simultaneously it is possible to utilize tabs:

To create a new tab, the user simply presses the AddTab-button (📄). This will create new (and empty) tab-page.

¹ Parts in square-brackets may not be required for all web-pages.

Should the user want to close a tab it is necessary to select the tab that should be closed and then press the RemoveTab-button ()

DEVELOPER GUIDE

5.1 IMPLEMENTATION OF THE VIEW

5.2 IMPLEMENTATION OF THE REQUIREMENTS

TESTING

CONCLUSIONS

Part II

APPENDIX

APPENDIX: SOURCE CODE

BIBLIOGRAPHY

- Balzert, Helmut (2009). *Lehrbuch der Software-Technik: Basiskonzepte und Requirements Engineering*. 3. Heidelberg: Spektrum. ISBN: 9783827417053.
- Boodhoo, Jean-Paul (2006). *Design Patterns: Model View Presenter*. English. Microsoft. URL: <http://msdn.microsoft.com/en-us/magazine/cc188690.aspx> (visited on 19/10/2010).
- Dorman, Scott (2010). *Sams Teach Yourself Visual CSharp®2010 in 24 Hours: Complete Starter Kit*. Sams Publishing. ISBN: 978-0-672-33101-5.
- Freeman, Eric and Elisabeth Freeman (2004). *Head First - Design Pattern*. Ed. by Mike Loukides. O'Reilly. ISBN: 0-596-00712-4.
- Noyes, Brian (2006). *Data Binding With Windows Forms 2.0 - Programming Smart Client Data Applications With .NET*. Addison Wesley Professional. ISBN: 978-0-321-26892-1.
- Sommerville, Ian (2006). *Software Engineering*. 8th ed. Addison Wesley. ISBN: 9780321210265.
- Stellman, Andrew and Jennifer Greene (2010). *Head First CSharp*. O'Reilly. ISBN: 978-1-449-38034-2.