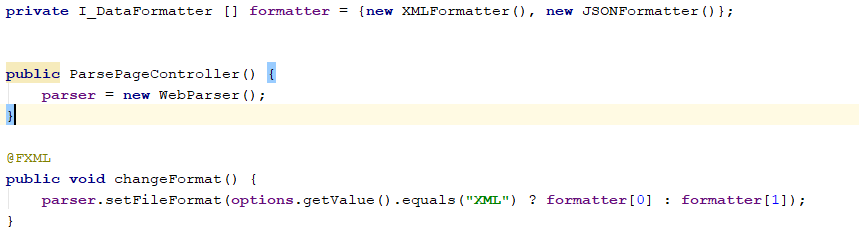
**Scenario**

The project is an automation testing framework used for testing front end web applications. It makes use of 3rd party APIs such as JSoup for scanning and extracting web elements from websites that can be stored in XML or JSON format which serves like a database. This gives the user the ability to construct tests from these collections of elements or edit previous tests. When running an automated test, elements are loaded from a file, read sequentially, and sent to the Selenium API that navigates a website from the input given which simulates a user and logs information of the validity of elements.

**Strategy**

Our use of the strategy pattern was used in our project to pick different file formats when saving web elements, with either the choice of XML or JSON. Our implementation was done using a I\_DataFormatter interface for XMLFormatter and JSONFormatter.

In the ParsePageController it holds a reference to an array of concrete types with the Interface and the WebParser.



They can be swapped in the WebParser that holds an interface I\_DataFormatter that done by changing in the view ParePageController by calling changeFormat which changes the concrete type using setFileFormat in WebParser which has a different convertFile implementation.

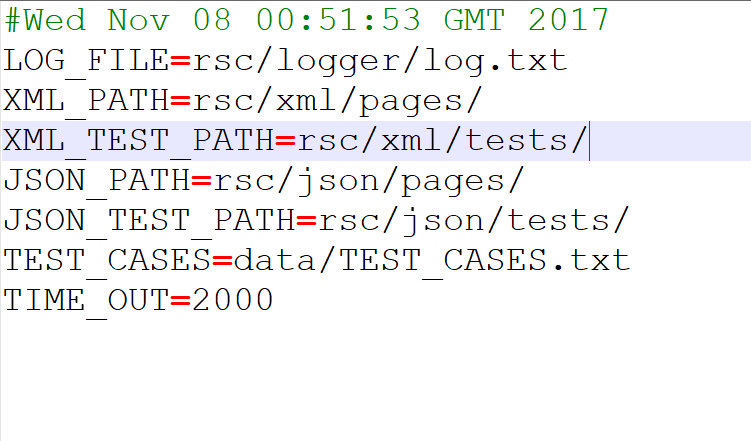


**Selenium**

A web automation tool with a rich API that is used to automate and control web browsers. It has many different bindings in other programming languages such as Java, C#, Perl, Python, Ruby, and PHP. Selenium works by mimicking a user that navigates a webpage by select different element types based on given attribute. It can simulate button clicks, enter input, and select dropdowns and much more. This is a very popular tool used for web crawling and widely used in the industry for testing web applications. To use selenium, it needs to make use of a web driver. That’s is passed to it, there are different types of drivers for different web browsers but the most supported is the geckodriver that used by FireFox which is an executable that can be used from any programming language.

**Properties**

The properties file main purpose was used to read and write to different file paths read in from dataConfig.properties which gives the program more flexibility rather than hardcoding file paths and as well changing speed of selenium navigation time between elements. This is much better for configuration management and avoids recompilation of the whole system which saves time and money.



This is done using key value pairs and changing the value will not affect the system code.

Above example shows the file paths that can be changed without

**Evaluate and critique**

Parsing a Web Page and writing out to different file formats was important. This has been achieved using the strategy design pattern which gives the uses the ability to add in more file output without modifying the code extensively that supports extensibility.

Creating test cases, this supports lots of flexibility by using the Memento pattern where a simple undo can revert changes to a previous state give the user a much better user experience and reusability where can load other test cases.

Running tests and mentioned about under the command design pattern already. This is very easy to extend and add extensibility with little or no modification. Where uses can add in new element types and create a handler for that type of element and perform different commands. This also makes use of the QueryBuilder which is efficient as it performs commands in advance before they run.

Logging messages use the interreceptor and can easily add new interceptors without much change to the code and extensibility if new interceptors needed to be added. This is important for readability for the log files