COMP 4911 – Final Project

Hayden Walker

April 13, 2024

1 Introduction

My project is a browser for the Gemini protocol. Gemini is a lightweight protocol for serving hypertext, called Gemtext, over the internet.

2 Design

The browser is written in Java, and the GUI is written using JavaFX. The program consists of three packages: `network`, `gemtext`, and `browser`. I used Git to manage the development of the project.

2.1 Packages and Classes

The `network` package contains the `URL` class, which is responsible for storing and parsing Gemini URLs, and the `GeminiRequest` class which takes strings in its constructor for a hostname and a URL and, on instantiation, attempts to open a connection to the server at the hostname and then sends a Gemini request for the document at the provided URL. If the connection fails, it will throw an exception. Otherwise, the `GeminiRequest` object stores the server's status code (as an integer), header message (as a string), and any content sent back by the server (as bytes). These can then be accessed through getter methods. This package also contains exception classes for `URL` and `GeminiRequest`.

The `gemtext` package contains the interface `Gemtext`, which has a single method `render()` that returns a JavaFX `Node` object. All the classes that implement `Gemtext` represent different Gemtext elements, like headings, block quotes, etc. The package also contains the class `GemtextParser` which takes in its constructor an array of bytes (like what would be sent back from a server) and parses it into a list of `Gemtext` objects, deciding which concrete class to instantiate based on the Gemtext specification. The class has a getter method that returns this list.

The `browser` package contains the `Browser` class, which contains the browser itself and is a JavaFX class. This package also contains the `Downloader` class which allows files to be downloaded that can't be displayed in the browser.

2.2 Control Flow

When the browser is launched, the user is presented with an address bar, "Submit," "Back," "Parent Folder," and "Home" buttons, a content pane, and a status bar. When a URL is entered into the address bar and "Submit" is clicked, the browser uses `network.URL` to parse the URL for a hostname, and then creates a new `network.GeminiRequest` object to create the request. If an exception is thrown, then an error message is displayed in the status bar. Otherwise, the successful request's status code and header information are shown in the status bar.

If the server returns status 20, any bytes that are returned by the server are, depending on the header, rendered as Gemtext using `gemtext.GemtextParser`, rendered as plaintext, or downloaded using `network.Downloader`.

Other status codes are handled differently: Status codes 10 and 11 mean that the server expects user input. In this case, a dialogue box appears with a field to enter data, and after it is closed the user's response is sent to the server. Codes 30 and 31 are redirects, and the browser will create a new request to the URL returned by the server in the header field. All other codes represent various errors, so the browser will do nothing except display the code and header in the status bar.

When the "Home" button is clicked, the browser returns to `gemini://gemini.haywalk.ca/browser.gmi`, which is also the page the browser starts on. This page, hosted on my server at home, contains other Gemini links as well as every kind of Gemtext element, for testing purposes. It also contains a link to an image file, used to test downloads.

When the "Back" button is clicked, the browser returns to the last page the user visited. This is achieved using a stack – everytime a new page is visited, its URL is pushed to the stack, and when "Back" is clicked, the most recent URL is popped from the stack and visited.

When the "Parent Folder" button is clicked, the browser moves up a folder.

3 Running the Program

This program requires JavaFX, which must be downloaded separately from the rest of the JDK. The JavaFX binaries are OS-specific and are available at `https://gluonhq.com/products/javafx/`. Once downloaded, the program can be compiled from the `src/` folder using the command:

```
javac --module-path PATH_TO_JAVAFX_FOLDER --add-modules
    javafx.controls,javafx.fxml browser/Browser.java
```

And run from the `src/` folder using the command:

```
java --module-path PATH_TO_JAVAFX_FOLDER --add-modules
    javafx.controls,javafx.fxml browser.Browser
```

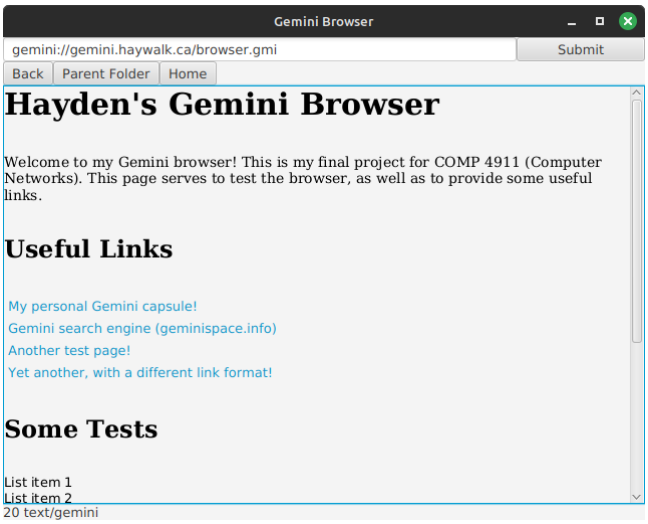

Figure 1: Startup page

4 Screenshots

The following screenshots of the browser running are included in this document:

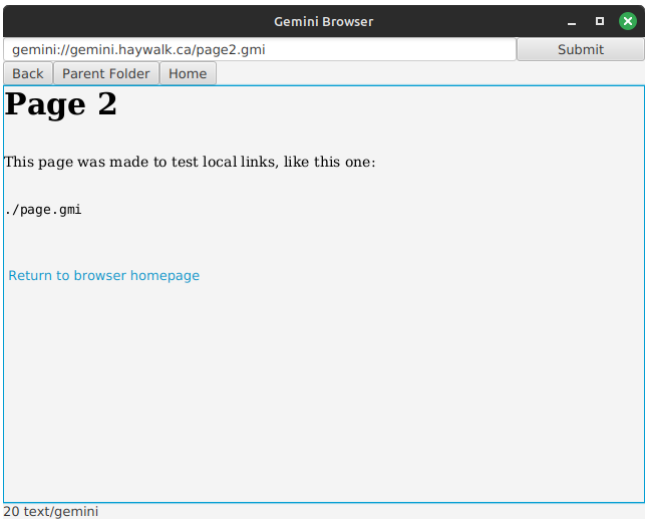

Figure 2: Another page

Figure 3: Displaying plaintext

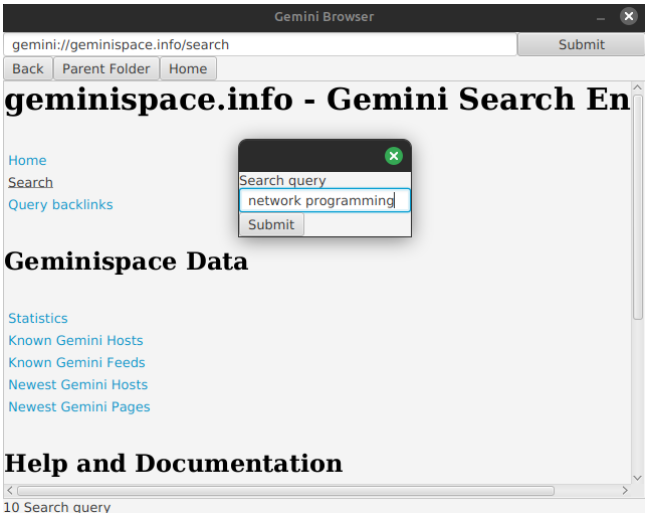

Figure 4: Entering a search query

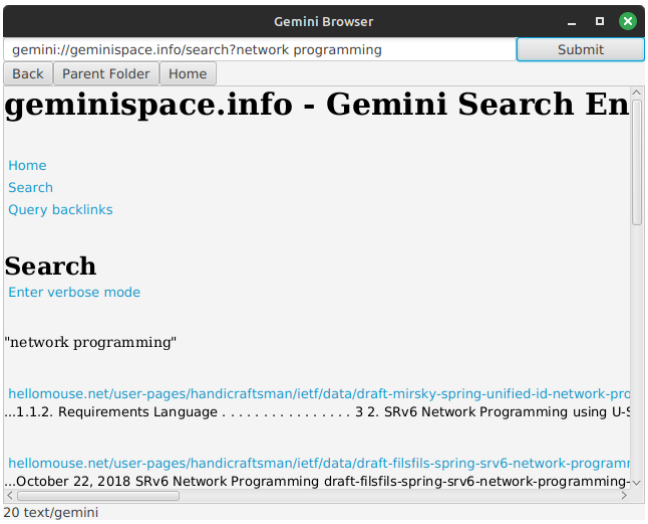

Figure 5: Result of search query from Figure 4