1st: Script Description: OpeningBrowser Method

Introduction

The `OpeningBrowser` method is a Java function designed to initialize and configure a web browser for automated web testing. It uses Selenium WebDriver, a popular tool for web automation. The method sets up different browser drivers based on the input parameter, maximizes the browser window, sets implicit wait times, deletes cookies, and navigates to a specified URL.

Features and Functionality

1. **Browser Selection**:
   - The method takes a `String` parameter `Browser`, which specifies the type of browser to be opened.
   - It supports Chrome, Opera, and Firefox browsers. If an unsupported browser type is passed, it defaults to Opera.
2. **System Property Setup**:
   - For each supported browser, the corresponding WebDriver executable path is set using `System.setProperty`.
   - The paths to the WebDriver executables are hardcoded.
3. **Driver Initialization**:
   - Based on the browser type, the appropriate WebDriver instance (`ChromeDriver`, `OperaDriver`, `FirefoxDriver`) is initialized.
4. **Implicit Wait**:
   - The method sets an implicit wait time of 10 seconds using `driver.manage().timeouts().implicitlyWait`.
5. **Window Management**:
   - The browser window is maximized for better visibility and testing conditions using `driver.manage().window().maximize`.
6. **Cookie Management**:
   - All cookies are deleted to ensure a clean state for testing using `driver.manage().deleteAllCookies`.
7. **Navigation**:
   - The browser navigates to the specified URL ("https://the-internet.herokuapp.com/") using `driver.navigate().to`.

Detailed Breakdown

- **Chrome Browser Setup**:

```
java
Copy code
if (Browser.equalsIgnoreCase("Chrome")) {
    System.setProperty("webdriver.chrome.driver",
"C:\\chromedriver_win32\\chromedriver1.exe");
    driver = new ChromeDriver();
}
```

  This block checks if the input string matches "Chrome" (case-insensitive). If it does, it sets up the Chrome WebDriver and initializes the `ChromeDriver`.

- **Opera Browser Setup**:

```
java
Copy code
else if (Browser.equalsIgnoreCase("Opera")) {
    System.setProperty("webdriver.opera.driver",
"C:\\operadriver_win32\\operadriver.exe");
    driver = new OperaDriver();
}
```

  This block sets up and initializes the Opera WebDriver if the input matches "Opera".

- **Firefox Browser Setup**:

```
java
Copy code
else if (Browser.equalsIgnoreCase("Firfox")) {
    System.setProperty("webdriver.gecko.driver", "C:\\geckodriver-
v0.32.2-win32\\geckodriver.exe");
    driver = new FirefoxDriver();
}
```

  This block contains a typo in the browser name ("Firfox" instead of "Firefox"). It sets up and initializes the Firefox WebDriver for the misspelled string.

- **Default and Additional Opera Setup**:

```
java
Copy code
else if(Browser.contains("per")) {
    System.setProperty("webdriver.opera.driver",
"C:\\operadriver_win32\\operadriver.exe");
    driver = new OperaDriver();
} else {
    System.setProperty("webdriver.opera.driver",
"C:\\operadriver_win32\\operadriver.exe");
    driver = new OperaDriver();
```

```
    System.out.print("I am Here");
}
```

These blocks handle any case where the input string contains "per" or does not match any of the specified browsers. They default to setting up and initializing the Opera WebDriver, with the last block also printing a message.

- **Common Configuration**:

```
java
Copy code
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
driver.manage().window().maximize();
driver.manage().deleteAllCookies();
driver.navigate().to("https://the-internet.herokuapp.com/");
```

After initializing the WebDriver, the method sets a 10-second implicit wait, maximizes the browser window, deletes all cookies, and navigates to the specified URL.

Conclusion

The `OpeningBrowser` method is a versatile function for initializing different web browsers for automated testing using Selenium WebDriver. It ensures that the browser is in a clean state and configured properly before starting any test by setting up implicit waits, maximizing the window, and deleting cookies. The method is designed to handle different browsers but contains a few areas that could be improved, such as correcting the typo for the Firefox browser and making the WebDriver paths more flexible.

2nd: Script Description: UploadFile Method

Introduction

The `UploadFile` method is a Java function designed to automate the process of uploading a file using a web browser, leveraging the Selenium WebDriver library. The method navigates to a file upload page, selects a file from the local filesystem, uploads it, and verifies whether the upload was successful.

Features and Functionality

1. **Navigation**:
   - The method navigates to a specific part of a webpage by finding and clicking a link.
2. **File Selection**:
   - The method locates the file input element and provides the path of the file to be uploaded.
3. **File Upload**:
   - The method triggers the file upload process by clicking the submit button.
4. **Verification**:
   - The method checks if the file upload was successful by inspecting the page for an element that indicates the file name.
5. **Thread Sleep**:
   - The method uses `Thread.sleep` to pause execution at various points, allowing time for page elements to load and actions to complete.

Detailed Breakdown

- **Navigation to Upload Page**:

```
java
Copy code
driver.findElement(By.xpath("//*[@id=\"content\"]/ul/li[18]/a")).click(
);
Thread.sleep(5000);
```

  This block finds the link to the file upload page using an XPath locator and clicks it. It then pauses for 5 seconds to allow the page to load.

- **File Input Element Interaction**:

```
java
Copy code
WebElement element = driver.findElement(By.name("file"));
Thread.sleep(5000);
element.sendKeys("C:\\Users\\USER\\Desktop\\testimg.png");
Thread.sleep(5000);
```

This block locates the file input element by its name attribute and pauses for 5 seconds. It then sends the file path of the file to be uploaded and pauses again.

- **File Upload Submission**:

```
java
Copy code
driver.findElement(By.id("file-submit")).click();
Thread.sleep(5000);
```

This block finds the submit button by its ID and clicks it to start the file upload. It then pauses for 5 seconds to allow the upload to complete.

- **Upload Verification**:

```
java
Copy code
String testupload = driver.findElement(By.cssSelector("#uploaded-
files")).getText();
System.out.print(testupload + "\n");
if (testupload.isEmpty()) {
    System.out.print("File Uploaded Fail" + "\n");
    Thread.sleep(5000);
} else {
    System.out.print("File Uploaded Successfully" + "\n");
    Thread.sleep(5000);
}
```

This block retrieves the text of the element that displays the uploaded file name using a CSS selector. It prints the file name or an error message if the upload failed, followed by a 5-second pause.

Conclusion

The `UploadFile` method automates the process of uploading a file through a web interface. It carefully navigates to the upload page, interacts with the file input element, submits the file, and verifies the success of the upload. The use of `Thread.sleep` ensures that the method waits for necessary page elements to load and actions to complete, but it can be replaced with more robust waiting mechanisms like WebDriverWait for better reliability.

3rd: Script Description: Verifytext and CloseBrowser Methods

Introduction

The `Verifytext` and `CloseBrowser` methods are designed to interact with a web page using Selenium WebDriver. The `Verifytext` method navigates through a series of interactions to verify if specific text appears on the page, while the `CloseBrowser` method ensures the browser is properly closed after the operations.

Features and Functionality

1. **Navigation and Interaction**:
   - The `Verifytext` method navigates through multiple elements on a web page and clicks buttons to trigger actions.
2. **Text Verification**:
   - The method checks for the presence of specific text on the web page and verifies its appearance.
3. **Browser Management**:
   - The `CloseBrowser` method handles the closure of the browser session.
4. **Thread Sleep**:
   - Both methods use `Thread.sleep` to pause execution at various points, allowing time for page elements to load and actions to complete.

Detailed Breakdown

Verifytext Method

- **Navigation and Initial Clicks**:

```
java
Copy code
driver.findElement(By.cssSelector("#content > ul:nth-child(4) > li:nth-
child(14) > a:nth-child(1)")).click();
Thread.sleep(5000);
driver.findElement(By.xpath("/html/body/div[2]/div/div/a[2]")).click();
Thread.sleep(5000);
driver.findElement(By.cssSelector("#start > button:nth-
child(1)")).click();
Thread.sleep(5000);
```

This block navigates through a series of page elements using CSS selectors and XPath to click specific links and buttons, with pauses in between to allow the page to load.

- **Text Appearance Verification**:

```
java
Copy code
for (int counter = 0; counter < 100; counter--) {
    if (driver.findElement(By.cssSelector("#finish")).isDisplayed()) {
        counter = 1000;
    }
}
```

This loop continually checks if the element with the CSS selector `#finish` is displayed. Once it is, the loop is exited by setting the counter to a high value.

- **Text Extraction and Validation**:

```
java
Copy code
String testupload2 =
driver.findElement(By.cssSelector("#finish")).getText();
System.out.print(testupload2 + "\n");
if (testupload2.isEmpty()) {
    System.out.print("Expected Text Not Appear" + "\n");
    Thread.sleep(5000);
} else {
    System.out.print("Expected Text Appear Successfully" + "\n");
    Thread.sleep(5000);
}
```

This block extracts the text from the element with the CSS selector `#finish`, prints it, and checks if it is empty. It prints a corresponding message based on whether the text was found or not, followed by a pause.

CloseBrowser Method

- **Browser Closure**:

```
java
Copy code
public void CloseBrowser() throws InterruptedException {
    driver.close();
    driver.quit();
}
```

This method closes the current browser window using `driver.close()` and then quits the WebDriver session using `driver.quit()`. This ensures that all browser windows are closed and the WebDriver instance is properly terminated.

Conclusion

The `Verifytext` method automates a series of interactions on a web page to verify the presence of specific text, ensuring that the required elements have loaded and displayed correctly. The `CloseBrowser` method ensures that the browser is properly closed after the interactions. The use of `Thread.sleep` provides necessary pauses, but can be improved with more robust waiting mechanisms like WebDriverWait for better reliability. These methods together enable automated testing of web page functionality and cleanup.