Noah Perz

“Simple Tooling” Application Design

Tools used

This program uses both Selenium WebDriver and JavaMail API. Selenium is used to gather data from the site, and JavaMail is used to send emails notifying the user of a change once the program deems it necessary.

SimpleTooling Class

- A WebDriver object linked to the site containing the desired data is created
- The WebDriver is used to store the SVG chart that contains the desired data within a WebElement object
- An email “Session” object that uses the preset sender email is prepared
- The program asks the user to enter the recipient's email address (the address that will be receiving the updates)
- A DataManager object is created using the previously discussed WebDriver, SVG WebElement, Session, and user-provided email as arguments
- The program then asks the DataManager object to run and check for updates constantly

DataManager Class

- The constructor initializes 4 instance variables using the arguments discussed above
- The “runner” method calls on the “getRecentData” method every 10 seconds and closes the program if it detects an error thrown when the browser is closed
- The “getRecentData” method goes through a series of steps:

    1. The approximate width of the SVG that actually contains data, the X position that includes the first data point (at 00:00 EST), and the distance between data points are calculated
    2. The time of the most recent data point is found through parsing the time displayed on the SVG chart
    3. The number of 5-minute intervals that have passed since 00:00 is calculated and is used to figure out approximately how far along the X axis (that represents time) the most recent data point is found. This is done by using the ratio of elapsed intervals to total intervals alongside the previously calculated size.

a. -The program forces the *estimated* position of the most recent data point to lie slightly to the left of the *actual* position of the data point. It accomplishes this by reducing the number of 5-minute intervals calculated above by 5 (meaning the position essentially backs up 25 minutes). This helps the program avoid triggering errors caused when the program fails to find an existing tooltip (ie. because the position is too far to the right where no data points exist).

4. The number of intervals is multiplied by the distance between each data point. This is then added to the X position of the data point found at 00:00.

5. The program then simulates a cursor being positioned over this X position. Incremental adjustments are made to this position until the resulting tooltip contains the data from the most recent time interval.

6. The “getDataFromString” method is called on whatever text is found within the tooltip, and stores a string containing the most recent load within the “data” instance variable.

7. The “valueManager” method sends an email if the previous data is different from this newly stored data (presuming this newly stored data is not the first piece of data collected during this particular run of the program). The email contains both the updated value and time, as well as the previous value and time. Note that the website refreshes its values every 5 minutes; furthermore, note that these values typically change with each refresh (ie. every 5 minutes). While this continuously changing load data is a typical characteristic of the website, the program does not assume such continuous variation. Rather, it detects whenever the current value changes. In this way, the program will correctly monitor the case where the load might stay constant for a larger than normal time duration for unforeseen reasons.

Instructions

1. Run the program
2. Enter the email from which you would like to receive updates
3. Let the program run in the background - do not minimize or resize the opened browser
4. Close the program directly from where it is being executed

Limitations

Although this program successfully accomplishes the task at hand, there are a few areas in which it could be improved.

1. **User inputs and lack of GUI**

a. The program lacks a proper way to stop execution. As of now, it can only be directly stopped from where it has been executed. A GUI could solve this issue by providing a “close” or “stop” button.
b. The user inputs their email directly into the console. Although this works, the process could be made much more streamlined through the use of a GUI.

2. **Finding the location of the most recent data point**
    a. Currently, the process in which the rough position of the most recent data point is found never returns a precise location, and the program must check the surrounding area for the actual location. The process in which the location is calculated could be made to be more accurate.

3. **Finding the right time to search for the most recent data point**
    a. As of now, the program simply locates the most recent data point every 10 seconds. Although this process does a sufficient job of determining when the most recent data has changed in a timely fashion, it lacks efficiency and introduces slight delays that arise when new data is sent to the website during this 10-second cooldown period.

4. **Detecting changes in the browser**
    a. The program lacks a proper way of preventing errors or issues that may result from the user resizing, minimizing, or closing the browser. I am confident that through the implementation of try-catch statements I could successfully manage these issues, but this would require additional testing. Note that I have made previous attempts at dealing with browser changes during the development of this program, but my attempts were interfering with the detection of other errors. As such, I abandoned these fixes in the interest of time.