

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with various Java files and libraries.
- BaseWebUtil.class:** The current file being edited, containing Java code for a `BrowserType` enum.
- BrowserType.class:** A secondary file shown in the editor.
- Java EE Debug:** A floating window showing a dropdown menu for browser selection, listing options like Firefox, Chrome, Internet Explorer, and others.

```
package com.ebay.maul.driver.web;
public enum BrowserType {
    Firefox("firefox"),
    Chrome("chrome"),
    InternetExplore("ieexplorer"),
    Safari("safari"),
    Opera("opera"),
    Android("android"),
    iPhone("iphone"),
    iPad("ipad"),
    PhantomJS("phantomjs"),
    HTMLUnit("htmlunit");
}

public static BrowserType getBrowserType(String type) {
    if (type == null) {
        return BrowserType.FireFox;
    }
    for (BrowserType browserType : BrowserType.values()) {
        if (browserType.getType().contains(type.toLowerCase())) {
            return browserType;
        }
    }
    return BrowserType.FireFox;
}

private String type;
BrowserType(String type) {
    this.type = type;
}
public String getType(){
    return this.type;
}
}
```

No operations to display at this time.

Project Explorer   Navigator   Type Hierarchy

BaseWebUtil.class   BrowserType.class   CustomEventFiringWebDriver.class

```

1 package com.ebay.mauli.driver.web;
2
3 /**
4  * This file is used to support upload file in remote webdriver
5  * Author Junjishi
6  */
7
8 public class CustomEventFiringWebDriver extends EventFiringWebDriver {
9     ...
10    private FileDetector fileDetector = new UselessFileDetector();
11    private WebDriver driver = null;
12
13    public CustomEventFiringWebDriver(WebDriver driver) {
14        super(driver);
15        this.driver = driver;
16    }
17
18    public void setFileDetector(FileDetector detector) {
19        if (detector == null)
20            throw new WebDriverException(
21                "You may not set a file detector that is null");
22        fileDetector = detector;
23    }
24
25    public FileDetector getFileDetector() {
26        return fileDetector;
27    }
28
29    public WebDriver getWebDriver() {
30        return driver;
31    }
32
33    ...
34
35    ...
36
37    ...
38}
39

```

Markers   Properties   Servers   Data Source Explorer   Snippets   Console   Progress   Search   Results of running class RaiseClaimTest   Package Explorer

No operations to display at this time.

Read-Only   Smart Insert   1:1

Project Explorer   Navigator   Type Hierarchy

BaseWebUtil.class   BrowserType.class   CustomEventFiringWebDriver.class   IScreenshotListener.class

```

1 package com.ebay.mauli.driver.web;
2
3 /**
4  * Screenshot Listener to be implemented to tap screenshots and HTMLs captured in MAUI.
5  * Author Knagnolla
6  */
7
8 public interface IScreenshotListener {
9
10    //below method was commented and instead exposed onScreenshotCaptured method.
11    /*public void doScreenshotCapture(String pageNo, String rlogId, String pageTitle,
12     * String imgType, String imgPath);*/
13
14    public void onScreenshotCaptured(ScreenShot screenshot);
15
16}
17

```

Markers   Properties   Servers   Data Source Explorer   Snippets   Console   Progress   Search   Results of running class RaiseClaimTest   Package Explorer

No operations to display at this time.

IScreenshotListener.onScreenshotCaptured(S

Read-Only   Smart Insert   1:1

Java EE Debug

ScreenShot

```

7 import com.ebay.maui.controller.ContextManager;
8
9 public class ScreenShot {
10
11     private String location;
12     private String htmlSourcePath;
13     private String callLog;
14     private String pageId;
15     private String rlogId;
16     private String title;
17     private String suiteName;
18     private boolean isException;
19     private String outputDirectory;
20
21     public ScreenShot() {
22         if (ContextManager.getGlobalContext().getTestNGContext() != null) {
23             suiteName = ContextManager.getGlobalContext().getTestNGContext()
24                 .getSuite().getName();
25             outputDirectory = ContextManager.getGlobalContext()
26                 .getTestNGContext().getOutputDirectory();
27         }
28     }
29
30     public boolean isException() {
31         return isException;
32     }
33
34     public void setException(boolean isException) {
35         this.isException = isException;
36     }
37
38     public String getSuiteName() {
39         return suiteName;
40     }
41
42     public String getOutputDirectory() {
43         return outputDirectory;
44     }
45
46     public void setOutputDirectory(String outputDirectory) {
47         this.outputDirectory = outputDirectory;
48     }
49
50     public void setSuiteName(String suiteName) {
51         this.suiteName = suiteName;
52     }
53
54     public String getLocation() {
55         return location;
56     }
57
58     public void setLocation(String location) {
59         this.location = location;
60     }
61
62     public void setHtmlSourcePath(String htmlSourcePath) {
63         this.htmlSourcePath = htmlSourcePath;
64     }
65
66     public void setImagePath(String imagePath) {
67         this.imagePath = imagePath;
68     }
69
70     public String getPageId() {
71         return pageId;
72     }
73
74     public void setPageId(String pageId) {
75         this.pageId = pageId;
76     }
77
78     public String getCallLog() {
79         return callLog;
80     }
81
82     public void setCallLog(String callLog) {
83         this.callLog = callLog;
84     }
85
86     public String getRlogId() {
87         return rlogId;
88     }
89
90     public void setRlogId(String rlogId) {
91         this.rlogId = rlogId;
92     }
93
94     public String getHtmlSourcePath() {
95         return htmlSourcePath;
96     }
97
98     public String getImagePath() {
99         return imagePath;
100    }
101
102    public String getTitle() {
103        return title;
104    }
105
106    public void setTitle(String title) {
107        this.title = title;
108    }
109
110    public String getFullImagePath() {
111        if (this.imagePath != null) {
112            return this.imagePath.replace(suiteName, outputDirectory);
113        } else {
114            return null;
115        }
116    }
117
118    public String getFullHtmlPath() {
119        if (this.htmlSourcePath != null) {
120            return this.htmlSourcePath.replace(suiteName, outputDirectory);
121        } else {
122            return null;
123        }
124    }
125    @Override
126    public String toString() {
127        return "exception:" + this.isException + "location:" + this.location
128            + "title:" + this.title + "htmlSource:"
129            + this.getFullHtmlPath() + "image:" + this.getFullImagePath();
130    }
131}
132

```

Read-Only Smart Insert 2 : 1

Java EE Debug

ScreenShot

```

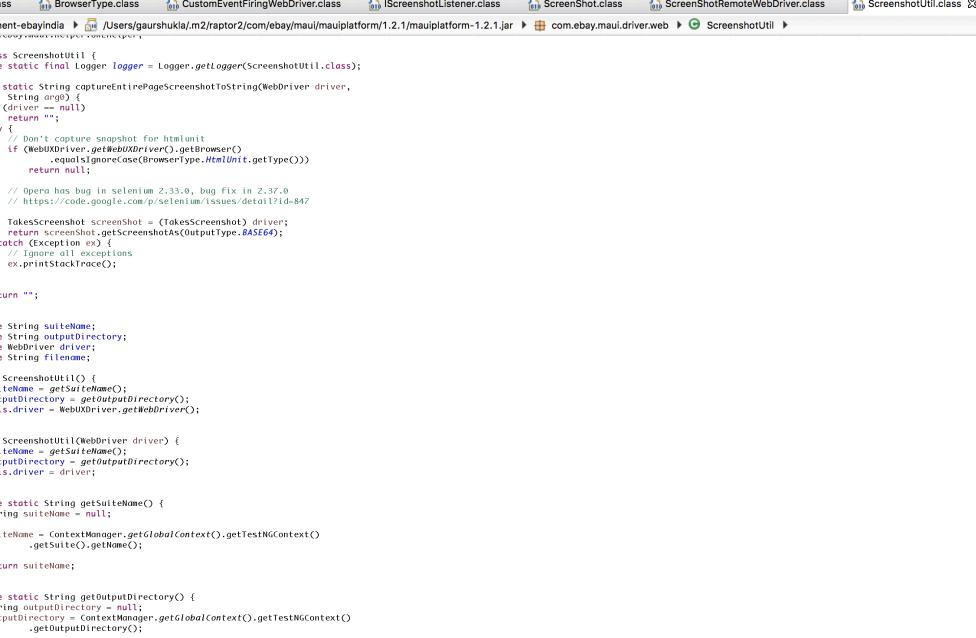
60     this.imagePath = imagePath;
61
62     public String getPageId() {
63         return pageId;
64     }
65
66     public void setPageId(String pageId) {
67         this.pageId = pageId;
68     }
69
70     public String getCallLog() {
71         return callLog;
72     }
73
74     public void setCallLog(String callLog) {
75         this.callLog = callLog;
76     }
77
78     public String getRlogId() {
79         return rlogId;
80     }
81
82     public void setRlogId(String rlogId) {
83         this.rlogId = rlogId;
84     }
85
86     public String getHtmlSourcePath() {
87         return htmlSourcePath;
88     }
89
90     public String getImagePath() {
91         return imagePath;
92     }
93
94     public String getTitle() {
95         return title;
96     }
97
98     public void setTitle(String title) {
99         this.title = title;
100    }
101
102    public String getFullImagePath() {
103        if (this.imagePath != null) {
104            return this.imagePath.replace(suiteName, outputDirectory);
105        } else {
106            return null;
107        }
108    }
109
110    public String getFullHtmlPath() {
111        if (this.htmlSourcePath != null) {
112            return this.htmlSourcePath.replace(suiteName, outputDirectory);
113        } else {
114            return null;
115        }
116    }
117
118    public String getFullImagePath() {
119        if (this.htmlSourcePath != null) {
120            return this.htmlSourcePath.replace(suiteName, outputDirectory);
121        } else {
122            return null;
123        }
124    }
125    @Override
126    public String toString() {
127        return "exception:" + this.isException + "location:" + this.location
128            + "title:" + this.title + "htmlSource:"
129            + this.getFullHtmlPath() + "image:" + this.getFullImagePath();
130    }
131}
132

```

Read-Only Smart Insert 2 : 1

The screenshot shows the Eclipse IDE interface with the Java EE perspective selected. The top menu bar includes 'File', 'Edit', 'Select', 'Run', 'Debug', 'View', 'Project Explorer', 'Navigator', 'Type Hierarchy', 'Outline', 'Properties', 'Servers', 'Data Source Explorer', 'Snippets', 'Console', 'Progress', 'Search', 'Results of running class RaiseClaimTest', and 'Package Explorer'. The left sidebar displays the 'Project Explorer' with various Java packages and files, including 'maulplatform-1.2.1.jar', 'com.ebay.maul.controller', 'com.ebay.maul.driver.api', 'com.ebay.maul.driver.db', 'com.ebay.maul.driver.email', 'com.ebay.maul.driver.ssh', 'com.ebay.maul.driver.web', 'BaseWebUtil.class', 'BaseWebUtil.java', 'BrowserType.class', 'CustomEventFiringWebDriver.class', 'IScreenshotListener.class', 'ScreenshotShot.class', 'ScreenShotRemoteWebDriver.class', 'ScreenshotUtil.class', 'WebDriverConfig.class', 'WebDriverExceptionListener.class', 'WebDriverMode.class', 'WebDriverDriver.class', 'com.ebay.maul.driver.web.element', 'com.ebay.maul.driver.web.factory', 'com.ebay.maul.exception', 'com.ebay.maul.helper', 'com.ebay.maul.reporter', 'com.ebay.maul.reporter.pluginmodel', 'reporter.css', 'reporter.images.lightbox', 'reporter.images.mktree', 'reporter.images.yukontoolbox', 'reporter.js', 'templates', 'META-INF', and 'tmp'. Below the project tree, a list of external jars is shown: 'guice-3.0.jar', 'javassist-1.jar', 'javax.inject-1.jar', 'aopalliance-1.0.jar', 'commons-HttpClient-3.1.jar', 'velocity-1.7.jar', 'commons-collections-3.2.1.jar', 'commons-lang-2.4.jar', 'mysql-connector-java-5.1.18.jar', 'ojdbc15-11.1.0.6.jar', 'sequelize-1.0.6.jar', 'mail-1.4.5.jar', and 'activation-1.1.jar'. The central workspace shows the 'BaseWebUtil.java' file open in the editor. The code implements the `ScreenShotRemoteWebDriver` interface, which extends `RemoteWebDriver`. It includes methods for taking screenshots and executing commands. The code uses annotations like `@Override`, `@Parameters`, and `@Test`. The right side of the interface shows the 'Outline' view with nodes like 'ScreenShotRemoteWebDriv', 'ScreenShotRemoteWebDriv', 'ScreenShotRemoteWebDriv', 'ScreenShotRemoteWebDriv', and 'getScreenshotAs(OutputType)'.

```
package com.ebay.maul.driver.web;
import java.net.URL;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriverException;
import org.openqa.selenium.remote.CapabilityType;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.RemoteWebElement;
public class ScreenShotRemoteWebDriver extends RemoteWebDriver implements TakesScreenshot {
    public ScreenShotRemoteWebDriver(DesiredCapabilities capabilities) {
        super(capabilities);
    }
    public ScreenShotRemoteWebDriver(URL url, DesiredCapabilities capabilities) {
        super(url, capabilities);
    }
    public ScreenShotRemoteWebDriver() {
    }
    public <X> X getScreenshotAs(OutputType<X> target)
            throws WebDriverException {
        if ((Boolean) getCapabilities().getCapability(
                CapabilityType.TAKES_SCREENSHOT)) {
            String output = executeDriverCommand(SCREENSHOT).getValue();
            return target.convertFromBase64Png(output);
        }
        return null;
    }
}
```



The screenshot shows an IDE interface with a Java code editor. The code is for a `ScreenShotUtil` class, which handles capturing screenshots from various web drivers. The code includes logic for different browser types like Selenium, HtmlUnit, and WebKit, and handles exceptions and file output paths. The IDE's toolbar and project navigation bar are visible at the top.

```
1 package com.automation.core.util;
2
3 import org.openqa.selenium.OutputType;
4 import org.openqa.selenium.TakesScreenshot;
5 import org.openqa.selenium.WebDriver;
6 import org.openqa.selenium.htmlunit.HtmlUnitDriver;
7 import org.openqa.selenium.webkit.WebKitDriver;
8 import org.openqa.selenium.firefox.FirefoxDriver;
9 import org.openqa.selenium.ie.IEDriver;
10 import org.openqa.selenium.chrome.ChromeDriver;
11 import org.openqa.selenium.safari.SafariDriver;
12 import org.openqa.selenium.phantomjs.PhantomJSDriver;
13 import org.openqa.selenium.edge.EdgeDriver;
14 import org.openqa.selenium.firefox.firebug.FirebugDriver;
15 import org.openqa.selenium.firefox.firefoxProfile.FirefoxProfileDriver;
16 import org.openqa.selenium.firefox.firefoxProfileManager.FirefoxProfileManagerDriver;
17 import org.openqa.selenium.firefox.firefoxSession.FirefoxSessionDriver;
18 import org.openqa.selenium.firefox.firefoxSessionManager.FirefoxSessionManagerDriver;
19
20 public class ScreenShotUtil {
21     private static final Logger logger = Logger.getLogger(ScreenShotUtil.class);
22
23     public static String captureEntirePageScreenshotToString(WebDriver driver,
24         String arg0) {
25         if (driver == null)
26             return "";
27         try {
28             // Don't capture snapshot for htmlunit
29             if (WebkitDriver.get#WebkitDriver().getBrowser()
30                 .equalsIgnoreCase(BrowserType.HTMLUNIT.getType()))
31                 return null;
32
33             // Opera has bug in selenium 2.33.0, bug fix in 2.37.0
34             // https://code.google.com/p/selenium/issues/detail?id=847
35
36             TakesScreenshot screenshot = (TakesScreenshot) driver;
37             return screenshot.getScreenshotAs(OutputType.BASE64);
38         } catch (Exception ex) {
39             // Ignore all exceptions
40             ex.printStackTrace();
41         }
42         return "";
43     }
44
45     private String suiteName;
46     private String outputDirectory;
47     private WebDriver driver;
48     private String filename;
49
50     public ScreenShotUtil() {
51         suiteName = getSuiteName();
52         outputDirectory = getOutputDirectory();
53         this.driver = WebkitDriver.get#WebkitDriver();
54     }
55
56     public ScreenShotUtil(WebDriver driver) {
57         suiteName = getSuiteName();
58         outputDirectory = getOutputDirectory();
59         this.driver = driver;
60     }
61
62     private static String getSuiteName() {
63         String suiteName = null;
64
65         suiteName = ContextManager.getGlobal(context).getTestNGContext()
66             .getSuite().getName();
67
68         return suiteName;
69     }
70
71     private static String getOutputDirectory() {
72         String outputDirectory = null;
73         outputDirectory = ContextManager.getGlobal(context).getTestNGContext()
74             .getOutputDirectory();
75
76         return outputDirectory;
77     }
78
79     private void handleSource(String htmlSource, ScreenShot screenShot) {
80         if (htmlSource == null) {
81             // driver.switchTo().defaultContent();
82             htmlSource = driver.getPageSource();
83         }
84     }
85
86     private void handleImage(String htmlSource, ScreenShot screenShot) {
87         if (htmlSource == null) {
88             // driver.switchTo().activeElement();
89             htmlSource = driver.getPageSource();
90         }
91     }
92
93 }
```



The screenshot shows the Eclipse IDE interface with the Java EE perspective selected. The top menu bar includes 'File', 'Edit', 'Search', 'Run', 'View', 'Project', 'Server', 'Help', and 'Quick Access'. The left sidebar contains 'File Explorer', 'Properties', 'Tasks', 'Problems', 'Console', 'Output', 'Javadoc', 'JavaDoc', 'Help', and 'About'. The central workspace displays the Java code for 'BaseWebUtil.class'.

```
1 package com.ebay.mau.driver.web;
2
3 import java.io.File;
4 import java.util.List;
5
6 import org.openqa.selenium.WebDriver;
7 import org.openqa.selenium.WebElement;
8
9 import com.ebay.mau.component.ebaydriver.IScreenshotListener;
10 import com.ebay.mau.component.ebaydriver.ScreenShot.class;
11 import com.ebay.mau.component.ebaydriver.ScreenShotRemoteWebDriver.class;
12 import com.ebay.mau.component.ebaydriver.ScreenshotUtil.class;
13
14 public class BaseWebUtil {
15
16     /**
17      * @param driver
18      * @param suiteHome
19      * @param filename
20      */
21     public void savePageSource(WebDriver driver, String suiteHome, String filename) {
22         String pageId = null;
23         String logId = null;
24
25         try {
26             FileHelper.writeToFile(suiteHome + "htmls/" + filename,
27                     suiteHome + "htmls/" + filename);
28             screenShot.setHtmlSourcePath(suiteHome + "htmls/" + filename
29                     + ".html");
30         } catch (IOException e) {
31             logger.warn("EX", e);
32         }
33
34         String htmlSource = null;
35         String pageId = null;
36         try {
37             int startIndex = htmlSource.indexOf("RlogId ");
38             if (startIndex != -1) {
39                 int endIndex = htmlSource.substring(startIndex).indexOf(">");
```

```
148     private void handlePageId(ScreenShot screenShot) {
149         String pageId = screenShot.getPageId();
150         if (pageId == null) {
151             URL url;
152             try {
153                 url = new URL(screenShot.getLocation());
154                 if (url.getQuery() != null) {
155                     pageId = url.getQuery();
156                 } else if (url.getIdentification() != null) {
157                     pageId = url.getIdentification();
158                 }
159                 pageId = pageId.substring(0, pageId.indexOf("&"));
160                 screenShot.setPageId(pageId);
161             }
162         } catch (MalformedURLException e) {
163         }
164     }
165 }
166 }
167
168 private void handleImage(ScreenShot screenShot) {
169     try {
170         String screenshotString = captureEntirePageScreenshotToString(
171             WebDriverFactory.getWebDriver(), suiteName);
172
173         if (screenshotString != null) {
174             byte[] screenshotStringBytes = screenshotString.getBytes();
175             File file = writeImageToOutputDirectory(suiteName + "/screenshots/" +
176                 filename + ".png", byteArray);
177             screenShot.setImagePath(suiteName + "/screenshots/" + filename +
178                 ".png");
179         }
180     } catch (IOException e) {
181         logger.warn("Exception: " + e);
182         e.printStackTrace();
183     }
184 }
185 }
186
187 private void handleTitle(String title, ScreenShot screenShot) {
188     if (title == null) {
189         // driver.switchTo().defaultContent();
190         title = driver.getTitle();
191     }
192     if (title == null) {
193         title = "";
194     }
195     screenShot.setTitle(title);
196 }
197
198 public ScreenShot captureWebPageSnapshot() {
199     ScreenShot screenShot = new ScreenShot();
200
201     if (ContextManager.getThreadContext() == null
202         || outputdirectory == null)
203         return screenShot;
204     screenShot.setSuiteName(this.suiteName);
205
206     try {
207         String url = null;
208         try {
209             url = driver.getCurrentUrl();
210         } catch (org.openqa.selenium.UnhandledAlertException ex) {
211             // ignore alert exception
212             ex.printStackTrace();
213         }
214     }
215 }
```

```

  1  // ignore alert exception
  2  ex.printStackTrace();
  3  url = driver.getCurrentUrl();
  4 
  5  String title = driver.getTitle();
  6  String pageSource = driver.getPageSource();
  7 
  8  String filename = URLHelper.getRandomHashCode("web");
  9  this.filename = filename;
 10  screenShot.setlocation(url);
 11 
 12  handleTitle(title, screenShot);
 13  handleSource(pageSource, screenShot);
 14  handleId(screenShot);
 15  if (ContextManager.getThreadContext().isCaptureSnapshot()) {
 16      handleImage(screenShot);
 17      Logging.logScreenshot(screenShot);
 18  }
 19  } catch (WebSessionTerminatedException e) {
 20      throw e;
 21  } catch (Exception ex) {
 22      ex.printStackTrace();
 23  }
 24  if (ContextManager.getThreadContext().isCaptureSnapshot())
 25      ContextManager.getThreadContext().addScreenShot(screenShot);
 26  return screenShot;
 27 }
 28 /**
 29 * Used by WebDriverExceptionListener, don't log the exception but put it
 30 * into context
 31 */
 32 @Override
 33 protected void driverError(WebDriverException e) {
 34     //parse driver
 35     //parse location
 36     //parse htmlSource
 37     //parse title
 38 
 39     public void capturePageSnapshotOnException() {
 40         Boolean capture = ContextManager.getThreadContext()
 41             .isCaptureSnapshot();
 42         ContextManager.getThreadContext().setAttribute(
 43             Context.CAPTURE_SNAPSHOT, "true");
 44         captureWebPageSnapshot();
 45         ContextManager.getThreadContext().setAttribute(
 46             Context.CAPTURE_SNAPSHOT, Boolean.toString(capture));
 47         // ContextManager.getThreadContext().setScreenshotName(filename);
 48         // ContextManager.getThreadContext().setWebExceptionURL(location);
 49         // ContextManager.getThreadContext().setWebExceptionMessage(title + " - "
 50         // screenshot);
 51         // screenshot.setException(true);
 52         if (ContextManager.getThreadContext().getScreenshots().size() > 0)
 53             ContextManager.getThreadContext().getScreenshots().getLast()
 54                 .setException(true);
 55     }
 56     public static void captureScreenshot(String messagePrefix) {
 57         if (ContextManager.getThreadContext() != null
 58             && ContextManager.getThreadContext().isCaptureSnapshot()
 59             && getOutputDirectory() != null) {
 60             String filename = URLHelper.getRandomHashCode("HtmlElement");
 61             StringBuffer sbMessage = new StringBuffer();
 62             try {
 63                 String img = ScreenshotUtil
 64                     .captureEntirePageScreenshotToString();
 65             }
 66         }
 67     }
 68     public static void captureScreenshot(String messagePrefix) {
 69         if (ContextManager.getThreadContext() != null
 70             && ContextManager.getThreadContext().isCaptureSnapshot()
 71             && getOutputDirectory() != null) {
 72             String filename = URLHelper.getRandomHashCode("HtmlElement");
 73             StringBuffer sbMessage = new StringBuffer();
 74             try {
 75                 String img = ScreenshotUtil
 76                     .captureEntirePageScreenshotToString(
 77                         WebDDriver.getWebDDriver(), **);
 78                 if (img == null)
 79                     return;
 80                 byte[] byteArray = img.getBytes(); // KEEPME
 81                 if (byteArray != null && byteArray.length > 0) {
 82                     String imgPath = "screenshots/" + filename + ".png";
 83                     FileHelper.writeImage(getOutputDirectory() + imgPath,
 84                         byteArray);
 85                     ScreenShot screenShot = new ScreenShot();
 86                     String imgName = getScreenshotName();
 87                     screenShot.setScreenshotName(imgName);
 88                     ContextManager.getThreadContext().addScreenShot(screenShot);
 89                     sbMessage.append(messagePrefix + "#<a href=\"" + imgPath
 90                         + "\" class=\"lightbox>Screenshot</a>");
 91                     Logging.logScreenshot(null, sbMessage.toString(), false);
 92                     Logging.logScreenshot(screenShot);
 93                     sbMessage = null;
 94                 }
 95             }
 96         }
 97     }
 98     // private static String readFileAsString(File file) throws
 99     // IOException {
100     //     StringBuffer fileData = new StringBuffer(10000);
101     //     BufferedReader reader = new BufferedReader(new FileReader(file));
102     //     char[] buf = new char[1024];
103     //     int numRead;
104     //     while ((numRead = reader.read(buf)) != -1) {
105         // String readData = String.valueOf(buf, 0, numRead);
106         // fileData.append(readData);
107         // buf = new char[1024];
108     }
109     // reader.close();
110     // return fileData.toString();
111     // }
112 }
113 
```

```

12 public class WebDriverConfig {
13
14     // Handle PayPal problem. For ebay file a ticket as SASRV15354682 or
15     // contact DL-eBay-DA-NTIntegration
16     private boolean setAssumeUntrustedCertificateIssuer = true;
17     private boolean setAcceptUntrustedCertificates = true;
18     private boolean enableJavaScript = true;
19     private boolean acceptInsecureCerts = true;
20
21     private WebDriver driver;
22     private BrowserType browser = BrowserType.FireFox;
23     private WebDriverMode mode = WebDriverMode.LocallyOnRC;
24     private String hubUrl;
25     private String ffbinPath;
26     private String ffProfilePath;
27     private String ffbinPath;
28     private String fedriverPath;
29     private String chromeDriverPath;
30     private String chromeExtensionPath;
31     private String operaDriverPath;
32     private String chromeDriverPath;
33     private static final int DEFAULT_IMPLICIT_WAIT_TIMEOUT = 90 * 1000;
34     final static public int DEFAULT_EXPLICIT_WAIT_TIMEOUT = 5;
35     final static public int DEFAULT_PAGE_LOAD_TIMEOUT = 15;
36     final static public int DEFAULT_SCRIPT_TIMEOUT = 90;
37     private int implicitWaitTimeout = DEFAULT_IMPLICIT_WAIT_TIMEOUT;
38     private int explicitWaitTimeout = DEFAULT_EXPLICIT_WAIT_TIMEOUT;
39     private int pageLoadTimeout = DEFAULT_PAGE_LOAD_TIMEOUT;
40     private String outputDirectory;
41     private String userAgent;
42     private Platform platform;
43     private String userAgentOverride;
44     private String authTrustLevel;
45     private String proxyHost;
46     private boolean addIFixerCollectorExtension = false;
47     private ArrayList<WebDriverEventListener> webDriverListeners;
48     private boolean useFirefoxDefaultProfile = true;
49     private String browserName;
50     private int browserWindowHeight = -1;
51
52     private String proxyHost;
53     private String appName;
54     private String appVersion;
55     private String androidTarget;
56     private String resolution;
57
58     public ArrayList<WebDriverEventListener> getWebDriverListeners() {
59         return webDriverListeners;
60     }
61
62     public void setWebDriverListeners(
63         ArrayList<WebDriverEventListener> webDriverListeners) {
64         this.webDriverListeners = webDriverListeners;
65     }
66
67     public void setWebDriverListeners(String listeners) {
68         ArrayList<WebDriverEventListener> listenerList = new ArrayList<WebDriverEventListener>;
69         String[] list = listeners.split(",");
70         for (int i = 0; i < list.length; i++) {
71             WebDriverEventListener listener = null;
72             try {
73                 if (!list[i].equals("")) {
74                     listener = (WebDriverEventListener) Class.forName(list[i])
75                         .newInstance();
76                 }
77             } catch (InstantiationException e) {
78                 e.printStackTrace();
79             } catch (IllegalAccessException e) {
80                 e.printStackTrace();
81             } catch (ClassNotFoundException e) {
82                 e.printStackTrace();
83             } catch (ClassCastException e) {
84                 e.printStackTrace();
85             } catch (NoSuchMethodException e) {
86                 e.printStackTrace();
87             }
88         }
89         this.webDriverListeners = listenerList;
90     }
91
92     public BrowserType getBrowser() {
93         return browser;
94     }
95
96     public String getBrowserDownloadDir() {
97         return browserDownloadDir;
98     }
99
100    public String getBrowserVersion() {
101        return browserVersion;
102    }
103
104    public String getChromeInPath() {
105        return chromeInPath;
106    }
107
108    public String getChromeDriverPath() {
109        return chromeDriverPath;
110    }
111
112    public WebDriver getDriver() {
113        return driver;
114    }
115
116    public int getExplicitWaitTimeout() {
117        if (explicitWaitTimeout < getImplicitWaitTimeout()) {
118            return (int) getImplicitWaitTimeout();
119        } else {
120            return explicitWaitTimeout;
121        }
122    }
123
124    public String getFFBinPath() {
125        return ffbinPath;
126    }
127
128    public String getFFProfilePath() {
129        if (ffProfilePath == null
130            & getClass().getResources("/profiles/customProfileDirCUSTFF") != null) {
131
132            try {
133                return getClass()
134                    .getResources("/profiles/customProfileDirCUSTFF")
135                    .toURI().getPath();
136            } catch (URISyntaxException e) {
137                throw new RuntimeException(e);
138            }
139        } else
140            return ffProfilePath;
141
142    }
143
144}

```

The screenshot shows the Java code for the `WebDriverConfig` class in an IDE. The code is annotated with line numbers from 140 to 204. The class implements several methods related to WebDriver configuration, such as getting hub URL, driver path, implicit wait timeout, mode, and profile path. It also handles proxy settings and returns the proxy host.

```
140 }
141     public String getHubUrl() {
142         return huburl;
143     }
144 
145     public String getDriverPath() {
146         return iedriverpath;
147     }
148 
149     public double getImplicitWaitTimeout() {
150         return implicitwaittimeout;
151     }
152 
153     public WebDriverMode getMode() {
154         return mode;
155     }
156 
157     public String getNtAuthTrustedUrls() {
158         return ntauthtrustedurls;
159     }
160 
161     public String getOpenProfilePath() {
162         if (operaprofilepath == null) {
163             //&gt;&gt; getClass().getresource("/profiles/operaprofile") != null {
164 
165                 try {
166                     return getClass().getResource("/profiles/operaprofile").toURI()
167                         .getPath();
168                 } catch (URISyntaxException e) {
169                     throw new RuntimeException(e);
170                 }
171             }
172         }
173         return operaprofilepath;
174     }
175 
176     public String getOutputDirectory() {
177         return outputdirectory;
178     }
179 
180     public int getPageLoadTimeout() {
181         return pageloadtimeout;
182     }
183 
184     public Platform getPlatform() {
185         return platform;
186     }
187 
188     public Proxy getProxy() {
189         Proxy proxy = null;
190         if (proxy == null) {
191             proxy = new Proxy();
192             proxy.setProxyType(ProxyType.MANUAL);
193             proxy.setHttpProxy(proxyhost);
194             proxy.setHttpsProxy(proxyhost);
195             proxy.setSslProxy(proxyhost);
196         }
197         return proxy;
198     }
199 
200     public String getProxyHost() {
201         return proxyhost;
202     }
203 }
```

The screenshot shows the Java code for the `WebDriverConfig` class in an IDE. The code is annotated with line numbers from 206 to 798. The class contains various methods for setting up browser configurations, including user agent override, session timeout, error collector extension, fire fox default profile, enable JavaScript, enable cookie, accept untrusted certificates, assume untrusted certificate issuer, set browser type, set download directory, set browser version, set chrome bin path, set chrome driver path, and set driver.

```
206     public String getUserAgentOverride() {
207         return this.userAgentoverride;
208     }
209 
210     public int getWebSessionTimeout() {
211         return webSessionTimeout;
212     }
213 
214     public boolean isAddJSErrorCollectorExtension() {
215         return addJSerrorCollectorExtension;
216     }
217 
218     public boolean isUseFirefoxDefaultProfile() {
219         return this.useFirefoxDefaultProfile;
220     }
221 
222     public void setUseFirefoxDefaultProfile(boolean useFirefoxDefaultProfile) {
223         this.useFirefoxDefaultProfile = useFirefoxDefaultProfile;
224     }
225 
226     public boolean isEnableJavascript() {
227         return enablejavascript;
228     }
229 
230     public boolean isEnableCookie() {
231         return enablecookie;
232     }
233 
234     public boolean isSetAcceptUntrustedCertificates() {
235         return setAcceptUntrustedCertificates;
236     }
237 
238     public boolean isSetAssumeUntrustedCertificateIssuer() {
239         return setAssumeUntrustedCertificateIssuer;
240     }
241 
242     public void setAddJSerrorCollectorExtension(
243         boolean addJSerrorCollectorExtension) {
244         this.addJSerrorCollectorExtension = addJSerrorCollectorExtension;
245     }
246 
247     public void setBrowser(BrowserType browser) {
248         this.browser = browser;
249     }
250 
251     public void setBrowserDownloadDir(String browserDownloadDir) {
252         this.browserDownloadDir = browserDownloadDir;
253     }
254 
255     public void setBrowserVersion(String browserVersion) {
256         this.browserVersion = browserVersion;
257     }
258 
259     public void setChromeBinPath(String chromeBinPath) {
260         this.chromeBinPath = chromeBinPath;
261     }
262 
263     public void setChromeDriverPath(String chromeDriverPath) {
264         this.chromeDriverPath = chromeDriverPath;
265     }
266 
267     public void setDriver(WebDriver driver) {
268         this.driver = driver;
269     }
270 }
```

Java EE Debug

Quick Access

BaseWebUtil.class BrowserType.class CustomEventFiringWebD IScreenshotListener.class ScreenShot.class ScreenShotRemoteWebD ScreenshotUtil.class WebDriverConfig

```

276 public void setEnableJavascript(boolean enableJavascript) {
277     this.enableJavascript = enableJavascript;
278 }
279 public void setEnableCookie(boolean enableCookie) {
280     this.enableCookie = enableCookie;
281 }
282 public void setExplicitWaitTimeout(int explicitWaitTimeout) {
283     this.explicitWaitTimeout = explicitWaitTimeout;
284 }
285 public void setFFBinPath(String fFBinPath) {
286     this.fFBinPath = fFBinPath;
287 }
288 public void setFFProfilePath(String ffProfilePath) {
289     this.ffProfilePath = ffProfilePath;
290 }
291 public void setHubUrl(String hubUrl) {
292     this.hubUrl = hubUrl;
293 }
294 public void settleDriverPath(String ieDriverPath) {
295     this.ieDriverPath = ieDriverPath;
296 }
297 public void setImplicitWaitTimeout(double implicitWaitTimeout) {
298     this.implicitWaitTimeout = implicitWaitTimeout;
299 }
300 public void setMode(WebDriverMode mode) {
301     this.mode = mode;
302 }
303 public void setNtlmAuthTrustedUrIs(String ntlmAuthTrustedUrIs) {
304     this.ntlmAuthTrustedUrIs = ntlmAuthTrustedUrIs;
305 }
306 public void setOperaProfilePath(String operaProfilePath) {
307     this.operaProfilePath = operaProfilePath;
308 }
309 public void setOutputDirectory(String outputDirectory) {
310     this.outputDirectory = outputDirectory;
311 }
312 public void setPageLoadTimeout(int pageLoadTimeout) {
313     this.pageLoadTimeout = pageLoadTimeout;
314 }
315 public void setPlatform(Platform platform) {
316     this.platform = platform;
317 }
318 public void setProxyHost(String proxyHost) {
319     this.proxyHost = proxyHost;
320 }
321 public void setSetAcceptUntrustedCertificates(
322     boolean setAcceptUntrustedCertificates) {
323     this.setAcceptUntrustedCertificates = setAcceptUntrustedCertificates;
324 }
325 }

```

Read-Only Smart Insert 1:1

Java EE Debug

Quick Access

BaseWebUtil.class BrowserType.class CustomEventFiringWebD IScreenshotListener.class ScreenShot.class ScreenShotRemoteWebD ScreenshotUtil.class WebDriverConfig

```

326 public void setAssumeUntrustedCertificateIssuer(
327     boolean setAssumeUntrustedCertificateIssuer) {
328     this.setAssumeUntrustedCertificateIssuer = setAssumeUntrustedCertificateIssuer;
329 }
330 public void setUserAgentOverride(String userAgentOverride) {
331     this.userAgentOverride = userAgentOverride;
332 }
333 public void setWebSessionTimeout(int webSessiontimeout) {
334     this.webSessiontimeout = webSessiontimeout;
335 }
336 public void setAppName(String appName) {
337     this.appName = appName;
338 }
339 public String getAppName() {
340     return appName;
341 }
342 public void setAppVersion(String appVersion) {
343     this.appVersion = appVersion;
344 }
345 public String getAppVersion() {
346     return appVersion;
347 }
348 public int getBrowserWindowWidth() {
349     return browserWindowWidth;
350 }
351 public void setBrowserWindowWidth(int browserWindowWidth) {
352     this.browserWindowWidth = browserWindowWidth;
353 }
354 public int getBrowserWindowHeight() {
355     return browserWindowHeight;
356 }
357 public void setBrowserWindowHeight(int browserWindowHeight) {
358     this.browserWindowHeight = browserWindowHeight;
359 }
360 public void setAndroidTarget(String androidTarget) {
361     this.androidTarget = androidTarget;
362 }
363 public String getAndroidTarget() {
364     return androidTarget;
365 }
366 public void setResolution(String resolution) {
367     this.resolution = resolution;
368 }
369 public String getResolution() {
370     return resolution;
371 }
372 public void setChromeExtensionPath(String chromeExtensionPath) {
373     this.chromeExtensionPath = chromeExtensionPath;
374 }

```

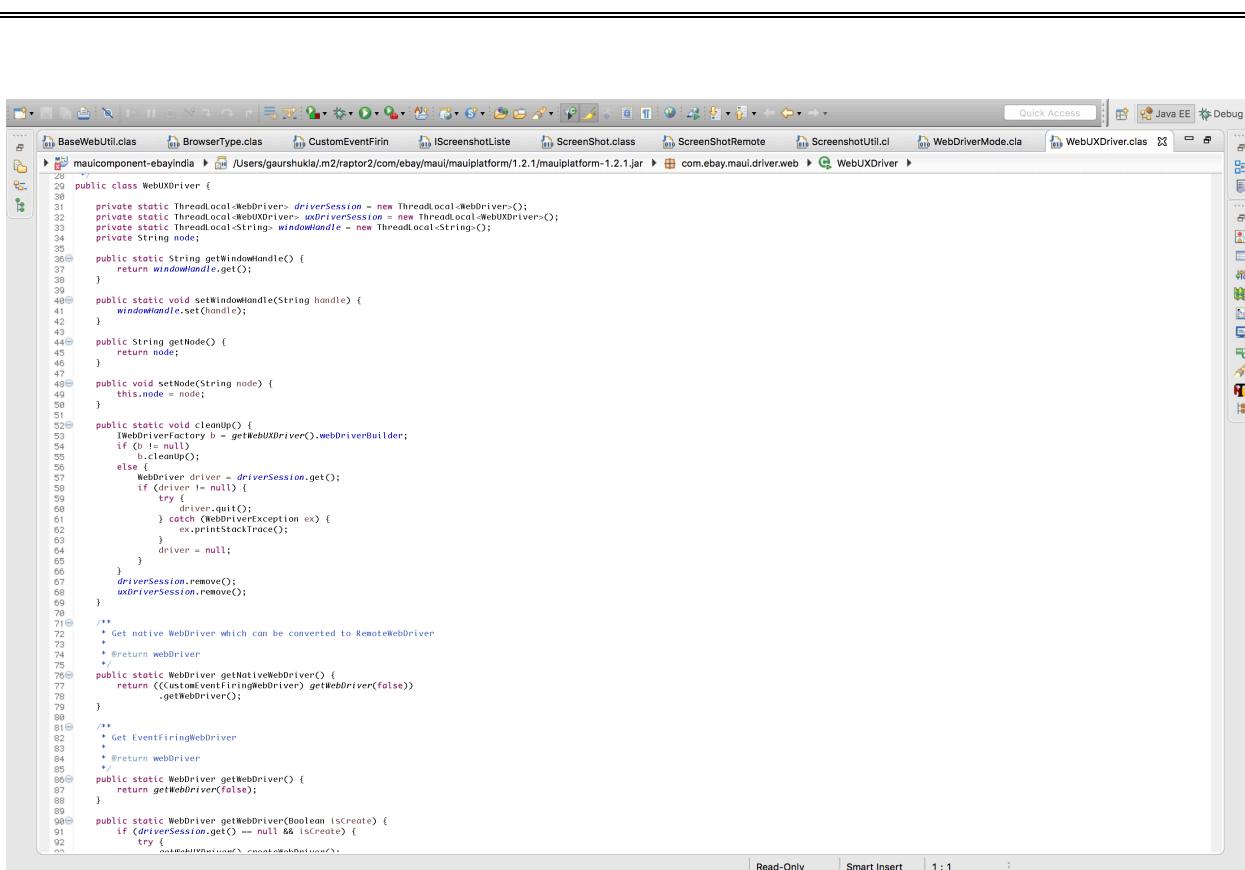
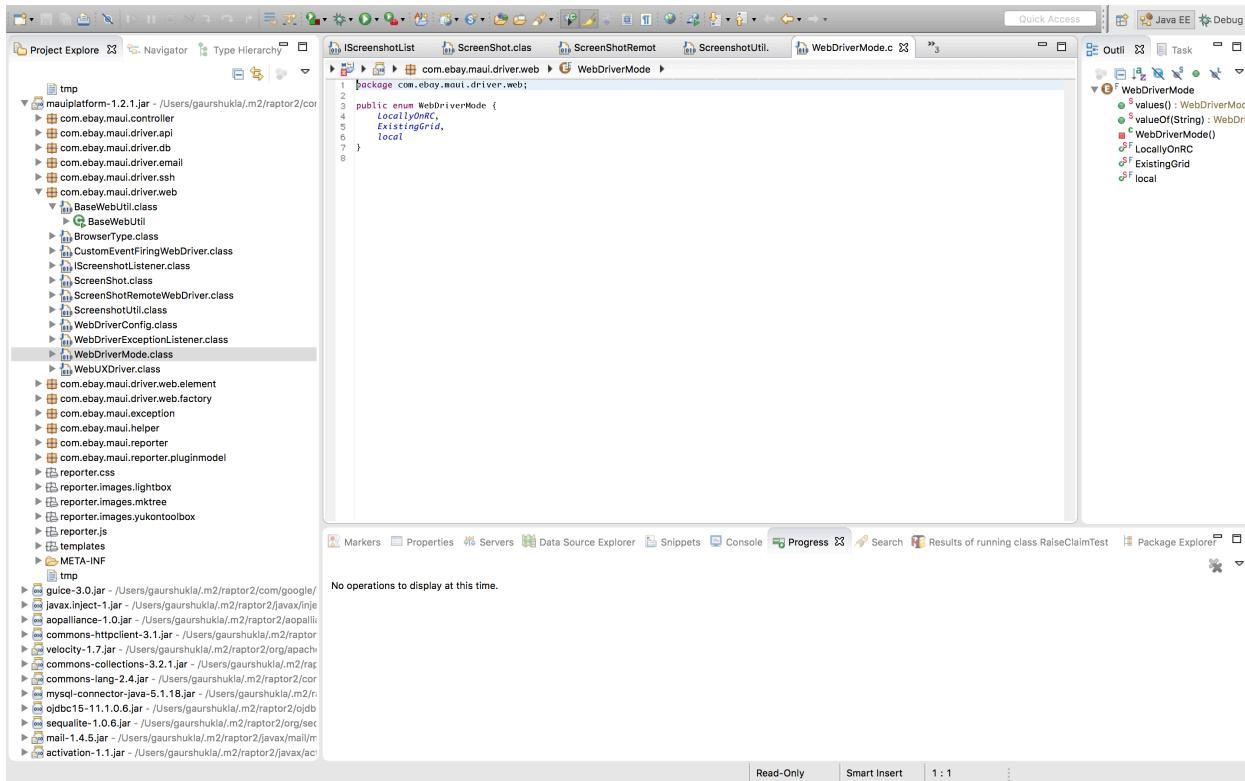
Read-Only Smart Insert 1:1

```
349     public void setAppName(String appName) {
350         this.appName = appName;
351     }
352
353     public String getAppName() {
354         return appName;
355     }
356
357     public void setAppVersion(String appVersion) {
358         this.appVersion = appVersion;
359     }
360
361     public String getAppVersion() {
362         return appVersion;
363     }
364
365     public int getBrowserWindowWidth() {
366         return browserWindowWidth;
367     }
368
369     public void setBrowserWindowWidth(int browserWindowWidth) {
370         this.browserWindowWidth = browserWindowWidth;
371     }
372
373     public int getBrowserWindowHeight() {
374         return browserWindowHeight;
375     }
376
377     public void setBrowserWindowHeight(int browserWindowHeight) {
378         this.browserWindowHeight = browserWindowHeight;
379     }
380
381     public void setAndroidTarget(String androidTarget) {
382         this.androidTarget = androidTarget;
383     }
384
385     public String getAndroidTarget() {
386         return androidTarget;
387     }
388
389     public void setResolution(String resolution) {
390         this.resolution = resolution;
391     }
392
393     public String getResolution() {
394         return resolution;
395     }
396
397     public void setChromedriverPath(String chromedriverPath) {
398         this.chromedriverPath = chromedriverPath;
399     }
400
401     public String getChromedriverPath() {
402         return chromedriverPath;
403     }
404
405     public void setOperadriverPath(String operadriverPath) {
406         this.operadriverPath = operadriverPath;
407     }
408
409     public String getOperadriverPath() {
410         return operadriverPath;
411     }
412 }
```

```
11     public class WebDriverExceptionListener implements WebDriverEventListener {
12
13         public void afterChangeValueOf(WebElement arg0, WebDriver arg1) {
14
15             public void afterClickOn(WebElement arg0, WebDriver arg1) {
16
17             }
18
19             public void afterFindBy(By arg0, WebElement arg1, WebDriver arg2) {
20
21             }
22
23             public void afterNavigateBack(WebDriver arg0) {
24
25             }
26             public void afterNavigateForward(WebDriver arg0) {
27
28             }
29
30             public void afterNavigateTo(String arg0, WebDriver arg1) {
31
32             }
33             public void afterScript(String arg0, WebDriver arg1) {
34
35             }
36             public void beforeChangeValueOf(WebElement arg0, WebDriver arg1) {
37
38             }
39             public void beforeClickOn(WebElement arg0, WebDriver arg1) {
40
41             }
42
43             public void beforeFindBy(By arg0, WebElement arg1, WebDriver arg2) {
44
45             }
46
47             public void beforeNavigateBack(WebDriver arg0) {
48
49             }
50
51             public void beforeNavigateForward(WebDriver arg0) {
52
53             }
54
55             public void beforeNavigateToString(String arg0, WebDriver arg1) {
56
57             }
58
59             public void beforeScript(String arg0, WebDriver arg1) {
60
61             }
62
63             public void onException(Throwable ex, WebDriver arg1) {
64
65                 if (ex.getMessage() == null) {
66                     return;
67                 } else if ((ex.getMessage()).contains(
68                     "Element must be user-editable in order to clear it"))
69                     return;
70                 else if ((ex.getMessage()).contains("Element is not clickable at point"))
71                     return;
72                 else if (ex instanceof UnsupportedCommandException)
73                     return;
74                 else if ((ex.getMessage()).contains(" read-only"))
```

The screenshot shows the Eclipse IDE interface with the Java EE perspective selected. The top menu bar includes 'File', 'Edit', 'Select', 'Run', 'Debug', 'View', 'Search', 'Help', and 'Quick Access'. The left sidebar displays the package structure under 'Java Packages' and lists several Java files: BaseWebUtil.class, BrowserType.class, CustomEventFiringWeb, IScreenshotListener.class, ScreenShot.class, ScreenShotRemoteWebDr, ScreenshotUtil.class, WebDriverExceptionListener.class, and WebDriverExceptionLister.class. The main editor area contains the source code for the WebDriverExceptionListener class, which handles various WebDriver exceptions and performs screenshot captures.

```
74     else if (ex.getMessage().contains("No response on ECMAScript evaluation command")) { // Opera
75         return;
76     }
77     else if (ex.getMessage().contains("Error communicating with the remote browser. It may have died.")) {
78         // Session has lost connection, reuse it then ignore quit() method
79         if (method.contains("getTitle")){
80             // method.contains("getWindowHandle")
81             // method.contains("click")
82             // method.contains("getPageSource"))
83             return;
84         }
85     }
86     ex.printStackTrace();
87     // return;
88 } else if (ex.getMessage().contains("Session timed out")) {
89     // Session has lost connection, reuse it then ignore quit() method
90     if (WebUI0Driver.getWebUI0Driver().getConfig().getMode() == WebDriverMode.ExistingGrid) {
91         WebUI0Driver.setWebIDriver(null);
92     }
93     throw new WebSessionTerminatedException(ex);
94 }
95 return;
96 } else if (ex instanceof org.openqa.selenium.remote.UnreachableBrowserException) {
97     return;
98 } else if (ex instanceof org.openqa.selenium.UnsupportedCommandException) {
99     return;
100 } else {
101     String message = ex.getMessage().split("\n")[0];
102     System.out.println("ExceptionListener:got exception--" + message);
103     if (message.matches("Session (.*) was terminated due to(.*)\\n*")){
104         // message
105         // matches("cannot forward the request Connection to(.*)\\n*"))
106         WebUI0Driver.setWebIDriver(null); // can't quit anymore, save time
107         // since the session was
108         // terminated.
109         throw new WebSessionTerminatedException(ex);
110     }
111     for (int i = 0; i < ex.getStackTrace().length; i++)// avoid dead loop
112     {
113         String method = ex.getStackTrace()[i].getMethodName();
114         if (method.contains("getScreenshotAs")){
115             // method.contains("capture WebElementSnapshot")||method.equals(ignoreCase("until"))
116             return;
117         }
118     }
119 }
120 }
121
122 if (org != null) {
123
124     try {
125         System.out.println("ExceptionListener:got exception--" + ex.getMessage());
126         new ScreenshotUtil(org).capturePageSnapshotInException();
127     } catch (Exception e) {
128         // Ignore all exceptions
129         e.printStackTrace();
130     }
131 }
132 }
133 }
134 }
135
136 public static void main(String[] args) {
137     String ex = "Session [92d6801d-7375-404d-bc32-5633c552b1c0] was terminated due to BROWSER_TIMEOUT"
138 }
```



```

  91     if (driverSession == null) {
  92         try {
  93             driverSession = getWebUXDriver().createWebDriver();
  94         } catch (Exception e) {
  95             System.out.println("Got exception when create web driver");
  96             e.printStackTrace();
  97             throw new RuntimeException(e);
  98         }
  99     }
 100    return driverSession.get();
 101}
 102
 103    public static WebUDriver getWebUXDriver() {
 104        if (uxDriverSession.get() == null) {
 105            uxDriverSession.set(new WebUDriver());
 106        }
 107        return uxDriverSession.get();
 108    }
 109
 110    public static void setWebDriver(WebDriver driver) {
 111        if (driver == null)
 112            driverSession.remove();
 113        else {
 114            if (getWebUDriver() == null)
 115                new WebUDriver();
 116            driverSession.set(driver);
 117        }
 118    }
 119
 120    private WebDriverConfig config = new WebDriverConfig();
 121    private WebDriver driver;
 122    private IWebDriverFactory webDriverBuilder;
 123
 124    public WebUDriver() {
 125        init();
 126        uxDriverSession.set(this);
 127    }
 128
 129    public WebUDriver(String browser, String mode) {
 130        init();
 131        this.setBrowser(browser);
 132        this.setMode(mode);
 133        uxDriverSession.set(this);
 134    }
 135
 136    public WebDriver createRemoteWebDriver(String browser, String mode)
 137        throws Exception {
 138        WebDriver driver = null;
 139        config.setBrowser(BrowserType.getBrowserType(browser));
 140        config.setMode(WebDriverMode.valueOf(mode));
 141
 142        if (config.getMode() == WebDriverMode.ExistingGrid) {
 143            webDriverBuilder = new RemoteDriverFactory(this.config);
 144        } else {
 145            if (config.getBrowser() == BrowserType.FireFox) {
 146                webDriverBuilder = new FirefoxDriverFactory(this.config);
 147            } else if (config.getBrowser() == BrowserType.Chrome) {
 148                webDriverBuilder = new ChromeDriverFactory(this.config);
 149            } else if (config.getBrowser() == BrowserType.InternetExplore) {
 150                webDriverBuilder = new IEDriverFactory(this.config);
 151            } else if (config.getBrowser() == BrowserType.Safari) {
 152                webDriverBuilder = new SafariDriverFactory(this.config);
 153            } else if (config.getBrowser() == BrowserType.Opera) {
 154                webDriverBuilder = new OperaDriverFactory(this.config);
 155            } else if (config.getBrowser() == BrowserType.HTMLUnit) {
 156                webDriverBuilder = new HtmlUnitDriverFactory(this.config);
 157            } else if (config.getBrowser() == BrowserType.IPhone) {
 158                webDriverBuilder = new iPhoneDriverFactory(this.config);
 159            } else if (config.getBrowser() == BrowserType.Android) {
 160                webDriverBuilder = new AndroidDriverFactory(this.config);
 161            } else if (config.getBrowser() == BrowserType.PhantomJS) {
 162                webDriverBuilder = new PhantomJSFactory(this.config);
 163            } else {
 164                throw new RuntimeException(
 165                    "does not support local mode, please use remote mode");
 166            }
 167        }
 168        // synchronized(this.getClass());
 169        driver = webDriverBuilder.createWebDriver();
 170
 171        driver = webDriverBuilder.createWebDriver();
 172
 173        if (config.getBrowserWindowWidth() > 0
 174            && config.getBrowserWindowHeight() > 0) {
 175            new BaseWebUtil(driver).resizeWindow(
 176                config.getBrowserWindowWidth(),
 177                config.getBrowserWindowHeight());
 178        } else {
 179            new BaseWebUtil(driver).maximizeWindow();
 180        }
 181        driver = handleListeners(driver);
 182
 183    }
 184
 185    protected WebDriver handleListeners(WebDriver driver) {
 186        // Support customized listeners
 187        ArrayList<WebDriverEventListener> listeners = config
 188            .getWebDriverEventListeners();
 189        if (listeners != null && listeners.size() > 0) {
 190            for (int i = 0; i < config.getWebDriverEventListeners().size(); i++) {
 191                driver = new CustomEventFiringWebDriver(driver)
 192                    .register(listeners.get(i));
 193            }
 194        }
 195    }
 196
 197    public WebDriver createWebDriver() throws Exception {
 198        System.out.println(Thread.currentThread().getName() + " : new WebDriver");
 199        "Start creating web driver instance:" + this.getBrowser();
 200        driver = createRemoteWebDriver(config.getBrowser(), config
 201            .getMode());
 202
 203        System.out.println(Thread.currentThread().getName() + ":" + new Date()
 204            + "Finish creating web driver instance:" + this.getBrowser());
 205
 206        driverSession.set(driver);
 207        setWindowHandle(new BaseWebUtil(driver).getWindowHandle());
 208        return driver;
 209    }
 210
 211    public String getBrowser() {
 212        return config.getBrowser();
 213    }
 214
 215    public void close() {
 216        uxDriverSession.remove();
 217    }

```



```

 342     config.setFFPath(ffBinPath);
 343     String chromePath = ContextManager.getThreadContext()
 344         .getChromePath();
 345     config.setChromePath(chromePath);
 346     String chromeDriverPath = ContextManager.getThreadContext()
 347         .getChromeDriverPath();
 348     config.setChromeExtensionPath(ContextManager.getThreadContext()
 349         .getChromeExtensionPath());
 350     config.setChromeExtensionPath(ChromeExtensionPath);
 351     String chromeDriverPath = ContextManager.getThreadContext()
 352         .getChromeDriverPath();
 353     config.setIDriverPath(IDriverPath);
 354     String operaDriverPath = ContextManager.getThreadContext()
 355         .getOperaDriverPath();
 356     config.setOperaDriverPath(operaDriverPath);
 357     int webSessionTimeout = ContextManager.getThreadContext()
 358         .getWebSessionTimeout();
 359     config.setWebSessionTimeout(webSessionTimeout);
 360     double implicitWaitTimeout = ContextManager.getThreadContext()
 361         .getImplicitWaitTimeout();
 362     config.setImplicitWaitTimeout(implicitWaitTimeout);
 363     config.setExplicitWaitTimeout(explicitWaitTimeout);
 364     int explicitWaitTimeout = ContextManager.getThreadContext()
 365         .getExplicitWaitTimeout();
 366     config.setExplicitWaitTimeout(explicitWaitTimeout);
 367     config.setPageLoadTimeout(ContextManager.getThreadContext()
 368         .getPageLoadTimeout());
 369     String outputDirectory = ContextManager.getGlobalContext()
 370         .getOutputDirectory();
 371     config.setOutputDirectory(outputDirectory);
 372     if (ContextManager.getThreadContext().isWebProxyEnabled()) {
 373         String proxyHost = ContextManager.getThreadContext()
 374             .getWebProxyAddress();
 375         config.setProxyHost(proxyHost);
 376     }
 377 }
 378
 379 if (*false*
 380     .equalsIgnoreCase((String) ContextManager
 381         .getThreadContext()
 382         .getAttribute(Context.SET_ASSUME_UNTRUSTED_CERTIFICATE_ISSUER))) {
 383     config.setSetAssumeUntrustedCertificateIssuer(false);
 384 }
 385 if (*false*
 386     .equalsIgnoreCase((String) ContextManager.getThreadContext()
 387         .getAttribute(Context.SET_ACCEPT_UNTRUSTED_CERTIFICATES))) {
 388     config.setAcceptUntrustedCertificates(false);
 389 }
 390 if (*false*
 391     .equalsIgnoreCase((String) ContextManager.getThreadContext()
 392         .getAttribute(Context.ENABLE_JAVASCRIPT))) {
 393     config.setEnableJavascript(false);
 394 }
 395 if (*false*
 396     .equalsIgnoreCase((String) ContextManager.getThreadContext()
 397         .getAttribute(Context.ENABLE_COOKIE))) {
 398     config.setEnableCookie(false);
 399 }
 400 if (ContextManager.getThreadContext().getNtLMAuthTrustedUrls() != null)
 401     config.setNtLMAuthTrustedUrls(ContextManager.getThreadContext()
 402         .getNtLMAuthTrustedUrls());
 403 if (ContextManager.getThreadContext().getBrowserDownloadDir() != null)
 404     config.setBrowserDownloadDir(ContextManager.getThreadContext()
 405         .getBrowserDownloadDir());
 406 if (ContextManager.getThreadContext().getAddJSErrorHandlerExtension() != null)
 407     config.setAddJSErrorHandlerExtension(Boolean
 408         .parseBoolean(ContextManager.getThreadContext()
 409             .getAddJSErrorHandlerExtension())));
 410 if (ContextManager.getThreadContext().getPCSettings() != null) {
 411     if (ContextManager.getThreadContext().getUserAgent() != null) {
 412         ua = ContextManager.getThreadContext().getPCSettings() + " "
 413             + ContextManager.getThreadContext().getUserAgent();
 414     } else {
 415         ua = ContextManager.getThreadContext().getPCSettings();
 416     }
 417 } else {
 418     if (ContextManager.getThreadContext().getUserAgent() != null) {
 419         ua = ContextManager.getThreadContext().getUserAgent();
 420     } else {
 421         ua = null;
 422     }
 423 }
 424 config.setUserAgent(ua);
 425 String listeners = ContextManager.getThreadContext()
 426     .getWebDriverListeners();
 427 if (ContextManager.getThreadContext().isExceptionListenerEnabled()) {
 428     if (listeners != null) {
 429         listeners = listeners + ",";
 430     } else {
 431         listeners = "";
 432     }
 433     listeners += null && !listeners.equals("") ?
 434         config.setWebDriverListeners(listeners);
 435     else
 436         config.setWebDriverListeners("");
 437 config.setSafeFirefoxProfile(ContextManager.getThreadContext()
 438     .isSafeFirefoxDefaultProfile());
 439 config.setAppVersion(ContextManager.getThreadContext().getAppName());
 440 config.setAppVersion(ContextManager.getThreadContext(). getAppVersion());
 441 config.setAndroidTarget(ContextManager.getThreadContext()
 442     .getAndroidTarget());
 443 config.setBrowserWidth(ContextManager.getThreadContext().getResolution());
 444 String size = ContextManager.getThreadContext().getBrowserWindowSize();
 445 if (size != null) {
 446     int width = -1;
 447     int height = -1;
 448     try {
 449         width = Integer.parseInt(size.split(",")[0].trim());
 450         height = Integer.parseInt(size.split(",")[1].trim());
 451     } catch (Exception ex) {
 452     }
 453     config.setBrowserWidth(width);
 454     config.setBrowserHeight(height);
 455 }
 456
 457 public static void main(String[] args) {
 458     System.out.println(WebDriverExceptionListener.class.getName());
 459 }
 460
 461 public boolean isSetAcceptUntrustedCertificates() {
 462     return config.isSetAcceptUntrustedCertificates();
 463 }
 464
 465 public boolean isAddJSErrorHandlerExtension() {
 466     return config.isAddJSErrorHandlerExtension();
 467 }

```

Java EE Debug

maucOMPONENT-eBayIndia > /Users/gaurshukla/m2/raptor2/com/ebay/maui/maulplatform/1.2.1/maulplatform-1.2.1.jar > com.ebay.maui.driver.web > WebUXDriver

```

177}
178}
179}
180}
181}
182}
183}
184}
185}
186}
187}
188}
189}
190}
191}
192}
193}
194}
195}
196}
197}
198}
199}
200}
201}
202}
203}
204}
205}
206}
207}
208}
209}
210}
211}
212}
213}
214}
215}
216}
217}
218}
219}
220}
221}
222}
223}
224}
225}
226}
227}
228}
229}
230}
231}
232}
233}
234}
235}
236}
237}
238}
239}
240}
241}
242}
243}
244}
245}
246}
247}
248}
249}
250}
251}
252}
253}
254}
255}
256}
257}
258}
259}
260}
261}
262}
263}
264}
265}
266}
267}
268}
269}
270}
271}
272}
273}
274}
275}
276}
277}
278}
279}
280}
281}
282}
283}
284}
285}
286}
287}
288}
289}
290}
291}
292}
293}
294}
295}
296}
297}
298}
299}
300}
301}
302}
303}
304}
305}
306}
307}
308}
309}
310}
311}
312}
313}
314}
315}
316}
317}
318}
319}
320}
321}
322}
323}
324}
325}
326}
327}
328}
329}
330}
331}
332}
333}
334}
335}
336}
337}
338}
339}
340}
341}
342}
343}
344}
345}
346}
347}
348}
349}
350}
351}
352}
353}
354}
355}
356}
357}
358}
359}
360}
361}
362}
363}
364}
365}
366}
367}
368}
369}
370}
371}
372}
373}
374}
375}
376}
377}
378}
379}
380}
381}
382}
383}
384}
385}
386}
387}
388}
389}
390}
391}
392}
393}
394}
395}
396}
397}
398}
399}
400}
401}
402}
403}
404}
405}
406}
407}
408}
409}
410}
411}
412}
413}
414}
415}
416}
417}
418}
419}
420}
421}
422}
423}
424}
425}
426}
427}
428}
429}
430}
431}
432}
433}
434}
435}
436}
437}
438}
439}
440}
441}
442}
443}
444}
445}
446}
447}
448}
449}
450}
451}
452}
453}
454}
455}
456}
457}
458}
459}
460}
461}
462}
463}
464}
465}
466}
467}
468}
469}
470}
471}
472}
473}
474}
475}
476}
477}
478}
479}
480}
481}
482}
483}
484}
485}
486}
487}
488}
489}
490}
491}
492}
493}
494}
495}
496}
497}
498}
499}
500}
501}
502}
503}
504}
505}
506}
507}
508}
509}
510}
511}
512}
513}
514}
515}
516}
517}
518}
519}
520}
521}
522}
523}
524}
525}
526}
527}
528}
529}
530}
531}
532}
533}
534}
535}
536}
537}
538}
539}
540}
541}
542}
543}
544}
545}
546}
547}
548}
549}
550}
551}
552}
553}
554}
555}
556}
557}
558}
559}
560}
561}
562}
563}
564}
565}
566}
567}
568}
569}
570}
571}
572}
573}
574}
575}
576}
577}
578}
579}
580}
581}

```

Read-Only Smart Insert 1 : 1

Java EE Debug

maucOMPONENT-eBayIndia > /Users/gaurshukla/m2/raptor2/com/ebay/maui/maulplatform/1.2.1/maulplatform-1.2.1.jar > com.ebay.maui.driver.web > WebUXDriver

```

518}
519}
520}
521}
522}
523}
524}
525}
526}
527}
528}
529}
530}
531}
532}
533}
534}
535}
536}
537}
538}
539}
540}
541}
542}
543}
544}
545}
546}
547}
548}
549}
550}
551}
552}
553}
554}
555}
556}
557}
558}
559}
560}
561}
562}
563}
564}
565}
566}
567}
568}
569}
570}
571}
572}
573}
574}
575}
576}
577}
578}
579}
580}
581}

```

Read-Only Smart Insert 1 : 1

## Web.element

```

1  package com.ebay.maulplatform;
2
3  import org.openqa.selenium.WebDriver;
4  import org.openqa.selenium.WebElement;
5  import org.openqa.selenium.support.ui.ExpectedConditions;
6  import org.openqa.selenium.support.ui.WebDriverWait;
7  import org.openqa.selenium.Alert;
8
9  public abstract class BaseHtmlPage {
10
11    protected WebDriver driver = WebUDriver.getWebDriver();
12    protected final WebUDriver webUDriver = WebUDriver.getWebUDriver();
13    private int explicitWaitTimeout = WebUDriver.getExplicitWait();
14    private int getExplicitWait();
15    private int sessionTimeout = WebUDriver.getSessionTimeout();
16    private getSessionTimeout();
17
18    public BaseHtmlPage() {
19    }
20
21    public void acceptAlert() throws PageNotCurrentException {
22      Alert alert = driver.switchTo().alert();
23      alert.accept();
24      driver.switchTo().defaultContent();
25    }
26
27    public void assertAlertPresent() {
28      Logging.logWebStep(null, "assert alert present.", false);
29      try {
30        driver.switchTo().alert();
31      } catch (Exception e) {
32        // AlertOverride alertOverride = new AlertOverride();
33        // assertHTML(alertOverride.isAlertPresent(driver),
34        // "assert alert present.");
35        // assertAlertHTML(false, "assert alert present.");
36      }
37    }
38
39    public void assertAlertText(String text) {
40      Logging.logWebStep(null, "assert " + element.toHTML() + " attribute=" +
41        attributeName + ", expectedValue=" + value + ")", false);
42
43      String attributeValue = element.getAttribute(attributeName);
44
45      assertEquals(text != null && value.equals(attributeValue),
46        element.toString() + " attribute=" + attributeName
47        + ", expectedValue = " + value + ")"
48        + ", attributeValue = (" + attributeValue + ")");
49    }
50
51    public void assertAttributeContains(HtmlElement element,
52      String attributeName, String keyword) {
53      Logging.logWebStep(null, "assert " + element.toHTML() + " attribute=" +
54        attributeName + ", contains keyword = (" + keyword + ")", false);
55
56      String attributeValue = element.getAttribute(attributeName);
57
58    }
59
60    /**
61     * Assert to check attribute value. Just pass in the attribute name and
62     * expected value.
63     *
64     * @param element
65     * @param attributeName
66     * @param attributeValue
67     */
68    public void assertAttribute(HtmlElement element, String attributeName,
69      String attributeValue) {
70      Logging.logWebStep(null, "assert " + element.toHTML() + " attribute=" +
71        attributeName + ", expectedValue=" + attributeValue + ")", false);
72
73      String attributeValue = element.getAttribute(attributeName);
74
75      assertEquals(attributeValue != null && value.equals(attributeValue),
76        element.toString() + " attribute=" + attributeName
77        + ", expectedValue = " + attributeValue + ")"
78        + ", attributeValue = (" + attributeValue + ")");
79    }
80
81    public void assertAttributeNotContains(HtmlElement element,
82      String attributeName, String keyword) {
83      Logging.logWebStep(null, "assert " + element.toHTML() + " attribute=" +
84        attributeName + ", does not contain keyword = (" + keyword + ")", false);
85
86      String attributeValue = element.getAttribute(attributeName);
87
88    }
89
90    public void assertConfirmationText(String text) {
91      Logging.logWebStep(null, "assert confirmation text.", false);
92      Alert alert = driver.switchTo().alert();
93      String seenText = alert.getText();
94
95      assertAlertHTML(seenText.contains(text), "assert confirmation text.");
96    }
97
98    protected void assertCurrentPage(boolean log)
99      throws PageNotCurrentException {
100      // do nothing
101    }
102
103    public void assertElementNotPresent(HtmlElement element) {
104      Logging.logWebStep(null, "assert " + element.toHTML() +
105        " is not present.", false);
106      assertHTML(element.isElementPresent(), element.toString() + " found.");
107    }
108
109    public void assertElementPresent(HtmlElement element) {
110      Logging.logWebStep(null, "assert " + element.toHTML() + " is present.",
111        "is not found.");
112      assertHTML(element.isElementPresent(), element.toString());
113    }
114
115    void assertHTML(boolean condition, String message) {
116      if (!condition) {
117        capturePageSnapshot();
118        Assertion.assertTrue(condition, message);
119      }
120    }
121
122    void assertAlertHTML(boolean condition, String message) {
123      if (!condition) {
124        if (condition) {
125          Assertion.assertTrue(condition, message);
126        }
127      }
128    }
129
130    @Deprecated
131  }

```

```

161@Deprecated
162    public void assertLabelPresent(Label label) {
163        Logging.logWebStep(null, "assert " + label.getLabel() + " is present.", false);
164        assertEquals(true, label.isLabelPresent());
165        assertEquals(label.getLabel(), label.getExpectedText() + " Found.");
166    }
167
168@Deprecated
169    public void assertLabelAbsent(Label label) {
170        Logging.logWebStep(null, "assert " + label.getLabel() + " is absent.", false);
171        assertEquals(false, label.isLabelPresent());
172        assertEquals(label.getLabel(), label.getExpectedText() + " not found.");
173    }
174
175
176    public void assertPromptText(String text) {
177        Logging.logWebStep(null, "assert prompt text.", false);
178        Alert alert = driver.switchTo().alert();
179        String seemText = alert.getText();
180        assertEquals(text, seemText);
181        assertEquals("alert", seemText.contains(text));
182    }
183
184    public void assertTable(Table table, int row, int col, String text) {
185        Logging.logWebStep(null, "assert text " + text + " equals " + table.toHTML());
186        assertEquals(" at (" + row + "," + col + ") = (" + row + "," + col + ")", false);
187        String content = table.getContent(row, col);
188        assertEquals(content != null && content.equals(text), "Text (" + text +
189            " ) not found " + table.toString());
190        assertEquals(" at cell(" + row + "," + col + ") = (" + row + "," + col + ")");
191    }
192
193
194    public void assertTableContains(Table table, int row, int col, String text) {
195        Logging.logWebStep(null, "assert text " + text + " contains " +
196            " at (" + row + "," + col + ") = (" + row + "," + col + ")", false);
197        String content = table.getContent(row, col);
198        assertEquals(content != null && content.contains(text), "Text (" + text +
199            " ) not found " + table.toString());
200        assertEquals(" at cell(" + row + "," + col + ") = (" + row + "," + col + ")");
201    }
202
203
204    public void assertTableMatches(Table table, int row, int col, String text) {
205        Logging.logWebStep(null, "assert text " + text + " matches " +
206            " at (" + row + "," + col + ") = (" + row + "," + col + ")", false);
207        String content = table.getContent(row, col);
208        assertEquals(content != null && content.matches(text), "Text (" + text +
209            " ) not found " + table.toString());
210        assertEquals(" at cell(" + row + "," + col + ") = (" + row + "," + col + ")");
211    }
212
213
214    ////////////////////////////////////////////////////////////////// assertion methods //////////////////////////////////////////////////////////////////
215
216    public void assertTextNotPresent(String text) {
217        Logging.logWebStep(null, "assert text " + text + " is not present.", false);
218        assertEquals(false, isTextPresent(text), "Text (" + text + ") Found.");
219    }
220
221    public void assertTextNotPresentIgnoreCase(String text) {
222        Logging.logWebStep(null, "assert text " + text +
223            " is not present.(ignore case).", false);
224        assertEquals(false, getBodyText().toLowerCase().contains(text.toLowerCase()),
225            "Text (" + text + ") Found.");
226    }
227
228
229    public void assertTextPresent(String text) {
230        Logging.logWebStep(null, "assert text " + text + " is present.", false);
231        assertEquals(true, isTextPresent(text), "Text (" + text + ") not found.");
232    }
233
234
235    public void assertTextPresentIgnoreCase(String text) {
236        Logging.logWebStep(null, "assert text " + text +
237            " is present.(ignore case).", false);
238        assertEquals(true, getBodyText().toLowerCase().contains(text.toLowerCase()),
239            "Text (" + text + ") not found.");
240    }
241
242    public String cancelConfirmation() throws PageNotCurrentException {
243        Alert alert = driver.switchTo().alert();
244        String seemText = alert.getText();
245        alert.dismiss();
246        driver.switchTo().defaultContent();
247        return seemText;
248    }
249
250    protected abstract void capturePageSnapshot();
251
252
253    ////////////////////////////////////////////////////////////////// get dialog methods //////////////////////////////////////////////////////////////////
254    public Alert getAlert() {
255        Alert alert = driver.switchTo().alert();
256        return alert;
257    }
258
259    public String getLabelText() {
260        Alert alert = driver.switchTo().alert();
261        String seemText = alert.getText();
262        return seemText;
263    }
264
265    private String getBodyText() {
266        WebElement body = driver.findElement(By.tagName("body"));
267        return body.getText();
268    }
269
270    public String getConfirmation() {
271        Alert alert = driver.switchTo().alert();
272        String seemText = alert.getText();
273        return seemText;
274    }
275
276    public WebDriver getDriver() {
277        driver = WebDDriver.getWebDriver();
278        return driver;
279    }
280
281
282    public String getKeyValue(String key) {
283        String site = null;
284        if (ContextManager.getThreadContext().getGBSite() != null) {
285            site = ContextManager.getThreadContext().getGBSite();
286        } else {
287            site = ContextManager.getThreadContext().getSite();
288        }
289    }

```

The screenshot shows the Java code for `BaseWebUtil.java` in an IDE. The code is part of the `mauicomponent-ebayindia` project. The code handles various UI interactions like switching contexts, getting text from alerts, and performing waits.

```
...  
285     if (ContextManager.getThreadContext().getGUISite() == null) {  
286         site = ContextManager.getThreadContext().getGHSite();  
287     } else {  
288         site = ContextManager.getThreadContext().getSite();  
289     }  
290     return Keyword.getKeyword(this.getClass(), key, site);  
291 }  
292  
293 ////////////////////////////////////////////////////////////////// Wait Methods //////////////////////////////////////////////////////////////////  
294  
295     public String getPromptO () {  
296         Alert alert = driver.switchTo().alert();  
297         String seenText = alert.getText();  
298         return seenText;  
299     }  
300  
301     public boolean isElementPresent(By by) {  
302         return new HtmlElement(by).isElementPresent();  
303     }  
304  
305     public boolean isFrame() {  
306         return false;  
307     }  
308  
309     public boolean isTextPresent(String text) {  
310         Assertion.assertNotNull(text, "isTextPresent: text should not be null");  
311         driver = WebDDriver.getWebDriver();  
312         WebElement body = driver.findElement(By.tagName("body"));  
313  
314         if (WebDDriver.getWebDriver().getBrowserType().equalsIgnorecase(BrowserType.HTMLUnit.getType())) {  
315             return body.getText().contains(text);  
316         }  
317         Boolean result = false;  
318         if (body.getText().contains(text))  
319             return true;  
320  
321         JavascriptLibrary js = new JavascriptLibrary();  
322         String script = js.getSelenuimScript("isTextPresent.js");  
323         result = (Boolean) ((JavascriptExecutor) driver).executeScript(  
324             "return (" + script + ")(arguments[0]);", text);  
325  
326         // Handle the null case  
327         if (result == null)  
328             return Boolean.TRUE == result;  
329     }  
330  
331     public void selectFrame(By by) {  
332         Logging.logWebStep(null, "select frame, locator=" + by.toString()  
333             + ")", false);  
334         driver.switchTo().frame(driver.findElement(by));  
335     }  
336  
337     /**  
338      * If current window is closed, you can't use this method to switch to  
339      * another window. Please use driver.switchTo.window(handle) to handle this  
340      * situation. You need to capture the handle you need to switch before this  
341      * statement.  
342      * @param windowName  
343     */  
344 }
```

The screenshot shows the Java code for `BaseWebUtil.java` in an IDE. The code is part of the `mauicomponent-ebayindia` project. It contains several methods for interacting with windows, frames, and elements, including waits and assertions.

```
...  
353     public final void selectWindow(String windowName) {  
354         if (windowName == null) {  
355             Windows windows = new Windows(driver);  
356             try {  
357                 windows.selectBlankWindow(driver);  
358             } catch (SeleniumException e) {  
359                 driver.switchTo().defaultContent();  
360             }  
361         } else {  
362             Windows windows = new Windows(driver);  
363             windows.selectWindow(driver, "name=" + windowName);  
364             windows.waitForSeconds(1);  
365             // Check whether it's the expected page.  
366             // assertCurrentPage(true);  
367         }  
368     }  
369  
370     public void waitForElementChecked(HtmlElement element) {  
371         Assertion.assertNotNull(element, "Element can't be null");  
372         Logging.logWebStep(null, "wait for " + element.toString()  
373             + " to be checked", false);  
374         WebDriverWait wait = new WebDriverWait(driver, explicitWaitTimeout);  
375         wait.until(ExpectedConditions.elementToBeSelected(element.getBy()));  
376     }  
377  
378     public void waitForElementEditable(HtmlElement element) {  
379         Assertion.assertNotNull(element, "Element can't be null");  
380         Logging.logWebStep(null, "wait for " + element.toString()  
381             + " to be editable", false);  
382         WebDriverWait wait = new WebDriverWait(driver, explicitWaitTimeout);  
383         wait.until(ExpectedConditions.elementToBeClickable(element.getBy()));  
384     }  
385  
386     public void waitForElementPresent(By by) {  
387         Logging.logWebStep(null, "wait for " + by.toString()  
388             + " to be present", false);  
389         WebDriverWait wait = new WebDriverWait(driver, explicitWaitTimeout);  
390         wait.until(ExpectedConditions.presenceOfElementLocated(by));  
391     }  
392  
393     public void waitForElementPresent(By by, int timeout) {  
394         Logging.logWebStep(null, "wait for " + by.toString()  
395             + " to be present", false);  
396         WebDriverWait wait = new WebDriverWait(driver, timeout);  
397         wait.until(ExpectedConditions.presenceOfElementLocated(by));  
398     }  
399  
400     public void waitForElementPresent(HtmlElement element) {  
401         Assertion.assertNotNull(element, "Element can't be null");  
402         Logging.logWebStep(null, "wait for " + element.toString()  
403             + " to be present", false);  
404         WebDriverWait wait = new WebDriverWait(driver, explicitWaitTimeout);  
405         wait.until(ExpectedConditions.presenceOfElementLocated(element.getBy()));  
406     }  
407  
408     public void waitForElementToBeVisible(HtmlElement element) {  
409         Assertion.assertNotNull(element, "Element can't be null");  
410         Logging.logWebStep(null, "wait for " + element.toString()  
411             + " to be visible", false);  
412         WebDriverWait wait = new WebDriverWait(driver, explicitWaitTimeout);  
413         wait.until(ExpectedConditions.visibilityOfElementLocated(element.  
414             getBy()));  
415     }  
416 }
```

Screenshot 1 (Top):

```

147 }
148     public void waitForElementToDisappear(HtmlElement element) {
149         Assert.assertNotNull(element, "Element can't be null");
150         Logging.LogError("Wait for " + element.toString()
151             + " to disappear.", false);
152         WebDriverWait wait = new WebDriverWait(driver, explicitWaitTimeout);
153         wait.until(ExpectedConditions.invisibilityOfElementLocated(element
154             .getBy()));
155     }
156 }
157     public void waitForPopup(String locator) throws Exception {
158         waitForPopup(locator, sessionTimeout + "");
159     }
160 }
161     public void waitForPopup(final String windowID, String timeout) {
162         final long millis = Long.parseLong(timeout);
163         final String current = driver.getWindowHandle();
164         final Windows windows = new Windows(driver);
165
166         if (ewebDriver.getConfig().getBrowser() == BrowserType.InternetExplore)
167             waitForSeconds();
168
169         new Wait() {
170             @Override
171             public boolean until() {
172                 try {
173                     if ("_blank".equals(windowID)) {
174                         windows.selectBlankWindow(driver);
175                     } else {
176                         driver.switchTo().window(windowID);
177                     }
178                     return !popup.equals(driver.getCurrentUrl());
179                 } catch (TimeoutException e) {
180                     // Swallow
181                 } catch (NoSuchWindowException e) {
182                     // ignore
183                 }
184                 return false;
185             }
186         }.until(String.format("Timed out waiting for %s. Waited %s", windowID,
187             timeout), millis);
188
189         driver.switchTo().window(current);
190     }
191 }
192 /**
193 * Wait for seconds. Provide a value less than WebSessionTimeout i.e. 180
194 * Seconds
195 * @param seconds
196 */
197 protected void waitForSeconds(int seconds) {
198     ThreadHelper.waitForSeconds(seconds);
199 }
200
201     public void waitForTextPresent(HtmlElement element, String text) {
202         Assert.assertNotNull(element, "Text can't be null");
203         Logging.LogError("Wait for text \'" + text
204             + "\' to be present.", false);
205         WebDriverWait wait = new WebDriverWait(driver, explicitWaitTimeout);
206         wait.until(ExpectedConditions.textToBePresentInElementLocated(element.getBy(),
207             text));
208     }
209 }

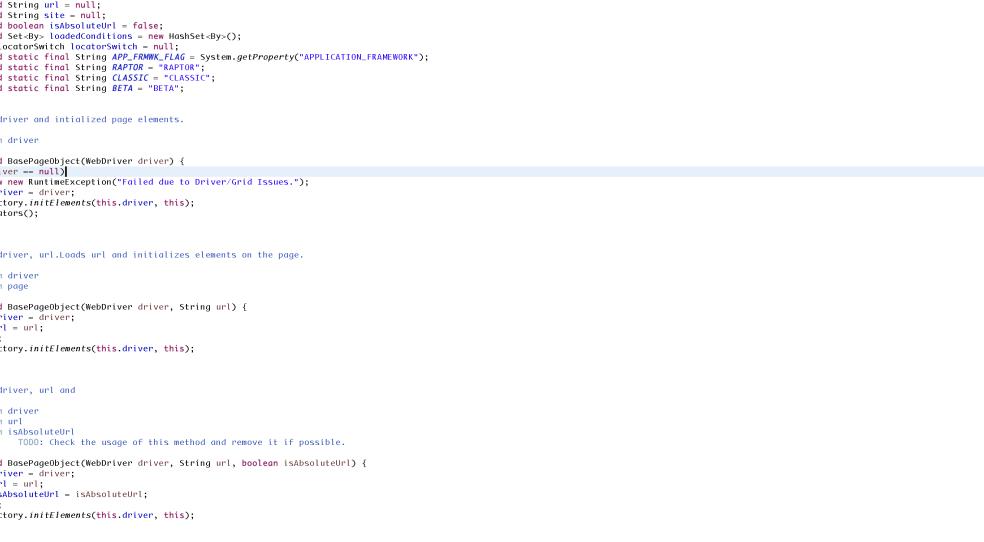
```

Screenshot 2 (Bottom):

```

464     * Wait for Seconds. Provide a value less than WebSessionTimeout i.e. 180
465     * Seconds
466     *
467     * @param seconds
468 */
469 protected void waitForSeconds(int seconds) {
470     ThreadHelper.waitForSeconds(seconds);
471 }
472
473     public void waitForTextPresent(HtmlElement element, String text) {
474         Assert.assertNotNull(element, "Text can't be null");
475         Logging.LogError("Wait for text \'" + text
476             + "\' to be present.", false);
477         WebDriverWait wait = new WebDriverWait(driver, explicitWaitTimeout);
478         wait.until(ExpectedConditions.textToBePresentInElementLocated(element.getBy(),
479             text));
480     }
481
482     public void waitForTextPresent(String text) {
483         Assert.assertNotNull(text, "Text can't be null");
484         Logging.LogError("Wait for text \'" + text
485             + "\' to be present.", false);
486         boolean b = false;
487         for (long millisec = 0; millisec < explicitWaitTimeout * 1000; millisec += 1000) {
488             try {
489                 if (!isTextPresent(text))
490                     b = true;
491                 break;
492             } catch (Exception ignore) {
493             }
494             try {
495                 Thread.sleep(1000);
496             } catch (InterruptedException e) {
497                 throw new RuntimeException(e.getMessage());
498             }
499         }
500         assertHTML(b, "Timed out waiting for text \'" + text
501             + "\' to be there.");
502     }
503
504     public void waitForTextToDisappear(String text) {
505         Assert.assertNotNull(text, "Text can't be null");
506         Logging.LogError("Wait for text \'" + text + "\' to disappear.", false);
507         boolean textPresent = true;
508         for (long millisec = 0; millisec < explicitWaitTimeout * 1000; millisec += 1000) {
509             try {
510                 if (!isTextPresent(text))
511                     textPresent = false;
512                 break;
513             } catch (Exception ignore) {
514             }
515             try {
516                 Thread.sleep(1000);
517             } catch (InterruptedException e) {
518                 throw new RuntimeException(e.getMessage());
519             }
520         }
521         assertHTML(!textPresent, "Timed out waiting for text \'" + text
522             + "\' to be gone.");
523     }
524 }
525
526 }
527 }
528

```



The screenshot shows an IDE interface with multiple tabs open. The current tab displays the Java code for the `BasePageObject` class. The code is annotated with Javadoc-style comments and includes imports for `java.util`, `com.mauli.platform`, `com.mauli.driver.web.element`, and `com.mauli.driver.web.page`. The class implements `LoadableComponent` and `PageObject`, and extends `BasePageObject`. It contains methods for setting the driver, URL, and page elements, as well as a constructor and various annotations like `@Page`, `@PageUrl`, and `@PageMethod`.

```
28 public abstract class BasePageObject extends LoadableComponent<BasePageObject> {
29     ...
30     private static final long TIMEOUT_IN_SECONDS = 10;
31     protected WebDriver driver = null;
32     protected String url = null;
33     protected PageObject pageObject;
34     protected boolean isAbsoluteUrl = false;
35     protected Set<By> loadedConditions = new HashSet<By>();
36     private LocatorWithLocationWithMatch match;
37     protected final String APPLICATION_FRAMEWORK = System.getProperty("APPLICATION_FRAMEWORK");
38     protected static final String RAPTOR = "RAPTOR";
39     protected static final String CLASSIC = "CLASSIC";
40     protected static final String BETA = "BETA";
41     ...
42     /**
43      * Sets driver and initialized page elements.
44      * @param driver
45      */
46     protected BasePageObject(WebDriver driver) {
47         if (driver == null) {
48             throw new RuntimeException("Failed due to Driver/Grid Issues.");
49         }
50         this.driver = driver;
51         PageFactory.initElements(this.driver, this);
52         setLocators();
53     }
54     ...
55     /**
56      * Sets driver, url Loads url and initializes elements on the page.
57      * @param driver
58      * @param url
59      */
60     protected BasePageObject(WebDriver driver, String url) {
61         this.driver = driver;
62         this.url = url;
63         load();
64         PageFactory.initElements(this.driver, this);
65     }
66     ...
67     /**
68      * Sets driver, url and
69      * @param driver
70      * @param url
71      */
72     protected BasePageObject(WebDriver driver, String url, boolean isAbsoluteUrl) {
73         this.driver = driver;
74         this.url = url;
75         this.isAbsoluteUrl = isAbsoluteUrl;
76         ...
77         /**
78          * Add a constraint that must be met in order for the page to be considered
79          * loaded. This should be used by implementing page objects in a domain
80          * specific manner.
81          */
82         PageFactory.initElements(this.driver, this);
83     }
84     ...
85     /**
86      * Add a constraint that must be met in order for the page to be considered
87      * loaded. This should be used by implementing page objects in a domain
88      * specific manner.
89      */
90     protected void matchCondition() {
91         ...
92     }
93 }
```

The screenshot shows the Eclipse IDE interface with the Java perspective active. The title bar includes tabs for 'Read-Only' and 'Smart Insert'. The top menu bar has options like 'File', 'Edit', 'Search', 'Run', 'View', 'Project', 'Tools', 'Help', and 'Quick Access'. The toolbar contains icons for file operations like Open, Save, and Print, as well as Java EE specific icons. The left side features the 'Navigator' view showing project structure and the 'Outline' view showing the class hierarchy. The main editor area displays the Java code for the `BasePageObject` class, which extends `LoadableComponent<BasePageObject>`. The code includes various protected fields and methods for interacting with a WebDriver, managing page elements, and handling URLs.

```
28  public abstract class BasePageObject extends LoadableComponent<BasePageObject> {
29
30     private static final long TIMEOUT_IN_SECONDS = 10;
31
32     protected WebDriver driver = null;
33     protected String url = null;
34     protected String siteUrl;
35     protected boolean isAbsoluteUrl = false;
36     protected Set<By> loadedCondition = new HashSet<By>();
37     private Locator locator;
38
39     protected static final String APP_FRAME_FLAG = System.getProperty("APPLICATION_FRAMEWORK");
40     protected static final String RAPTOR = "RAPTOR";
41     protected static final String CLASSIC = "CLASSIC";
42     protected static final String BETA = "BETA";
43
44     /**
45      * Sets driver and initialized page elements.
46      *
47      * @param driver
48      */
49     protected BasePageObject(WebDriver driver) {
50         if (driver == null) {
51             throw new RuntimeException("Failed due to Driver/Grid Issues.");
52         }
53         this.driver = driver;
54         Pagefactory.initElements(this.driver, this);
55         setLocators();
56     }
57
58     /**
59      * Sets driver, url Loads url and initializes elements on the page.
60      *
61      * @param driver
62      * @param page
63      */
64     protected BasePageObject(WebDriver driver, String url) {
65         this.driver = driver;
66         this.url = url;
67         load();
68         Pagefactory.initElements(this.driver, this);
69     }
70
71     /**
72      * Sets driver, url and
73      * @param driver
74      * @param url
75      * @param isAbsoluteUrl
76      * TODO: Check the usage of this method and remove it if possible.
77      */
78     protected BasePageObject(WebDriver driver, String url, boolean isAbsoluteUrl) {
79         this.driver = driver;
80         this.url = url;
81         this.isAbsoluteUrl = isAbsoluteUrl;
82         load();
83         Pagefactory.initElements(this.driver, this);
84     }
85
86     /**
87      * Add a constraint that must be met in order for the page to be considered
88      * loaded. This should be used by implementing page objects in a domain
89      * specific manner.
90      *
91      * @param matchCondition
92      */
93 }
```

Project Explorer   Navigator   Type Hierarchy

CustomEventFiri   IScreenshotList   ScreenShot.clas   BasePageObject.   Button.class

com.ebay.maul.driver.web.element

```

1 package com.ebay.maul.driver.web.element;
2
3 import org.openqa.selenium.By;
4
5 import com.ebay.maul.controller.Logging;
6
7 public class Button extends HtmlElement {
8
9     public Button(String label, By by) {
10         super(label, by);
11     }
12
13     public Button(String label, String locator) {
14         super(label, locator);
15     }
16
17     @Override
18     public void click() {
19         captureSnapshot("before clicking");
20         Logging.logWebStep(null, "click on " + toHTML(), false);
21         super.click();
22     }
23
24     public void submit() {
25         captureSnapshot("before submitting form");
26         Logging.logWebStep(null, "submit form by clicking on " + toHTML(),
27             false);
28         findElement();
29         element.submit();
30     }
31 }

```

Markers   Properties   Servers   Data Source Explorer   Snippets   Console   Progress   Search   Results of running class RaiseClaimTest   Package Explorer

No operations to display at this time.

Read-Only   Smart Insert   1:1

Project Explorer   Navigator   Type Hierarchy

IScreenshotList   ScreenShot.clas   BasePageObject.   Button.class   CheckBox.class

com.ebay.maul.driver.web.element

```

1 package com.ebay.maul.driver.web.element;
2
3 import org.openqa.selenium.By;
4
5 import com.ebay.maul.controller.Logging;
6
7 public class CheckBox extends HtmlElement {
8
9     public CheckBox(String label, By by) {
10         super(label, by);
11     }
12
13     public CheckBox(String label, String locator) {
14         super(label, locator);
15     }
16
17     public void check() {
18         Logging.logWebStep(null, "check " + toHTML(), false);
19         if (!isSelected()) {
20             super.click();
21         }
22     }
23
24     @Override
25     public void click() {
26         Logging.logWebStep(null, "click on " + toHTML(), false);
27         super.click();
28     }
29
30     //TODO this code is repeated in other areas--there should be an AbstractClickable and an IClickable where this gets notified
31     //and implemented
32     public boolean isSelected() {
33         findElement();
34         return element.isSelected();
35     }
36
37     public void uncheck() {
38         Logging.logWebStep(null, "uncheck " + toHTML(), false);
39         if (!isSelected()) {
40             super.click();
41         }
42     }
43 }

```

Markers   Properties   Servers   Data Source Explorer   Snippets   Console   Progress   Search   Results of running class RaiseClaim   Package Explorer

No operations to display at this time.

Read-Only   Smart Insert   1:1

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages like com.ebay.mau.driver.web.element, com.ebay.mau.exception, and com.ebay.mau.reporter.
- Code Editor:** Displays the Java code for the **Form** class. The code defines a class **Form** that extends **HTMLElement**. It includes methods for creating forms based on labels and locators, and a **submit()** method that logs a step before submitting and finds the element to submit.
- Outline View:** On the right, it shows the outline of the **Form** class, including its methods: **Form(String label, By by)**, **Form(String label, String locator)**, and **submit()**.
- Toolbars and Menus:** Standard Eclipse toolbars and menus are visible at the top.
- Status Bar:** Shows "Read-Only", "Smart Insert", and "1:1".

```

1 package com.ebay.mau.driver.web.element;
2
3 import org.openqa.selenium.By;
4
5 import com.ebay.mau.controller.Logging;
6
7 public class Form extends HTMLElement {
8
9     public Form(String label, By by) {
10         super(label, by);
11     }
12
13     public Form(String label, String locator) {
14         super(label, locator);
15     }
16
17     public void submit() {
18         Logging.logWebStep(null, "before submitting form");
19         Logging.logWebStep(null, "submit form " + toHTML(), false);
20         findElement();
21         element.submit();
22     }
23 }
24

```

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages like com.ebay.mau.driver.web.element, com.ebay.mau.util, and com.ebay.mau.component.
- Code Editor:** Displays the Java code for the **GenericWebPage** class. This is a generic implementation of **IPage** that interacts with a **WebUI** driver. It handles assertions for element presence and text, and provides methods for navigating and interacting with the page.
- Outline View:** On the right, it shows the outline of the **GenericWebPage** class, including its methods: **convert()**, **setConvert(Boolean convert)**, **assertElementNotPresent(WebElement element)**, **assertElementPresent(WebElement element)**, **assertTextNotPresent(String text)**, **assertTextPresent(String text)**, and **submit()**.
- Toolbars and Menus:** Standard Eclipse toolbars and menus are visible at the top.
- Status Bar:** Shows "Read-Only", "Smart Insert", and "1:1".

```

38 /**
39  * @generated
40  */
41 public class GenericWebPage<T> extends GenericWebPage<T> implements IPage {
42     private static final Logger logger = Logger.getLogger(GenericWebPage.class);
43     private static final long EXPLICIT_WAIT_TIME_OUT = WebUIDriver
44         .getExplicitWait();
45     protected WebDriver webUIDriver = WebUIDriver.getWebUIDriver();
46     protected final WebDriver webDriver = webUIDriver.getWebDriver(true);
47     private String title = null;
48     private String url = null;
49     private String bodyText = null;
50     private String htmlSource = null;
51     private String htmlSavedPath = null;
52     private String imageFilePath = null;
53     private String rpgId = null;
54     private String pageId = null;
55     private Boolean convert = true;
56
57     public Boolean getConvert() {
58         return convert;
59     }
60
61     @SuppressWarnings("unchecked")
62     public T setConvert(Boolean convert) {
63         this.convert = convert;
64         return (T) this;
65     }
66
67     public GenericWebPage<T> {
68     }
69
70     public void assertElementNotPresent(WebElement element) {
71         Logging.logWebStep(null, "assert " + element.toHTML()
72             + " is not present.", false);
73         assertHTML(element.isElementPresent(), element.toString() + " found.");
74     }
75
76     public void assertElementPresent(WebElement element) {
77         Logging.logWebStep(null, "assert " + element.toHTML() + " is present.",
78             false);
79         assertHTML(element.isElementPresent(), element.toString()
80             + " not found.");
81     }
82
83     protected void assertHTML(boolean condition, String message) {
84         if (!condition) {
85             capturePageSnapshot();
86             Assertion.assertITrue(condition, message);
87         }
88     }
89
90     public void assertTextNotPresent(String text) {
91         Logging.logWebStep(null,
92             "assert text '" + text + "' is not present.", false);
93         assertHTML(getBodyText().contains(text), "text=" + text + " found.");
94     }
95
96     public void assertTextPresent(String text) {
97         Logging.logWebStep(null, "assert text '" + text + "' is present.",
98             false);
99         assertHTML(getBodyText().contains(text), "Text=" + text
100            + " not found.");
101    }
102}

```

```

  183     protected void capturePageSnapshot() {
  184         String suiteName = ContextManager.getGlobalContext().getTestNGContext()
  185             .getSuite().getName();
  186         String outputDirectory = ContextManager.getGlobalContext()
  187             .getTestNGContext().getOutputDirectory();
  188
  189         ScreenShotUtil screenshotUtil = new ScreenShotUtil(driver);
  190         screenshotUtil.captureWebPageScreenshot();
  191         this.title = screenshotUtil.getTitle();
  192         this.pageId = screenshotUtil.getPageId();
  193         this.filePath = screenshotUtil.getFilePath();
  194         location = screenshotUtil.getLocation();
  195         htmlSource = screenshotUtil.getHtmlSource();
  196         // Handle StateElementReferenceException for IE and Opera
  197         try {
  198             bodyText = driver.findElement(By.tagName("body")).getText();
  199         } catch (StateElementReferenceException ignore) {
  200             bodyText = htmlSource;
  201         }
  202
  203         if (screenshotUtil.getHtmlSourcePath() != null) {
  204             htmlFullPath = screenshotUtil.getHtmlSourcePath().replace(suiteName,
  205                 outputDirectory);
  206             htmlSavedFilePath = screenshotUtil.getHtmlSourcePath();
  207         }
  208         if (screenshotUtil.getImagePath() != null) {
  209             imagePath = screenshotUtil.getImagePath().replace(suiteName,
  210                 outputDirectory);
  211         }
  212         Logging.logWebOutput(url,
  213             title + " (" + Logging.buildScreenshotLog(screenshotUtil) + ")",
  214             false);
  215     }
  216
  217     private String convert(String url) {
  218         if (ContextManager.getThreadContext() == null) {
  219             String urlToConvert = ContextManager.getThreadContext();
  220             if (curlConvertClass != null) {
  221                 try {
  222                     Class<?> converterClass = Class.forName(curlConvertClass);
  223                     Object converter = converterClass.newInstance();
  224                     Method convert = converterClass.getMethod("convertURL",
  225                         String.class);
  226                     return convert.invoke(converter, url);
  227                 } catch (Exception e) {
  228                     logger.warn("Convert URL failed", e);
  229                 }
  230             }
  231         }
  232         return url;
  233     }
  234
  235     /**
  236      * Ensure that the page is currently loaded.
  237      *
  238      * @return The page.
  239      * @throws Exception
  240      */
  241     @SuppressWarnings("unchecked")
  242     public T get() throws Exception {
  243         if (curl != null) {
  244             load(curlConvert());
  245         }
  246     }
  247
  248     /**
  249      * Ensure that the page is currently loaded.
  250      *
  251      * @return The page.
  252      * @throws Exception
  253      */
  254     @SuppressWarnings("unchecked")
  255     public T get(Boolean convert) throws Exception {
  256         if (curl != null) {
  257             load(convert);
  258         }
  259         waitForPageToLoad();
  260         isLoaded();
  261         return (T) this;
  262     }
  263
  264     /**
  265      * Ensure that the page is currently loaded.
  266      *
  267      * @return The page.
  268      * @throws Exception
  269      */
  270     @SuppressWarnings("unchecked")
  271     public T get(Boolean convert) throws Exception {
  272         if (curl != null) {
  273             load(convert);
  274         }
  275         waitForPageToLoad();
  276         isLoaded();
  277         return (T) this;
  278     }
  279
  280     /**
  281      * Get the body text of the page.
  282      *
  283      * @return The body text.
  284      */
  285     public String getBodyText() {
  286         return bodyText;
  287     }
  288
  289     /**
  290      * Get the element count of the page.
  291      *
  292      * @param element The element to count.
  293      * @return The element count.
  294      */
  295     public int getElementCount(HtmlElement element) {
  296         return element.getElements().size();
  297     }
  298
  299     /**
  300      * Get the saved file path of the page.
  301      *
  302      * @return The saved file path.
  303      */
  304     public String getHtmlSavedFilePath() {
  305         return htmlSavedFilePath;
  306     }
  307
  308     /**
  309      * Get the html source of the page.
  310      *
  311      * @return The html source.
  312      */
  313     public String getHtmlSource() {
  314         return htmlSource;
  315     }
  316
  317     /**
  318      * Get the javascript errors of the page.
  319      *
  320      * @param driver The driver to get the errors from.
  321      * @return The javascript errors.
  322      */
  323     public String getJavaScriptErrors() {
  324         if (WebDriver.getWebDriver().isAddJSSErrorCollectorExtension()) {
  325             List<JavaScriptError> jsErrorList = JavaScriptError
  326                 .readErrors(driver);
  327             if (!jsErrorList.isEmpty()) {
  328                 String jsErrors = "";
  329                 for (int i = 0; i < jsErrorList.size(); i++) {
  330                     jsErrors += jsErrorList.get(i).getLineNumber() + ":" +
  331                         jsErrorList.get(i).getColumnNumber() + ":" +
  332                         jsErrorList.get(i).getErrorMessage() + ":" +
  333                         jsErrorList.get(i).getSourceName() + "\n";
  334                 }
  335             }
  336             return jsErrors;
  337         }
  338         return null;
  339     }
  340
  341     /**
  342      * Get the keyword value of the page.
  343      *
  344      * @param key The keyword to get the value for.
  345      * @return The keyword value.
  346      */
  347     public String getKeywordValue(String key) {
  348         String site = null;
  349         if (ContextManager.getThreadContext().getGnsSite() != null) {
  350             site = ContextManager.getThreadContext().getGnsSite();
  351         } else {
  352             site = ContextManager.getThreadContext().getSite();
  353         }
  354     }

```

```

239     site = contextManager.getThreadContext().getSite();
240 }
241 
242     return Keyword.get(this.getClass(), key, site);
243 }
244 
245     public String getLocation() {
246         return location;
247     }
248 
249     public String getPageId() {
250         return pageId;
251     }
252 
253     public String getLogId() {
254         return rlogId;
255     }
256 
257     public String getTitle() {
258         return title;
259     }
260 
261     public String getUrl() {
262         return url;
263     }
264 
265     /**
266      * Add this method in load to improve the stability of page loading
267      * especially for opera
268     */
269     private void waitForPageToLoad() {
270         try {
271             driver.switchTo().defaultContent();
272         } catch (UnhandledAlertException e) {
273             System.out.println("Handling alert in waitForPageToLoad");
274             ThreadHelper.waitForSeconds(2);
275         } catch (WebDriverException e) {
276             e.printStackTrace();
277         }
278     }
279 
280     /**
281      * Determine whether or not the page is loaded.
282      * @throws PageNotCurrentException
283     */
284     protected void isPageLoaded() throws PageNotCurrentException {
285         if (!BaseWebUtil.isPageReady(getType())) {
286             ContextManager.getThreadContext().getWebRunBrowser());
287             if (!isPageLoaded()) {
288                 // populateAndCapturePageSnapshot();
289                 new ScreenShotUtil(driver).capturePageSnapshotOnException();
290             } catch (Exception e) {
291                 // Ignore all exceptions
292                 e.printStackTrace();
293             }
294             throw new PageNotCurrentException(getClass().getCanonicalName()
295                     + " is not the current page.");
296         }
297         populateAndCapturePageSnapshot();
298         AbstractPageListener.notifyPageLoad(this);
299     }
300 }
301 
302 /**
303  * Determine whether the page is loaded or not. It return true by default.
304  * Override is needed in the real page if we want to add any condition for
305  * page load. The expected sample usage is:
306  *
307  * -pre class="PageNotCurrentException"
308  * <return>${!#isElementPresent}
309  * </pre>
310  */
311 
312     protected boolean isPageLoaded() {
313         return true;
314     }
315 
316     /**
317      * Check HTTP Status Code Errors. This is going be tricky. Selenium does not
318      * provide us with HTTP Response codes. So have to rely on the DOM content
319      * to check for errors. For example: QA PM Page If page error, take error
320      * snapshot and throw exception
321      * @throws PageNotCurrentException
322     */
323     public final void assertNoPageErrors() {
324         for (HTTPStatusCode statusCode : HTTPStatusCode.values()) {
325             if (gettbodyText() != null
326                 && gettbodyText().toLowerCase().contains(
327                     statusCode.getTitle().toLowerCase())) {
328                 throw new RuntimeException(statusCode.toString() + " found.");
329             }
330         }
331     }
332 
333     /**
334      * Load page.
335      */
336     protected void load(boolean convert) {
337         if (convert) {
338             url = convert(url);
339             Logging.logWebStep(url, "Open url => href=" + url + ">" + url
340                               + "</a>", false);
341         }
342         try {
343             driver.get(url);
344         } catch (org.openqa.selenium.TimeoutException e) {
345             Logging.log("Got time out when loading " + url + ", ignored");
346         } catch (org.openqa.selenium.UnsupportedCommandException e) {
347             Logging.log("Got UnsupportedCommandException when loading " + url
348                         + " retry");
349             driver.get(url);
350         } catch (org.openqa.selenium.UnhandledAlertException ex) {
351             Logging.log("got UnhandledAlertException, retry");
352             driver.navigate().to(url);
353         }
354         url = null;
355     }
356 
357     private void populateAndCapturePageSnapshot() {
358         try {
359             title = driver.getTitle();
360         } catch (org.openqa.selenium.UnhandledAlertException ex) {

```

```

  361     // ignore silent exception
  362     title = driver.getTitle();
  363   }
  364   capturePageSnapshot();
  365   assertEqualsNoPageErrors();
  366 }
  367
  368 public void resizeWindow(int width, int height) {
  369   new BaseWebUtil(driver).resizeWindow(width, height);
  370 }
  371
  372 public void maximizeWindow() {
  373   new BaseWebUtil(driver).maximizeWindow();
  374 }
  375
  376 @SuppressWarnings("unchecked")
  377 public T refresh() throws Exception {
  378   Logging.logWebStep(null, "refresh page", false);
  379   try {
  380     driver.navigate().refresh();
  381   } catch (org.openqa.selenium.TimeoutException ex) {
  382     Logging.log("got time out exception, ignore");
  383   }
  384   isloaded();
  385   return (T) this;
  386 }
  387
  388 public void selectFrame(By by) {
  389   Logging.logWebStep(null, "select frame, locator:" + by.toString()
  390   + ")", false);
  391   driver.switchTo().frame(driver.findElement(by));
  392 }
  393
  394 public void setBodyText(String bodyText) {
  395   this.bodyText = bodyText;
  396 }
  397
  398 public void setHtmlSavedToPath(String htmlSavedToPath) {
  399   this.htmlSavedToPath = htmlSavedToPath;
  400 }
  401
  402 public void setHtmlSource(String htmlSource) {
  403   this.htmlSource = htmlSource;
  404 }
  405
  406 public void setLocation(String location) {
  407   this.location = location;
  408 }
  409
  410 public void setPageId(String pageId) {
  411   this.pageId = pageId;
  412 }
  413
  414 public void setRigId(String rigId) {
  415   this.rigId = rigId;
  416 }
  417
  418 public void setTitle(String title) {
  419   this.title = title;
  420 }
  421
  422 @SuppressWarnings("unchecked")
  423 public T setUrl(String url) {
  424   this.url = url;
  425   return (T) this;
  426 }

  427
  428 public void waitForElementToBeClickable(WebElement element) {
  429   Assert.assertNotNull(element, "Element can't be null");
  430   Logging.logWebStep(null, "wait for " + element.toString()
  431   + " to be clickable", false);
  432   WebDriveWait wait = new WebDriveWait(driver, EXPLICIT_WAIT_TIME_OUT);
  433   wait.until(ExpectedConditions.elementToBeClickable(element.getBy())));
  434 }

  435 public void waitForElementToBeSelected(WebElement element) {
  436   Assert.assertNotNull(element, "Element can't be null");
  437   Logging.logWebStep(null, "wait for " + element.toString()
  438   + " to be selected", false);
  439   WebDriveWait wait = new WebDriveWait(driver, EXPLICIT_WAIT_TIME_OUT);
  440   wait.until(ExpectedConditions.elementToBeSelected(element.getBy())));
  441 }

  442 public void waitForElementToBeVisible(WebElement element) {
  443   Assert.assertNotNull(element, "Element can't be null");
  444   Logging.logWebStep(null, "wait for " + element.toString()
  445   + " to be visible", false);
  446   WebDriveWait wait = new WebDriveWait(driver, EXPLICIT_WAIT_TIME_OUT);
  447   wait.until(ExpectedConditions.visibilityOfElementLocated(
  448   element.getBy())));
  449 }

  450 public void waitForElementToDisappear(WebElement element) {
  451   Assert.assertNotNull(element, "Element can't be null");
  452   Logging.logWebStep(null, "wait for " + element.toString()
  453   + " to disappear", false);
  454   WebDriveWait wait = new WebDriveWait(driver, EXPLICIT_WAIT_TIME_OUT);
  455   wait.until(ExpectedConditions.invisibilityOfElementLocated(
  456   element.getBy())));
  457 }

  458 public void waitForElementToPresent(WebElement element) {
  459   Assert.assertNotNull(element, "Element can't be null");
  460   Logging.logWebStep(null, "wait for " + element.toString()
  461   + " to present", false);
  462   WebDriveWait wait = new WebDriveWait(driver, EXPLICIT_WAIT_TIME_OUT);
  463   wait.until(ExpectedConditions.textToBePresentInElementLocated(
  464   element.getBy(), text)));
  465 }

  466 protected void waitForSeconds(int seconds) {
  467   ThreadHelper.waitForSeconds(seconds);
  468 }

  469 public void waitForTextToPresent(WebElement element, String text) {
  470   Assert.assertNotNull(text, "Text can't be null");
  471   Logging.logWebStep(null, "wait for text '" + text + "' to present.");
  472   WebDriveWait wait = new WebDriveWait(driver, EXPLICIT_WAIT_TIME_OUT);
  473   wait.until(ExpectedConditions.textToBePresentInElementLocated(
  474   element.getBy(), text));
  475 }

  476 public String getHTMLFilePath() {
  477   return htmlFilePath;
  478 }
  479
  480 public String getImageFilePath() {
  481   return imagePath;
  482 }
  483 }
  484
  485 }

  486 public String getScreenshotPath() {
  487   return screenshotPath;
  488 }
  489
  490 }

  491

```

```

43 public class HtmlElement {
44     private static enum LocatorType {
45         ID, NAME, CLASS_NAME, LINK_TEXT, PARTIAL_LINK_TEXT, CSS_SELECTOR, TAG_NAME, XPATH,
46     }
47
48     private static final int EXPLICIT_WAIT_TIMEOUT = WebUI.WebDriver
49             .getWebUIWebDriver().getExplicitWait();
50     protected static final Logger logger = Logging.getLogger(HtmlElement.class);
51     protected WebDriver driver = WebUI.WebDriver.getWebUIWebDriver();
52     protected IWebElement element = WebUI.WebDriver.getWebUIWebDriver();
53     protected BrowserType browser = WebUI.WebDriver.getWebUIWebDriver().getBrowser();
54     protected WebElementBy by;
55     protected WebElement element = null;
56     private String label = null;
57     private String locator = null;
58     private By by1 = null;
59
60     /**
61      * Find element using BY locator. Make sure to initialize the driver before
62      * calling findElement()
63      *
64      * @param label
65      * @param by - element name for logging
66      */
67     public HtmlElement(String label, By by) {
68         this.label = label;
69         this.by = by;
70     }
71
72     /**
73      * This constructor will attempt to locate the element using the locator
74      * string. It will accept "xpath[locator]", or anything starting with / or C
75      * os-is, and us it properly. If anything else is passed, it will attempt to
76      * locate the element using the id or name as given
77      *
78      * @param label
79      * @param locator - xpath locator
80      */
81     public HtmlElement(String label, String locator) {
82         this.label = label;
83         // For backward compatibility
84         if (locator.startsWith("xpath")) {
85             locator = locator.substring(5);
86         } else if (locator.startsWith("//") && !locator.startsWith("//")) {
87             locator = "//[" + locator + " or @name=" + locator + "]";
88         }
89         this.locator = locator;
90         this.by = By.xpath(locator);
91     }
92
93     /**
94      * This constructor locates the element using locator and locator type
95      *
96      * @param label
97      * @param locator
98     */
99
100    public HtmlElement(String label, String locator, LocatorType locatorType) {
101        this.label = label;
102        this.locator = locator;
103        this.by = getLocatorBy(locator, locatorType);
104    }
105
106    /**
107     * Capture a snapshot of the current browser window
108     */
109    public void captureSnapshot() {
110        captureSnapshot(ClassContextHelper.getCallerMethod() + " on ");
111    }
112
113    /**
114     * Capture a snapshot of the current browser window, and prefix the file
115     * name with the assigned string
116     */
117    protected void captureSnapshot(String messagePrefix) {
118        ScreenShotUtil.captureSnapshot(messagePrefix);
119    }
120
121    /**
122     * Click the element
123     */
124    public void click() {
125        click();
126    }
127
128    /**
129     * Click the element in native way by Actions
130     */
131    public void clickAt() {
132        clickAt("1,1");
133    }
134
135    /**
136     * Handle Confirm Navigation pop up for Chrome and IE
137     */
138    protected void handleLeaveAlert() {
139        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
140            ThreadHelper.waitForSeconds(2);
141        }
142
143        // If is "too Fast"
144        if (browser == BrowserType.InternetExplore) {
145            ThreadHelper.waitForSeconds(1);
146        }
147        handleLeaveAlert();
148    }
149
150    /**
151     * Click element in native way by Actions
152     */
153    public void clickAt() {
154        clickAt("1,1");
155    }
156
157    /**
158     * Handle Confirmation pop up for Chrome and IE
159     */
160    protected void handleLeaveAlert() {
161        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
162            ThreadHelper.waitForSeconds(2);
163        }
164
165        // If is "too Fast"
166        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
167            ThreadHelper.waitForSeconds(1);
168        }
169        handleLeaveAlert();
170    }
171
172    /**
173     * Handle Confirmation pop up for Chrome and IE
174     */
175    protected void handleLeaveAlert() {
176        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
177            ThreadHelper.waitForSeconds(2);
178        }
179    }
180
181    /**
182     * Click element in native way by Actions
183     */
184    public void clickAt() {
185        clickAt("1,1");
186    }
187
188    /**
189     * Handle Confirmation pop up for Chrome and IE
190     */
191    protected void handleLeaveAlert() {
192        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
193            ThreadHelper.waitForSeconds(2);
194        }
195        handleLeaveAlert();
196    }
197
198    /**
199     * Click element in native way by Actions
200     */
201    public void clickAt() {
202        clickAt("1,1");
203    }
204
205    /**
206     * Handle Confirmation pop up for Chrome and IE
207     */
208    protected void handleLeaveAlert() {
209        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
210            ThreadHelper.waitForSeconds(2);
211        }
212        handleLeaveAlert();
213    }
214
215    /**
216     * Click element in native way by Actions
217     */
218    public void clickAt() {
219        clickAt("1,1");
220    }
221
222    /**
223     * Handle Confirmation pop up for Chrome and IE
224     */
225    protected void handleLeaveAlert() {
226        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
227            ThreadHelper.waitForSeconds(2);
228        }
229        handleLeaveAlert();
230    }
231
232    /**
233     * Click element in native way by Actions
234     */
235    public void clickAt() {
236        clickAt("1,1");
237    }
238
239    /**
240     * Handle Confirmation pop up for Chrome and IE
241     */
242    protected void handleLeaveAlert() {
243        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
244            ThreadHelper.waitForSeconds(2);
245        }
246        handleLeaveAlert();
247    }
248
249    /**
250     * Click element in native way by Actions
251     */
252    public void clickAt() {
253        clickAt("1,1");
254    }
255
256    /**
257     * Handle Confirmation pop up for Chrome and IE
258     */
259    protected void handleLeaveAlert() {
260        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
261            ThreadHelper.waitForSeconds(2);
262        }
263        handleLeaveAlert();
264    }
265
266    /**
267     * Click element in native way by Actions
268     */
269    public void clickAt() {
270        clickAt("1,1");
271    }
272
273    /**
274     * Handle Confirmation pop up for Chrome and IE
275     */
276    protected void handleLeaveAlert() {
277        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
278            ThreadHelper.waitForSeconds(2);
279        }
280        handleLeaveAlert();
281    }
282
283    /**
284     * Click element in native way by Actions
285     */
286    public void clickAt() {
287        clickAt("1,1");
288    }
289
290    /**
291     * Handle Confirmation pop up for Chrome and IE
292     */
293    protected void handleLeaveAlert() {
294        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
295            ThreadHelper.waitForSeconds(2);
296        }
297        handleLeaveAlert();
298    }
299
300    /**
301     * Click element in native way by Actions
302     */
303    public void clickAt() {
304        clickAt("1,1");
305    }
306
307    /**
308     * Handle Confirmation pop up for Chrome and IE
309     */
310    protected void handleLeaveAlert() {
311        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
312            ThreadHelper.waitForSeconds(2);
313        }
314        handleLeaveAlert();
315    }
316
317    /**
318     * Click element in native way by Actions
319     */
320    public void clickAt() {
321        clickAt("1,1");
322    }
323
324    /**
325     * Handle Confirmation pop up for Chrome and IE
326     */
327    protected void handleLeaveAlert() {
328        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
329            ThreadHelper.waitForSeconds(2);
330        }
331        handleLeaveAlert();
332    }
333
334    /**
335     * Click element in native way by Actions
336     */
337    public void clickAt() {
338        clickAt("1,1");
339    }
340
341    /**
342     * Handle Confirmation pop up for Chrome and IE
343     */
344    protected void handleLeaveAlert() {
345        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
346            ThreadHelper.waitForSeconds(2);
347        }
348        handleLeaveAlert();
349    }
350
351    /**
352     * Click element in native way by Actions
353     */
354    public void clickAt() {
355        clickAt("1,1");
356    }
357
358    /**
359     * Handle Confirmation pop up for Chrome and IE
360     */
361    protected void handleLeaveAlert() {
362        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
363            ThreadHelper.waitForSeconds(2);
364        }
365        handleLeaveAlert();
366    }
367
368    /**
369     * Click element in native way by Actions
370     */
371    public void clickAt() {
372        clickAt("1,1");
373    }
374
375    /**
376     * Handle Confirmation pop up for Chrome and IE
377     */
378    protected void handleLeaveAlert() {
379        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
380            ThreadHelper.waitForSeconds(2);
381        }
382        handleLeaveAlert();
383    }
384
385    /**
386     * Click element in native way by Actions
387     */
388    public void clickAt() {
389        clickAt("1,1");
390    }
391
392    /**
393     * Handle Confirmation pop up for Chrome and IE
394     */
395    protected void handleLeaveAlert() {
396        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
397            ThreadHelper.waitForSeconds(2);
398        }
399        handleLeaveAlert();
400    }
401
402    /**
403     * Click element in native way by Actions
404     */
405    public void clickAt() {
406        clickAt("1,1");
407    }
408
409    /**
410     * Handle Confirmation pop up for Chrome and IE
411     */
412    protected void handleLeaveAlert() {
413        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
414            ThreadHelper.waitForSeconds(2);
415        }
416        handleLeaveAlert();
417    }
418
419    /**
420     * Click element in native way by Actions
421     */
422    public void clickAt() {
423        clickAt("1,1");
424    }
425
426    /**
427     * Handle Confirmation pop up for Chrome and IE
428     */
429    protected void handleLeaveAlert() {
430        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
431            ThreadHelper.waitForSeconds(2);
432        }
433        handleLeaveAlert();
434    }
435
436    /**
437     * Click element in native way by Actions
438     */
439    public void clickAt() {
440        clickAt("1,1");
441    }
442
443    /**
444     * Handle Confirmation pop up for Chrome and IE
445     */
446    protected void handleLeaveAlert() {
447        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
448            ThreadHelper.waitForSeconds(2);
449        }
450        handleLeaveAlert();
451    }
452
453    /**
454     * Click element in native way by Actions
455     */
456    public void clickAt() {
457        clickAt("1,1");
458    }
459
460    /**
461     * Handle Confirmation pop up for Chrome and IE
462     */
463    protected void handleLeaveAlert() {
464        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
465            ThreadHelper.waitForSeconds(2);
466        }
467        handleLeaveAlert();
468    }
469
470    /**
471     * Click element in native way by Actions
472     */
473    public void clickAt() {
474        clickAt("1,1");
475    }
476
477    /**
478     * Handle Confirmation pop up for Chrome and IE
479     */
480    protected void handleLeaveAlert() {
481        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
482            ThreadHelper.waitForSeconds(2);
483        }
484        handleLeaveAlert();
485    }
486
487    /**
488     * Click element in native way by Actions
489     */
490    public void clickAt() {
491        clickAt("1,1");
492    }
493
494    /**
495     * Handle Confirmation pop up for Chrome and IE
496     */
497    protected void handleLeaveAlert() {
498        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
499            ThreadHelper.waitForSeconds(2);
500        }
501        handleLeaveAlert();
502    }
503
504    /**
505     * Click element in native way by Actions
506     */
507    public void clickAt() {
508        clickAt("1,1");
509    }
510
511    /**
512     * Handle Confirmation pop up for Chrome and IE
513     */
514    protected void handleLeaveAlert() {
515        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
516            ThreadHelper.waitForSeconds(2);
517        }
518        handleLeaveAlert();
519    }
520
521    /**
522     * Click element in native way by Actions
523     */
524    public void clickAt() {
525        clickAt("1,1");
526    }
527
528    /**
529     * Handle Confirmation pop up for Chrome and IE
530     */
531    protected void handleLeaveAlert() {
532        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
533            ThreadHelper.waitForSeconds(2);
534        }
535        handleLeaveAlert();
536    }
537
538    /**
539     * Click element in native way by Actions
540     */
541    public void clickAt() {
542        clickAt("1,1");
543    }
544
545    /**
546     * Handle Confirmation pop up for Chrome and IE
547     */
548    protected void handleLeaveAlert() {
549        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
550            ThreadHelper.waitForSeconds(2);
551        }
552        handleLeaveAlert();
553    }
554
555    /**
556     * Click element in native way by Actions
557     */
558    public void clickAt() {
559        clickAt("1,1");
560    }
561
562    /**
563     * Handle Confirmation pop up for Chrome and IE
564     */
565    protected void handleLeaveAlert() {
566        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
567            ThreadHelper.waitForSeconds(2);
568        }
569        handleLeaveAlert();
570    }
571
572    /**
573     * Click element in native way by Actions
574     */
575    public void clickAt() {
576        clickAt("1,1");
577    }
578
579    /**
580     * Handle Confirmation pop up for Chrome and IE
581     */
582    protected void handleLeaveAlert() {
583        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
584            ThreadHelper.waitForSeconds(2);
585        }
586        handleLeaveAlert();
587    }
588
589    /**
590     * Click element in native way by Actions
591     */
592    public void clickAt() {
593        clickAt("1,1");
594    }
595
596    /**
597     * Handle Confirmation pop up for Chrome and IE
598     */
599    protected void handleLeaveAlert() {
600        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
601            ThreadHelper.waitForSeconds(2);
602        }
603        handleLeaveAlert();
604    }
605
606    /**
607     * Click element in native way by Actions
608     */
609    public void clickAt() {
610        clickAt("1,1");
611    }
612
613    /**
614     * Handle Confirmation pop up for Chrome and IE
615     */
616    protected void handleLeaveAlert() {
617        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
618            ThreadHelper.waitForSeconds(2);
619        }
620        handleLeaveAlert();
621    }
622
623    /**
624     * Click element in native way by Actions
625     */
626    public void clickAt() {
627        clickAt("1,1");
628    }
629
630    /**
631     * Handle Confirmation pop up for Chrome and IE
632     */
633    protected void handleLeaveAlert() {
634        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
635            ThreadHelper.waitForSeconds(2);
636        }
637        handleLeaveAlert();
638    }
639
640    /**
641     * Click element in native way by Actions
642     */
643    public void clickAt() {
644        clickAt("1,1");
645    }
646
647    /**
648     * Handle Confirmation pop up for Chrome and IE
649     */
650    protected void handleLeaveAlert() {
651        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
652            ThreadHelper.waitForSeconds(2);
653        }
654        handleLeaveAlert();
655    }
656
657    /**
658     * Click element in native way by Actions
659     */
660    public void clickAt() {
661        clickAt("1,1");
662    }
663
664    /**
665     * Handle Confirmation pop up for Chrome and IE
666     */
667    protected void handleLeaveAlert() {
668        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
669            ThreadHelper.waitForSeconds(2);
670        }
671        handleLeaveAlert();
672    }
673
674    /**
675     * Click element in native way by Actions
676     */
677    public void clickAt() {
678        clickAt("1,1");
679    }
680
681    /**
682     * Handle Confirmation pop up for Chrome and IE
683     */
684    protected void handleLeaveAlert() {
685        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
686            ThreadHelper.waitForSeconds(2);
687        }
688        handleLeaveAlert();
689    }
690
691    /**
692     * Click element in native way by Actions
693     */
694    public void clickAt() {
695        clickAt("1,1");
696    }
697
698    /**
699     * Handle Confirmation pop up for Chrome and IE
700     */
701    protected void handleLeaveAlert() {
702        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
703            ThreadHelper.waitForSeconds(2);
704        }
705        handleLeaveAlert();
706    }
707
708    /**
709     * Click element in native way by Actions
710     */
711    public void clickAt() {
712        clickAt("1,1");
713    }
714
715    /**
716     * Handle Confirmation pop up for Chrome and IE
717     */
718    protected void handleLeaveAlert() {
719        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
720            ThreadHelper.waitForSeconds(2);
721        }
722        handleLeaveAlert();
723    }
724
725    /**
726     * Click element in native way by Actions
727     */
728    public void clickAt() {
729        clickAt("1,1");
730    }
731
732    /**
733     * Handle Confirmation pop up for Chrome and IE
734     */
735    protected void handleLeaveAlert() {
736        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
737            ThreadHelper.waitForSeconds(2);
738        }
739        handleLeaveAlert();
740    }
741
742    /**
743     * Click element in native way by Actions
744     */
745    public void clickAt() {
746        clickAt("1,1");
747    }
748
749    /**
750     * Handle Confirmation pop up for Chrome and IE
751     */
752    protected void handleLeaveAlert() {
753        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
754            ThreadHelper.waitForSeconds(2);
755        }
756        handleLeaveAlert();
757    }
758
759    /**
760     * Click element in native way by Actions
761     */
762    public void clickAt() {
763        clickAt("1,1");
764    }
765
766    /**
767     * Handle Confirmation pop up for Chrome and IE
768     */
769    protected void handleLeaveAlert() {
770        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
771            ThreadHelper.waitForSeconds(2);
772        }
773        handleLeaveAlert();
774    }
775
776    /**
777     * Click element in native way by Actions
778     */
779    public void clickAt() {
780        clickAt("1,1");
781    }
782
783    /**
784     * Handle Confirmation pop up for Chrome and IE
785     */
786    protected void handleLeaveAlert() {
787        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
788            ThreadHelper.waitForSeconds(2);
789        }
790        handleLeaveAlert();
791    }
792
793    /**
794     * Click element in native way by Actions
795     */
796    public void clickAt() {
797        clickAt("1,1");
798    }
799
800    /**
801     * Handle Confirmation pop up for Chrome and IE
802     */
803    protected void handleLeaveAlert() {
804        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
805            ThreadHelper.waitForSeconds(2);
806        }
807        handleLeaveAlert();
808    }
809
810    /**
811     * Click element in native way by Actions
812     */
813    public void clickAt() {
814        clickAt("1,1");
815    }
816
817    /**
818     * Handle Confirmation pop up for Chrome and IE
819     */
820    protected void handleLeaveAlert() {
821        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
822            ThreadHelper.waitForSeconds(2);
823        }
824        handleLeaveAlert();
825    }
826
827    /**
828     * Click element in native way by Actions
829     */
830    public void clickAt() {
831        clickAt("1,1");
832    }
833
834    /**
835     * Handle Confirmation pop up for Chrome and IE
836     */
837    protected void handleLeaveAlert() {
838        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
839            ThreadHelper.waitForSeconds(2);
840        }
841        handleLeaveAlert();
842    }
843
844    /**
845     * Click element in native way by Actions
846     */
847    public void clickAt() {
848        clickAt("1,1");
849    }
850
851    /**
852     * Handle Confirmation pop up for Chrome and IE
853     */
854    protected void handleLeaveAlert() {
855        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
856            ThreadHelper.waitForSeconds(2);
857        }
858        handleLeaveAlert();
859    }
860
861    /**
862     * Click element in native way by Actions
863     */
864    public void clickAt() {
865        clickAt("1,1");
866    }
867
868    /**
869     * Handle Confirmation pop up for Chrome and IE
870     */
871    protected void handleLeaveAlert() {
872        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
873            ThreadHelper.waitForSeconds(2);
874        }
875        handleLeaveAlert();
876    }
877
878    /**
879     * Click element in native way by Actions
880     */
881    public void clickAt() {
882        clickAt("1,1");
883    }
884
885    /**
886     * Handle Confirmation pop up for Chrome and IE
887     */
888    protected void handleLeaveAlert() {
889        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
890            ThreadHelper.waitForSeconds(2);
891        }
892        handleLeaveAlert();
893    }
894
895    /**
896     * Click element in native way by Actions
897     */
898    public void clickAt() {
899        clickAt("1,1");
900    }
901
902    /**
903     * Handle Confirmation pop up for Chrome and IE
904     */
905    protected void handleLeaveAlert() {
906        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
907            ThreadHelper.waitForSeconds(2);
908        }
909        handleLeaveAlert();
910    }
911
912    /**
913     * Click element in native way by Actions
914     */
915    public void clickAt() {
916        clickAt("1,1");
917    }
918
919    /**
920     * Handle Confirmation pop up for Chrome and IE
921     */
922    protected void handleLeaveAlert() {
923        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
924            ThreadHelper.waitForSeconds(2);
925        }
926        handleLeaveAlert();
927    }
928
929    /**
930     * Click element in native way by Actions
931     */
932    public void clickAt() {
933        clickAt("1,1");
934    }
935
936    /**
937     * Handle Confirmation pop up for Chrome and IE
938     */
939    protected void handleLeaveAlert() {
940        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
941            ThreadHelper.waitForSeconds(2);
942        }
943        handleLeaveAlert();
944    }
945
946    /**
947     * Click element in native way by Actions
948     */
949    public void clickAt() {
950        clickAt("1,1");
951    }
952
953    /**
954     * Handle Confirmation pop up for Chrome and IE
955     */
956    protected void handleLeaveAlert() {
957        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
958            ThreadHelper.waitForSeconds(2);
959        }
960        handleLeaveAlert();
961    }
962
963    /**
964     * Click element in native way by Actions
965     */
966    public void clickAt() {
967        clickAt("1,1");
968    }
969
970    /**
971     * Handle Confirmation pop up for Chrome and IE
972     */
973    protected void handleLeaveAlert() {
974        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
975            ThreadHelper.waitForSeconds(2);
976        }
977        handleLeaveAlert();
978    }
979
980    /**
981     * Click element in native way by Actions
982     */
983    public void clickAt() {
984        clickAt("1,1");
985    }
986
987    /**
988     * Handle Confirmation pop up for Chrome and IE
989     */
990    protected void handleLeaveAlert() {
991        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
992            ThreadHelper.waitForSeconds(2);
993        }
994        handleLeaveAlert();
995    }
996
997    /**
998     * Click element in native way by Actions
999     */
1000    public void clickAt() {
1001        clickAt("1,1");
1002    }
1003
1004    /**
1005     * Handle Confirmation pop up for Chrome and IE
1006     */
1007    protected void handleLeaveAlert() {
1008        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1009            ThreadHelper.waitForSeconds(2);
1010        }
1011        handleLeaveAlert();
1012    }
1013
1014    /**
1015     * Click element in native way by Actions
1016     */
1017    public void clickAt() {
1018        clickAt("1,1");
1019    }
1020
1021    /**
1022     * Handle Confirmation pop up for Chrome and IE
1023     */
1024    protected void handleLeaveAlert() {
1025        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1026            ThreadHelper.waitForSeconds(2);
1027        }
1028        handleLeaveAlert();
1029    }
1030
1031    /**
1032     * Click element in native way by Actions
1033     */
1034    public void clickAt() {
1035        clickAt("1,1");
1036    }
1037
1038    /**
1039     * Handle Confirmation pop up for Chrome and IE
1040     */
1041    protected void handleLeaveAlert() {
1042        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1043            ThreadHelper.waitForSeconds(2);
1044        }
1045        handleLeaveAlert();
1046    }
1047
1048    /**
1049     * Click element in native way by Actions
1050     */
1051    public void clickAt() {
1052        clickAt("1,1");
1053    }
1054
1055    /**
1056     * Handle Confirmation pop up for Chrome and IE
1057     */
1058    protected void handleLeaveAlert() {
1059        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1060            ThreadHelper.waitForSeconds(2);
1061        }
1062        handleLeaveAlert();
1063    }
1064
1065    /**
1066     * Click element in native way by Actions
1067     */
1068    public void clickAt() {
1069        clickAt("1,1");
1070    }
1071
1072    /**
1073     * Handle Confirmation pop up for Chrome and IE
1074     */
1075    protected void handleLeaveAlert() {
1076        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1077            ThreadHelper.waitForSeconds(2);
1078        }
1079        handleLeaveAlert();
1080    }
1081
1082    /**
1083     * Click element in native way by Actions
1084     */
1085    public void clickAt() {
1086        clickAt("1,1");
1087    }
1088
1089    /**
1090     * Handle Confirmation pop up for Chrome and IE
1091     */
1092    protected void handleLeaveAlert() {
1093        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1094            ThreadHelper.waitForSeconds(2);
1095        }
1096        handleLeaveAlert();
1097    }
1098
1099    /**
1100     * Click element in native way by Actions
1101     */
1102    public void clickAt() {
1103        clickAt("1,1");
1104    }
1105
1106    /**
1107     * Handle Confirmation pop up for Chrome and IE
1108     */
1109    protected void handleLeaveAlert() {
1110        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1111            ThreadHelper.waitForSeconds(2);
1112        }
1113        handleLeaveAlert();
1114    }
1115
1116    /**
1117     * Click element in native way by Actions
1118     */
1119    public void clickAt() {
1120        clickAt("1,1");
1121    }
1122
1123    /**
1124     * Handle Confirmation pop up for Chrome and IE
1125     */
1126    protected void handleLeaveAlert() {
1127        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1128            ThreadHelper.waitForSeconds(2);
1129        }
1130        handleLeaveAlert();
1131    }
1132
1133    /**
1134     * Click element in native way by Actions
1135     */
1136    public void clickAt() {
1137        clickAt("1,1");
1138    }
1139
1140    /**
1141     * Handle Confirmation pop up for Chrome and IE
1142     */
1143    protected void handleLeaveAlert() {
1144        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1145            ThreadHelper.waitForSeconds(2);
1146        }
1147        handleLeaveAlert();
1148    }
1149
1150    /**
1151     * Click element in native way by Actions
1152     */
1153    public void clickAt() {
1154        clickAt("1,1");
1155    }
1156
1157    /**
1158     * Handle Confirmation pop up for Chrome and IE
1159     */
1160    protected void handleLeaveAlert() {
1161        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1162            ThreadHelper.waitForSeconds(2);
1163        }
1164        handleLeaveAlert();
1165    }
1166
1167    /**
1168     * Click element in native way by Actions
1169     */
1170    public void clickAt() {
1171        clickAt("1,1");
1172    }
1173
1174    /**
1175     * Handle Confirmation pop up for Chrome and IE
1176     */
1177    protected void handleLeaveAlert() {
1178        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1179            ThreadHelper.waitForSeconds(2);
1180        }
1181        handleLeaveAlert();
1182    }
1183
1184    /**
1185     * Click element in native way by Actions
1186     */
1187    public void clickAt() {
1188        clickAt("1,1");
1189    }
1190
1191    /**
1192     * Handle Confirmation pop up for Chrome and IE
1193     */
1194    protected void handleLeaveAlert() {
1195        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1196            ThreadHelper.waitForSeconds(2);
1197        }
1198        handleLeaveAlert();
1199    }
1200
1201    /**
1202     * Click element in native way by Actions
1203     */
1204    public void clickAt() {
1205        clickAt("1,1");
1206    }
1207
1208    /**
1209     * Handle Confirmation pop up for Chrome and IE
1210     */
1211    protected void handleLeaveAlert() {
1212        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1213            ThreadHelper.waitForSeconds(2);
1214        }
1215        handleLeaveAlert();
1216    }
1217
1218    /**
1219     * Click element in native way by Actions
1220     */
1221    public void clickAt() {
1222        clickAt("1,1");
1223    }
1224
1225    /**
1226     * Handle Confirmation pop up for Chrome and IE
1227     */
1228    protected void handleLeaveAlert() {
1229        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1230            ThreadHelper.waitForSeconds(2);
1231        }
1232        handleLeaveAlert();
1233    }
1234
1235    /**
1236     * Click element in native way by Actions
1237     */
1238    public void clickAt() {
1239        clickAt("1,1");
1240    }
1241
1242    /**
1243     * Handle Confirmation pop up for Chrome and IE
1244     */
1245    protected void handleLeaveAlert() {
1246        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1247            ThreadHelper.waitForSeconds(2);
1248        }
1249        handleLeaveAlert();
1250    }
1251
1252    /**
1253     * Click element in native way by Actions
1254     */
1255    public void clickAt() {
1256        clickAt("1,1");
1257    }
1258
1259    /**
1260     * Handle Confirmation pop up for Chrome and IE
1261     */
1262    protected void handleLeaveAlert() {
1263        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1264            ThreadHelper.waitForSeconds(2);
1265        }
1266        handleLeaveAlert();
1267    }
1268
1269    /**
1270     * Click element in native way by Actions
1271     */
1272    public void clickAt() {
1273        clickAt("1,1");
1274    }
1275
1276    /**
1277     * Handle Confirmation pop up for Chrome and IE
1278     */
1279    protected void handleLeaveAlert() {
1280        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1281            ThreadHelper.waitForSeconds(2);
1282        }
1283        handleLeaveAlert();
1284    }
1285
1286    /**
1287     * Click element in native way by Actions
1288     */
1289    public void clickAt() {
1290        clickAt("1,1");
1291    }
1292
1293    /**
1294     * Handle Confirmation pop up for Chrome and IE
1295     */
1296    protected void handleLeaveAlert() {
1297        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1298            ThreadHelper.waitForSeconds(2);
1299        }
1300        handleLeaveAlert();
1301    }
1302
1303    /**
1304     * Click element in native way by Actions
1305     */
1306    public void clickAt() {
1307        clickAt("1,1");
1308    }
1309
1310    /**
1311     * Handle Confirmation pop up for Chrome and IE
1312     */
1313    protected void handleLeaveAlert() {
1314        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1315            ThreadHelper.waitForSeconds(2);
1316        }
1317        handleLeaveAlert();
1318    }
1319
1320    /**
1321     * Click element in native way by Actions
1322     */
1323    public void clickAt() {
1324        clickAt("1,1");
1325    }
1326
1327    /**
1328     * Handle Confirmation pop up for Chrome and IE
1329     */
1330    protected void handleLeaveAlert() {
1331        if (browser == BrowserType.Chrome || browser == BrowserType.InternetExplore) {
1332            ThreadHelper.waitForSeconds(2);
1333        }
1334        handleLeaveAlert();
1335    }
1336
1337    /**
1338     * Click element in native way by Actions
1339     */
1340    public void clickAt() {
1341        clickAt("1,1");
1342    }
1343
1344    /**
1345     * Handle Confirmation pop up for Chrome and IE
1346     */
1347    protected void handleLeaveAlert() {
134
```

```

175         try {
176             Alert alert = driver.switchTo().alert();
177             String text = alert.getText();
178             if (text.contains("Leave")) {
179                 alert.accept();
180             } else {
181                 ThreadHelper.waitForSeconds(2);
182             }
183             System.out.println("Get alert - " + text);
184         } catch (NoAlertPresentException e) {
185         }
186     }
187 }
188 /**
189 * Click element in native way by Actions
190 * @param class<code>
191 * @param clickAt{quot;1, 1quot;};
192 * </pre>
193 * @param value
194 */
195 public void click(String value) {
196     captureScreenshot("before clicking");
197     Logging.logInfo("click on " + toHTML());
198     findElement();
199     if (browser == BrowserType.Safari) {
200         element.click();
201     } else {
202         String[] parts = value.split(",");
203         int xOffset = Integer.parseInt(parts[0]);
204         int yOffset = Integer.parseInt(parts[1]);
205         try {
206             new Actions(driver).moveToElement(element, xOffset, yOffset)
207                 .click().perform();
208         } catch (InvalidOperationException e) {
209             e.printStackTrace();
210         }
211     }
212     try {
213         element.click();
214     } catch (ElementNotVisibleException env) {
215         // catching and re-wrapping by including useful information
216         // like what element is not visible.
217         throw new ElementNotVisibleException("Element " +
218             + by.toString() + " was not visible. " +
219             + env.getMessage(), env);
220     }
221 }
222 try {
223     if ((browser == BrowserType.Chrome || browser == BrowserType.InternetExplore)
224         && this.getDriver().switchTo().alert().getText()
225         .contains("Leave")) {
226         // handle Confirmation pop up in
227         // chrome,IE,5/6/2013,Java
228         this.getDriver().switchTo().alert().accept();
229     }
230 } catch (NoAlertPresentException e) {
231 }
232 }
233 }
234 }
235 /**
236 * public void simulateClick() {
237 *     findElement();
238 *     String mouseoverScript = "if(document.createEvent){var evObj = document.createEvent('MouseEvents');evObj.initEvent('mouseover', true, false); arguments[0].dispatchEvent(evObj);} else if(document.createEventObject) {
239 *         JavascriptExecutor js = (JavascriptExecutor) driver;
240 *         js.executeScript(mouseoverScript, element);
241 *     ThreadHelper.waitForSeconds(2);
242 *     String clickScript = "if(document.createEvent){var evObj = document.createEvent('MouseEvents');evObj.initEvent('click', true, false); arguments[0].dispatchEvent(evObj);} else if(document.createEventObject) { arguments[0].fireEvent('onmouseup');";
243 *         js.executeScript(clickScript, element);
244 *     ThreadHelper.waitForSeconds(2);
245 * }
246 */
247 public void simulateMoveToElement(int x, int y) {
248     findElement();
249     (JavascriptExecutor) driver
250         .executeScript(
251             "function simulate(f,c,d,e){var b=a=null;for(b in eventMatchers)if(eventMatchers[b].test(c)){o=b;break;}if(o)return 1;document.createEvent(o,o=='HTMLEvents'?'b.initEvent(' +
252                 + "simulate(arguments[0], 'mousemove', arguments[1],arguments[2]);",
253                 element, x, y);
254 }
255 */
256 /**
257 * Finds the element using By type. Implicit Waits is built in
258 * createWebDriver in WebUI WebDriver to handle dynamic element problem. This
259 * method is used instead of all the basic operations like click, sendKeys,
260 * getAttribute etc. due to better performance. Even it waits to deal with
261 * special element which needs long time to present.
262 */
263 protected void findElement() {
264     driver = WebUI.driver.getWebDriver();
265     if (browser == BrowserType.InternetExplore) {
266         try {
267             Thread.sleep(500);
268         } catch (InterruptedException e) {
269         }
270     }
271     try {
272         if (by != null) {
273             // catching NoSuchElementException and wrapping up with error
274             // message to know what by value causing this exception.
275             try {
276                 element = driver.findElement(by);
277             } catch (NoSuchElementException e1) {
278                 throw new NoSuchElementException("Element " +
279                     + by.toString() + " was not found. " +
280                     + e1.getMessage(), e1);
281             }
282         }
283         // Don't try to find element again for null by
284         // catch NoSuchElementException
285         // ignore the NoSuchElementException for IDE driver
286         ThreadHelper.waitForSeconds(1);
287     }
288 }
289 /**
290 * Fires a Javascript event on the underlying element
291 * @param eventName
292 * @param value
293 */
294 public void fireEvent(String eventName) {
295     findElement();
296     try {
297         JavascriptLibrary jslib = new JavascriptLibrary();
298

```

```

if(document.createEvent){var evObj = document.createEvent('MouseEvents');evObj.initEvent('mouseover', true, false); arguments[0].dispatchEvent(evObj);} else if(document.createEventObject) { arguments[0].fireEvent('onmouseover');}"
```

```

if(document.createEvent){var evObj = document.createEvent('MouseEvents');evObj.initEvent('click', true, false); arguments[0].dispatchEvent(evObj);} else if(document.createEventObject) { arguments[0].fireEvent('onclick');};}"
```

```

function simulate(f,c,d,e){var b,a=null;for(b in
eventMatchers)if(eventMatchers[b].test(c)){a=b;break}if(!a)return!1;document.createEvent?(b=document.createEvent(a),a==
\"HTMLEvents\"?b.initEvent(c,!0,!0):b.initMouseEvent(c,!0,!0,document.defaultView,0,d,e,d,e,!1,!1,!1,!1,0,null),f.dispa
tchEvent(b)):({=document.createEventObject(),a.detail=0,a.screenX=d,a.screenY=e,a.clientX=d,a.clientY=e,a.ctrlKey=!1,a.
altKey=!1,a.shiftKey=!1,a.metaKey=!1,a.button=1,f.fireEvent(\"on\"+c,a));return!0} var
eventMatchers={HTMLEvents:/^(:load|unload|abort|error|select|change|submit|reset|focus|blur|resize|scroll)$/,MouseEven
ts:/^(?:click|dblclick|mouse(?:down|up|over|move|out))$/}; "

```

The screenshot shows an IDE interface with a code editor displaying Java code. The code is part of a class named `HtmlElement`. The code includes various methods for interacting with web elements, such as `simulateMoveToElement(int, int)`, `getElements(By)`, `getElements(By)` (which returns a list of `HtmlElement` objects), `getAttribute(String)`, `getBy()`, and `getCssValue(String)`. The code also handles JavaScript execution using a `JavascriptLibrary`. The IDE's toolbar and sidebar are visible, showing other files and project structure.

```

1  package com.ebay.mau.driver.web.element;
2
3  import org.openqa.selenium.By;
4  import org.openqa.selenium.WebElement;
5  import org.openqa.selenium.interactions.Actions;
6  import org.openqa.selenium.support.ui.ExpectedConditions;
7  import org.openqa.selenium.support.ui.WebDriverWait;
8
9  import java.util.List;
10
11  public class HtmlElement {
12
13     protected WebElement element;
14
15     public HtmlElement(WebElement element) {
16         this.element = element;
17     }
18
19     /**
20      * Simulates a mouse move to the element
21      * @param x
22      * @param y
23      */
24     public void simulateMoveToElement(int x, int y) {
25
26         JavascriptLibrary jsLib = new JavascriptLibrary();
27
28         try {
29             jsLib.callEmbeddedSelenium(driver, "doFireEvent", element,
30                                         eventNone);
31         } catch (Exception e) {
32             // Handle OperaDriver
33         }
34     }
35
36     /**
37      * Get all elements in the current page with same locator
38      * @param by
39      * @return
40      */
41     public List<WebElement> getElements(By by) {
42         findElement();
43         return driver.findElements(by);
44     }
45
46     /**
47      * Get all HTML elements in the current page with same locator
48      * @param by
49      * @return
50      */
51     public List<HtmlElement> getElements(By by) {
52         findElement();
53         List<WebElement> elements = driver.findElements(by);
54         List<HtmlElement> es = new ArrayList<HtmlElement>();
55         for (int i = 0; i < elements.size(); i++) {
56             es.add(new HtmlElement(elements.get(i)));
57         }
58         return es;
59     }
60
61     /**
62      * Gets an attribute (using standard key-value pair) from the underlying
63      * attribute
64      * @param name
65      * @return
66      */
67     public String getAttribute(String name) {
68         findElement();
69         return element.getAttribute(name);
70     }
71
72     /**
73      * Returns the BY locator stored in the HtmlElement
74      * @return
75      */
76     public By getBy() {
77         return by;
78     }
79
80     /**
81      * Returns the value for the specified CSS key
82      * @param propertyName
83      * @return
84      */
85     public String getCssValue(String propertyName) {
86         findElement();
87         return element.getCssValue(propertyName);
88     }
89
90 }

```

Java EE Debug

```

365 /**
366 * Get and refresh underlying WebDriver
367 */
368 protected WebDriver getDriver() {
369     return WebDDriver.getWebDriver();
370 }
371 /**
372 * Returns the underlying WebDriver WebElement
373 */
374 public WebElement getElement() {
375     if (by != null)
376         element = driver.findElement(by);
377     return element;
378 }
379 /**
380 * Executes the given JavaScript against the underlying WebElement
381 */
382 @Script
383 public String getEval(String script) {
384     String name = (String) ((JavascriptExecutor) driver).executeScript(
385         script, element);
386     return name;
387 }
388 /**
389 * Returns the 'height' property of the underlying WebElement's Dimension
390 */
391 public int getHeight() {
392     findElement();
393     return element.getSize().getHeight();
394 }
395 /**
396 * Returns the label used during initialization
397 */
398 public String getLabel() {
399     return label;
400 }
401 /**
402 * Gets the Point location of the underlying WebElement
403 */
404 public Point getLocation() {
405     findElement();
406     return element.getLocation();
407 }
408 /**
409 * Returns the locator used to find the underlying WebElement
410 */
411 public Locator getLocator() {
412     return locator;
413 }
414 /**
415 * Returns the Dimension property of the underlying WebElement
416 */
417 public Dimension getSize() {
418     findElement();
419     return element.getSize();
420 }
421 /**
422 * Returns the HTML Tag for the underlying WebElement (div, a, input, etc)
423 */
424 public String getTagName() {
425     findElement();
426     return element.getTagName();
427 }
428 /**
429 * Returns the text body of the underlying WebElement
430 */
431 public String getText() {
432     findElement();
433     return element.getText();
434 }
435 /**
436 * Returns the 'value' attribute of the underlying WebElement
437 */
438 public String getValue() {
439     findElement();
440     return element.getAttribute("value");
441 }

```

Read-Only Smart Insert 251 : 698

Java EE Debug

```

422 public String getLocator() {
423     return locator;
424 }
425 /**
426 * Returns the By locatorBy(String locator, LocatorType locatorType)
427 */
428 private By getLocatorBy(String locator, LocatorType locatorType) {
429     switch (locatorType) {
430         case ID:
431             return By.id(locator);
432         case NAME:
433             return By.name(locator);
434         case CLASS_NAME:
435             return By.className(locator);
436         case LINK_TEXT:
437             return By.linkText(locator);
438         case PARTIAL_LINK_TEXT:
439             return By.partialLinkText(locator);
440         case CSS_SELECTOR:
441             return By.cssSelector(locator);
442         case TAG_NAME:
443             return By.tagName(locator);
444         default:
445             return By.xpath(locator);
446     }
447 }
448 /**
449 * Returns the Dimension property of the underlying WebElement
450 */
451 public Dimension getSize() {
452     findElement();
453     return element.getSize();
454 }
455 /**
456 * Returns the HTML Tag for the underlying WebElement (div, a, input, etc)
457 */
458 public String getTagName() {
459     findElement();
460     return element.getTagName();
461 }
462 /**
463 * Returns the text body of the underlying WebElement
464 */
465 public String getText() {
466     findElement();
467     return element.getText();
468 }
469 /**
470 * Returns the 'value' attribute of the underlying WebElement
471 */
472 public String getValue() {
473     findElement();
474     return element.getAttribute("value");
475 }

```

Read-Only Smart Insert 251 : 698

Java EE Debug

Quick Access

mauicomponent-ebayIndia > /Users/gaurshukla/m2/raptor2/com/ebay/maui/maulplatform/1.2.1/maulplatform-1.2.1.jar > com.ebay.maui.driver.web.element > HtmlElement.java > simulateMoveToElement(int, int) : void

```

493 /**
494 * Returns the 'width' property of the underlying webElement's Dimension
495 *
496 * @return
497 */
498 public int getWidthO {
499     findElement();
500     return element.getSizeO.getWidthO;
501 }
502 /**
503 * Refreshes the WebUIORiver before locating the element, to ensure we have
504 * the current version (useful for when the state of an element has changed
505 * via AJAX or non-page-turn action)
506 */
507 public void initO {
508     driver = WebUIORiver.getWebDriverO;
509     element = driver.findElement(by);
510 }
511 /**
512 * Indicates whether or not the web element is currently displayed in the
513 * browser
514 *
515 * @return
516 */
517 public boolean isDisplayedO {
518     findElement();
519     try {
520         return element.isDisplayedO;
521     } catch (StaleElementReferenceException e) {
522         return false;
523     }
524 }
525 /**
526 * Searches for the element using the by locator, and indicates whether or
527 * not it exists in the page. This can be used to look for hidden objects,
528 * whereas isDisplayedO only looks for things that are visible to the user
529 */
530 public boolean isElementPresentO {
531     if (WebUIORiver.getWebDriverO == null) {
532         Logging.log("Web Driver is terminated! Exception might caught in last action.");
533         throw new RuntimeException(
534             "Web Driver is terminated! Exception might caught in last action.");
535     }
536     int count = 0;
537     try {
538         count = WebUIORiver.getWebDriverO.findElements(by).sizeO;
539     } catch (NoSuchElementException e) {
540         if (e instanceof InvalidSelectorException) {
541             Logging.log("not InvalidSelectorException, retry");
542             ThreadHelper.waitForSeconds(2);
543             count = WebUIORiver.getWebDriverO.findElements(by).sizeO;
544         } else if (e.getMessage() != null
545                   && e.getMessage().contains(
546                       "com.google.common.collect.Maps$EntrySetIterator cannot be cast to java.util.List")) {
547             Logging.log("got NoSuchElementException, retry");
548             ThreadHelper.waitForSeconds(2);
549             count = WebUIORiver.getWebDriverO.findElements(by).sizeO;
550         } else if (e instanceof NoSuchElementException) // handle some more
551             ;
552     }
553 }
554 /**
555 * Indicates whether or not the element is enabled in the browser
556 */
557 public boolean isEnabledO {
558     findElement();
559     return element.isEnabledO;
560 }
561 /**
562 * Indicates whether or not the element is selected in the browser
563 */
564 public boolean isSelectedO {
565     findElement();
566     return element.isSelectedO;
567 }
568 /**
569 * Whether or not the indicated text is contained in the element's getText()
570 * attribute
571 */
572 public boolean isTextPresent(String text) {
573     findElement();
574     return element.getTextO.contains(text);
575 }
576 /**
577 * Forces a mouseDown event on the WebElement
578 */
579 public void mouseDownO {
580     Logging.log("mouseDown = " + this.toString());
581     findElement();
582     Mouse mouse = ((HasInputDevices) driver).getMouse();
583 }
584 
```

Read-Only Smart Insert 251:698

Java EE Debug

Quick Access

BaseWebUtil.java > /Users/gaurshukla/m2/raptor2/com/ebay/maui/maulplatform/1.2.1/maulplatform-1.2.1.jar > com.ebay.maui.driver.web.element > HtmlElement.java > simulateMoveToElement(int, int) : void

```

255 /**
256 * Waits for the element to be present in the DOM
257 */
258 public void waitForElementO {
259     ThreadHelper.waitForSeconds(2);
260     Count = WebUIORiver.getWebDriverO.findElements(by).sizeO;
261     if (Count > 0) {
262         // handle some page
263         // throwing this
264         // error for
265         // IE
266         if (e.getMessage().contains(
267             "Finding elements returned an unexpected error")) {
268             try {
269                 if (Centor.findElement(by) != null)
270                     Count = 1;
271             } catch (NoSuchElementException ex2) {
272                 // ignore
273             }
274         } else {
275             throw e;
276         }
277     }
278     if (Count == 0)
279         return false;
280     return true;
281 }
282 /**
283 * Indicates whether or not the element is enabled in the browser
284 */
285 public boolean isEnabledO {
286     findElement();
287     return element.isEnabledO;
288 }
289 /**
290 * Indicates whether or not the element is selected in the browser
291 */
292 public boolean isSelectedO {
293     findElement();
294     return element.isSelectedO;
295 }
296 /**
297 * Whether or not the indicated text is contained in the element's getText()
298 * attribute
299 */
300 public boolean isTextPresent(String text) {
301     findElement();
302     return element.getTextO.contains(text);
303 }
304 /**
305 * Forces a mouseDown event on the WebElement
306 */
307 public void mouseDownO {
308     Logging.log("mouseDown = " + this.toString());
309     findElement();
310     Mouse mouse = ((HasInputDevices) driver).getMouse();
311 }
312 
```

Read-Only Smart Insert 251:698

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Shows "Read-Only", "Smart Insert", "251:698", "Quick Access", and "Java EE".
- Toolbar:** Includes icons for file operations (New, Open, Save, Print), search, and other common functions.
- Project Explorer:** Lists projects like "BaseWebUtil.java", "BrowserType.java", "CustomEventFir", "IScreenshotList", "ScreenShot.java", "BasePageObject.java", "Button.java", "CheckBox.java", "Form.java", and "HtmlElement.java".
- Code Editor:** Displays the source code for the "HtmlElement.java" class. The code includes methods for sending keys, sleeping, converting to friendly strings, and waiting for elements to be present. It also contains annotations like `@Param` and `@Param ArgB`.
- Outline View:** Shows the class structure with nodes for `HtmlElement`, `SimpleElement`, `ComplexElement`, and `CompositeElement`.
- Search View:** Shows results for "simulateMoveToElement(int, int) : void".

Project Explorer   Navigator   Type Hierarchy

com.ebay.maul.driver.web.element

```

1 package com.ebay.maul.driver.web.element;
2
3 import org.openqa.selenium.*;
4
5 public class Image extends HtmlElement {
6
7     public Image(String label, By by) {
8         super(label, by);
9     }
10
11    public Image(String label, String locator) {
12        super(label, locator);
13    }
14
15    public int getHeight() {
16        return super.getSize().getHeight();
17    }
18
19    public int getWidth() {
20        return super.getSize().getWidth();
21    }
22
23    public String getUrl(){
24        return super.getAttribute("src");
25    }
26 }

```

Image

- Image(String, By)
- Image(String, String)
- getHeight() - int
- getWidth() - int
- getUrl() - String

Markers Properties Servers Data Source Explorer Snippets Console Progress Search Results of running class RaiseClaimTest Package Explorer

No operations to display at this time.

Read-Only Smart Insert 1:1

Project Explorer   Navigator   Type Hierarchy

com.ebay.maul.driver.web.element

```

1 package com.ebay.maul.driver.web.element;
2
3 /**
4 * This is the page interface. It will be useful for testing Global Headers,
5 * RTM, SEO, Java Script and Accessibility.
6 */
7
8 public interface IPage {
9
10    String getBodyText();
11
12    String getHtmlSavedToPath();
13
14    String getHTMLSource();
15
16    String getJSErrors();
17
18    String getLocation();
19
20    String getLogId();
21
22    String getTitle();
23
24 }
25

```

IPage

- getBodyText() - String
- getHtmlSavedToPath() - S
- getHTMLSource() - String
- getJSErrors() - String
- getLocation() - String
- getLogId() - String
- getTitle() - String

Markers Properties Servers Data Source Explorer Snippets Console Progress Search Results of running class RaiseClaimTest Package Explorer

No operations to display at this time.

Read-Only Smart Insert 1:1

The image displays two screenshots of an IDE interface, likely Eclipse, showing code editors for two Java classes: `Label.class` and `Link.class`.

**Top Screenshot (Label.class):**

```

1 package com.ebay.maul.driver.web.element;
2
3 import org.openqa.selenium.By;
4
5 import com.ebay.maul.controller.Assertion;
6 import com.ebay.maul.controller.Logging;
7
8 public class Label extends HtmlElement {
9     public Label(String label, By by) {
10         super(label, by);
11     }
12
13     public Label(String label, String locator) {
14         super(label, locator);
15     }
16
17     @Override
18     public String getText() {
19         Logging.logWebStep(null, "get text from " + toHTML(), false);
20         return super.getText();
21     }
22
23     public boolean isTextPresent(String pattern){
24         String text = getText();
25         return (text != null && (text.contains(pattern) || text.matches(pattern)));
26     }
27
28     @Deprecated
29     public String getExpectedText()
30     {
31         Assertion.assertTrue(false,"Not supported!");
32         return null;
33     }
34 }

```

**Bottom Screenshot (Link.class):**

```

1 package com.ebay.maul.driver.web.element;
2
3 import org.openqa.selenium.By;
4
5 import com.ebay.maul.controller.Logging;
6 import com.ebay.maul.helper.URLHelper;
7
8 public class Link extends HtmlElement {
9     public Link(String label, By by) {
10         super(label, by);
11     }
12
13     public Link(String label, String locator) {
14         super(label, locator);
15     }
16
17     @Override
18     public void click() {
19         Logging.logWebStep(null, "click on " + toHTML(), false);
20         super.click();
21
22         /**
23          * Checks if the link is not dead
24          */
25         public boolean isValid() {
26             try {
27                 URLHelper.opengetAttribute("href");
28             } catch (Exception e) {
29                 return false;
30             }
31             return true;
32         }
33
34     }
35
36     public String getUrl(){
37         return super.getAttribute("href");
38     }
39 }
40

```

The interface includes a Project Explorer, Navigator, Type Hierarchy, and various toolbars. The code editors show Java code with syntax highlighting and code completion suggestions. The bottom of each editor shows a status bar with "Read-Only", "Smart Insert", and "1 : 1".

```

1  package com.ebay.maui.driver.web.element;
2
3  import java.util.HashMap;
4  import java.util.Map;
5
6  import org.openqa.selenium.By;
7  import org.openqa.selenium.WebElement;
8
9  public class LocatorSwitch {
10
11     private LocatorType appFrameworkType = null;
12     private final Map<String, By> LOCATOR_MAP = new HashMap<String, By>();
13     private boolean isAppFrameworkValidationComplete = false;
14
15     private LocatorType projectAppFrameworkType = null;
16
17     public LocatorType getAppFrameworkType() {
18         return appFrameworkType;
19     }
20
21     public LocatorType getProjectAppFrameworkType() {
22         return projectAppFrameworkType;
23     }
24
25     public void setAppFrameworkType(LocatorType appFrameworkType) {
26         this.appFrameworkType = appFrameworkType;
27     }
28
29     public void setProjectAppFrameworkType(LocatorType projectAppFrameworkType) {
30         this.projectAppFrameworkType = projectAppFrameworkType;
31     }
32
33     public void setLocator(LocatorType pageType, String elementName, By locator) {
34         validateAndAssignPageType();
35         LOCATOR_MAP.put(pageType.toString() + elementName, locator);
36     }
37
38     /**
39      * Returns the (@link By) locator for the given elementString.
40      */
41     public By getLocator(String elementString) {
42         By locator = grabLocatorFromLocatorMap(elementString, LocatorType.COMMON_LOCATOR);
43         if (null == locator) {
44             locator = grabLocatorFromLocatorMap(elementString, getAppFrameworkType());
45         }
46         if (null == locator) {
47             locator = grabLocatorFromLocatorMap(elementString, getProjectAppFrameworkType());
48         }
49         if (null != locator) {
50             return locator;
51         }
52         // If the locator cannot be found in the LOCATOR_MAP and if it is not a
53         // COMMON_LOCATOR, then the locator is not found,
54         // So throw a RuntimeException saying so.
55         throw new RuntimeException("Could not find Locators for " + elementString
56         + " int " + getAppFrameworkType()
57         + ". Please verify if the locator is set in "
58         + new Throwable().fillInStackTrace().getClassName());
59     }
60
61     private By grabLocatorFromLocatorMap(String elementString, LocatorType pageType) {
62         @SuppressWarnings("rawtypes")
63         final Iterator<Object> iterator = LOCATOR_MAP.keySet().iterator();
64         // Iterate through the LOCATOR_MAP and try to find the elementString based on the pageType
65         while (null != pageType && iterator.hasNext()) {
66             final String key = iterator.next().toString();
67             if (key.equals(pageType.toString() + elementString)) {
68                 return LOCATOR_MAP.get(key);
69             }
70         }
71         return null;
72     }
73
74     /**
75      * Validates the APP_FRAMEWORK flag for tests.
76      */
77     private void validateAndAssignPageType() {
78         String pageTitleFromSystemProperty = null;
79         if (!isAppFrameworkValidationComplete) {
80             pageTitleFromSystemProperty = System.getProperty("APPLICATION_FRAMEWORK");
81             for (LocatorType pageType : LocatorType.values()) {
82                 if (pageTypeFromSystemProperty.equalsIgnoreCase(pageType.name())) {
83                     setPageType(pageType);
84                     isAppFrameworkValidationComplete = true;
85                     break;
86                 }
87             }
88             if (!isAppFrameworkValidationComplete && getPageType() != null) {
89                 return;
90             }
91             isAppFrameworkValidationComplete = true;
92             throw new RuntimeException("Invalid APPLICATION FRAMEWORK Code:" + pageTitleFromSystemProperty
93             + ". Supported FRAMEWORKS are: CLASSIC, RAPTOR OR RETA."
94             + "-Do you use setLocator and forgot to specify the APPLICATION_FRAMEWORK.");
95         }
96     }
97
98     /**
99      * Sets the page type for the application framework.
100     */
101    public void setPageType(LocatorType pageType) {
102        this.pageType = pageType;
103    }
104
105    public LocatorType getPageType() {
106        return pageType;
107    }
108
109    public void validatePageType() {
110        validateAndAssignPageType();
111    }
112

```

```

1 package com.ebay.mau.driver.web.element;
2
3 /**
4  * Enum for various application versions where tests can run. This helps saving time and LOC by eliminating needs to have multiple versions
5  * of page objects for various UI experiences and to some extent helps avoiding re-writing tests to test same functionalities
6  * in various versions of pages just because the locators "By" (css, xpath etc) changes.
7  */
8 /**
9  * use COMMON_LOCATOR in cases where the locator stays same across all versions of your web app or devices.<br>
10 * - RAPTOR_LOCATOR for raptor locators.<br>
11 * - IPHONE_LOCATOR for iphone version of your webpage and so on.<br>
12 * - BETA or ALPHA if you have any experimental version of your webpage.<br>
13 * - Several other locator types are available for various devices too.
14 */
15 *
16 * @author arunkannan (Aravind Kannan)
17 */
18
19 public enum LocatorType {
20     CLASSIC_LOCATOR("CLASSIC"),
21     COMMON_LOCATOR("COMMON"),
22     RAPTOR_LOCATOR("RAPTOR"),
23     BETA_LOCATOR("BETA"),
24     ALPHA_LOCATOR("ALPHA"),
25     /**
26      * Locator type to define a element that can be located by some
27      * element irrespective of browser sizes in desktop.
28      */
29     DESKTOP_LOCATOR("DESKTOP"),
29
30     /**
31      * Locator type to define a element that has specific locator for small size browser due to rendering of a different UI experience.
32      * "small" is dependent on individual projects on how they define it.Normal guide line is < 1024. Browser size can be controlled in your test run
33      * on run time. <a href="https://wiki.vip.corp.ebay.com/display/VERTQA/Cotrolling-Browser+Size">Click to see how.</a>
34     */
35     DESKTOP_SMALL_LOCATOR("DESKTOPSMALL"),
36
37     /**
38      * Locator type to define a element that has specific locator for medium size browser due to rendering of a different UI experience.
39      * "Medium" is dependent on individual projects on how they define it. Browser size can be controlled in your test run
40      * on run time. <a href="https://wiki.vip.corp.ebay.com/display/VERTQA/Cotrolling-Browser+Size">Click to see how.</a>
41     */
42     DESKTOP_MEDIUM_LOCATOR("DESKTOPMEDIUM"),
43
44     /**
45      * Locator type to define a element that has specific locator for large size browser due to rendering of a different UI experience.
46      * "Large" is dependent on individual projects on how they define it. Normal guide line is > 1024. Browser size can be controlled in your test run
47      * on run time. <a href="https://wiki.vip.corp.ebay.com/display/VERTQA/Cotrolling-Browser+Size">Click to see how.</a>
48     */
49     DESKTOP_LARGE_LOCATOR("DESKTOPLARGE"),
50     /**
51      * Locator type to define a element that can be located by some
52      * element irrespective of los device type or orientation of the device
53      * (i.e. Landscape or Portrait/ iphone or ipod)
54     */
55     IOS_LOCATOR("IOS"),
56
57     /**
58      * Locator type to define a element that can be located by some
59      * element irrespective of device but specific to portrait orientation.
60     */
61     IOS_PORTRAIT("IOSPORTRAIT"),
62     /**
63      * Locator type to define a element that can be located by some
64      * element irrespective of device but specific to landscape orientation.
65     */
65

```

```

66     IOS_LANDSCAPE("IOSLANDSCAPE"),
67     /**
68      * Locator type to define a element that can be located by some
69      * element irrespective of device or orientation.
70     */
71     IPHONE_LOCATOR("IPHONE"),
72     /**
73      * Locator type to define a element that has specific locator for iphone
74      * + landscape mode,
75      */
76     IPHONE_LANDSCAPE_LOCATOR("IPHONELANDSCAPE"),
77     /**
78      * Locator type to define a element that has specific locator for iphone
79      * + portrait mode.
80     */
81     IPHONE_PORTRAIT_LOCATOR("IPHONEPORTRAIT"),
82     /**
83      * Locator type to define a element that can be located by some
84      * element irrespective of orientation of the ipad (i.e. Landscape or Portrait)
85     */
86     IPAD_LOCATOR("IPAD"),
87     /**
88      * Locator type to define a element that has specific locator for ipad
89      * + landscape mode,
90      */
91     IPAD_LANDSCAPE_LOCATOR("IPADLANDSCAPE"),
92     /**
93      * Locator type to define a element that has specific locator for ipad
94      * + portrait mode.
95     */
96     IPAD_PORTRAIT_LOCATOR("IPADPORTRAIT"),
97     /**
98      * Locator type to define a element that can be located by some
99      * element irrespective of orientation of the ipad (i.e. Landscape or Portrait)
100 */
101     IPADMINI_LOCATOR("IPADMINI"),
102     /**
103      * Locator type to define a element that has specific locator for ipad
104      * + landscape mode,
105      */
106     IPADMINI_LANDSCAPE_LOCATOR("IPADMINILANDSCAPE"),
107     /**
108      * Locator type to define a element that has specific locator for ipad
109      * + portrait mode.
110     */
111     IPADMINI_PORTRAIT_LOCATOR("IPADMINIPORTRAIT"),
112     /**
113      * Locator type to define a element that can be located by some
114      * element irrespective of orientation or size of the android device
115      * (i.e. Landscape or Portrait or any sizes)
116      */
117     ANDROID_LOCATOR("ANDROID"),
118     /**
119      * Locator type to define a element that can be located by some
120      * element irrespective of size of the android device
121      * but, specific to portrait orientation.
122     */
123     ANDROID_PORTRAIT_LOCATOR("ANDROID_PORTRAIT"),
124     /**
125      * Locator type to define a element that can be located by some
126      * element irrespective of size of the android device
127      * but, specific to landscape orientation.
128      */
129     ANDROID_LANDSCAPE_LOCATOR("ANDROID_LANDSCAPE_PORTRAIT"),
130     /**

```

```

128 /**
129 * Locator type to define a element that has specific locator for android 2 inch devices
130 * irrespective of device orientation
131 */
132 /**
133 * Locator type to define a element that has specific locator for android 2 inch devices
134 * irrespective of device orientation
135 */
136 /**
137 * Locator type to define a element that has specific locator for android 2 inch devices
138 * landscape mode.
139 */
140 /**
141 * Locator type to define a element that has specific locator for android 2 inch devices
142 * portrait mode.
143 */
144 /**
145 * Locator type to define a element that has specific locator for android 4 inch devices
146 * irrespective of device orientation
147 */
148 /**
149 * Locator type to define a element that has specific locator for android 4 inch devices
150 */
151 /**
152 * Locator type to define a element that has specific locator for android 4 inch devices
153 * landscape mode.
154 */
155 /**
156 * Locator type to define a element that has specific locator for android 4 inch devices
157 * portrait mode.
158 */
159 /**
160 * Locator type to define a element that has specific locator for android 7 inch devices
161 * irrespective of device orientation
162 */
163 /**
164 * Locator type to define a element that has specific locator for android 7 inch devices
165 */
166 /**
167 * Locator type to define a element that has specific locator for android 7 inch devices
168 * landscape mode.
169 */
170 /**
171 * Locator type to define a element that has specific locator for android 7 inch devices
172 * portrait mode.
173 */
174 /**
175 * Locator type to define a element that has specific locator for android 10 inch devices
176 * irrespective of device orientation
177 */
178 /**
179 * Locator type to define a element that has specific locator for android 10 inch devices
180 */
181 /**
182 * Locator type to define a element that has specific locator for android 10 inch devices
183 * landscape mode.
184 */
185 /**
186 * Locator type to define a element that has specific locator for android 10 inch devices
187 */
188 /**
189 * Locator type to define a element that has specific locator for android 10 inch devices
190 */
191 /**
192 */

```

Read-Only Smart Insert 1 : 1

```

141 /**
142 * Locator type to define a element that has specific locator for android 2 inch devices
143 * portrait mode.
144 */
145 /**
146 * Locator type to define a element that has specific locator for android 4 inch devices
147 * irrespective of device orientation
148 */
149 /**
150 * Locator type to define a element that has specific locator for android 4 inch devices
151 * landscape mode.
152 */
153 /**
154 * Locator type to define a element that has specific locator for android 4 inch devices
155 */
156 /**
157 * Locator type to define a element that has specific locator for android 4 inch devices
158 * portrait mode.
159 */
160 /**
161 * Locator type to define a element that has specific locator for android 7 inch devices
162 * irrespective of device orientation
163 */
164 /**
165 * Locator type to define a element that has specific locator for android 7 inch devices
166 * landscape mode.
167 */
168 /**
169 * Locator type to define a element that has specific locator for android 7 inch devices
170 * portrait mode.
171 */
172 /**
173 * Locator type to define a element that has specific locator for android 10 inch devices
174 * irrespective of device orientation
175 */
176 /**
177 * Locator type to define a element that has specific locator for android 10 inch devices
178 * landscape mode.
179 */
180 /**
181 * Locator type to define a element that has specific locator for android 10 inch devices
182 * portrait mode.
183 */
184 /**
185 * Locator type to define a element that has specific locator for android 10 inch devices
186 */
187 /**
188 * Locator type to define a element that has specific locator for android 10 inch devices
189 */
190 /**
191 */
192 /**
193 */
194 private String pagetype;
195 /**
196 * private LocatorType(final String locatorType) {
197 * this.pagetype = locatorType;
198 * }
199 */
200 /**
201 * @Override
202 * public String toString() {
203 * return pagetype;
204 * }
205 */

```

Read-Only Smart Insert 1 : 1

Screenshot of an IDE showing the Java code for the `RadioButton` class. The code implements the `WebElement` interface and extends `HtmlElement`. It includes methods for checking and clicking the radio button, and overriding the `click()` method to log the action.

```

1 package com.ebay.mau.driver.web.element;
2
3 import org.openqa.selenium.By;
4
5 import com.ebay.mau.controller.Logging;
6
7 public class RadioButton extends HtmlElement {
8
9     public RadioButton(String label, By by) {
10         super(label, by);
11     }
12
13     public RadioButton(String label, String locator) {
14         super(locator);
15     }
16
17     public void check() {
18         Logging.logWebStep(null, "check " + toHTML(), false);
19         super.click();
20     }
21
22     @Override
23     public void click() {
24         Logging.logWebStep(null, "click on " + toHTML(), false);
25         super.click();
26     }
27
28     // TODO this code is repeated in other objects (like CheckBox)--there should be an AbstractClickable
29     // and implement where this gets specified and implemented
30     public boolean isSelected() {
31         findElement();
32         return element.isSelected();
33     }
34 }

```

Screenshot of an IDE showing the Java code for the `SelectList` class. This class extends `HtmlElement` and provides methods for selecting options by text, index, or value. It also handles deselection operations.

```

18 public class SelectList extends HtmlElement {
19
20     protected Select select = null;
21     protected List<WebElement> options = null;
22
23     public SelectList(String text, By by) {
24         super(text, by);
25     }
26
27     public SelectList(String text, String locator) {
28         super(text, locator);
29     }
30
31     /**
32      * De-selects all options in a multi-select list element
33      */
34     public void deselectAll() {
35         Logging.logWebStep(null, "deselect all options on " + toHTML(), false);
36         findElement();
37         if (!isMultiple()) {
38             throw new UnsupportedOperationException("You may only deselect all options of a multi-select");
39         }
40
41         for (WebElement option : options) {
42             if (!option.isSelected()) {
43                 option.click();
44             }
45         }
46     }
47
48     public void deselectByIndex(int index) {
49         Logging.logWebStep(null, "deselect index" + index + " on " + toHTML(), false);
50         findElement();
51         WebElement option = options.get(index);
52         if (option.isSelected()) {
53             option.click();
54         }
55     }
56
57     public void deselectByText(String text) {
58         Logging.logWebStep(null, "deselect text" + text + " on " + toHTML(), false);
59         findElement();
60         for (WebElement option : options) {
61             if (option.getAttribute("text").equals(text)) {
62                 if (option.isSelected()) {
63                     option.click();
64                 }
65                 break;
66             }
67         }
68     }
69
70     public void deselectByValue(String value) {
71         Logging.logWebStep(null, "deselect value" + value + " on " + toHTML(), false);
72         findElement();
73         for (WebElement option : options) {
74             if (option.getAttribute("value").equals(value)) {
75                 if (option.isSelected()) {
76                     option.click();
77                 }
78                 break;
79             }
80         }
81     }

```

Java EE Debug

Quick Access

CheckBox.class Form.class HtmlElement.class Image.class IPage.class Label.class Link.class LocatorSwitch.c RadioButton.class SelectList.clas SelectList

```

83     protected void findElement() {
84         driver = WebDDriver.getWebDriver();
85         element = driver.findElement(this.getBy());
86         try {
87             select = getNewSelectElement(element);
88             options = select.getOptions();
89         } catch (UnexpectedTagNameException e) {
90             if (element.getTagName().equalsIgnoreCase("ul")) {
91                 options = element.findElements(By.tagName("li"));
92             }
93         }
94     }
95     /**
96      * Returns a new Select element (created to facilitate unit testing)
97      */
98     @return
99     protected Select getNewSelectElement(WebElement element) {
100        return new Select(element);
101    }
102    public List<WebElement> getOptions() {
103        findElement();
104        return options;
105    }
106    public String getSelectedText() {
107        findElement();
108        for (WebElement option : options) {
109            if (option.isSelected()) {
110                return option.getAttribute("text");
111            }
112        }
113        return null;
114    }
115    public String[] getSelectedTexts() {
116        findElement();
117        List<String> textlist = new ArrayList<String>();
118        for (WebElement option : options) {
119            if (option.isSelected()) {
120                textlist.add(option.getAttribute("text"));
121            }
122        }
123        String[] texts = new String[textlist.size()];
124        return textlist.toArray(texts);
125    }
126    public String[] getSelectedValues() {
127        findElement();
128        List<String> valueList = new ArrayList<String>();
129        for (WebElement option : options) {
130            if (option.isSelected()) {
131                valueList.add(option.getAttribute("value"));
132            }
133        }
134        return valueList.toArray(values);
135    }
136    public void init() {
137        super.init();
138        try {
139            select = getNewSelectElement(element);
140            options = select.getOptions();
141        } catch (UnexpectedTagNameException e) {
142            if (element.getTagName().equalsIgnoreCase("ul")) {
143                options = element.findElements(By.tagName("li"));
144            }
145        }
146    }
147    public boolean isMultiple() {
148        findElement();
149        String value = element.getAttribute("multiple");
150        return value != null && !value.equals("false");
151    }
152    public void selectByIndex(int index) {
153        Logging.logWebStep(null, "select index'" + index + "' on " + toHTML(), false);
154        findElement();
155        WebElement option = options.get(index);
156        setSelected(option);
157    }
158    public void selectByIndex(int[] indexes) {
159        Logging.logWebStep(null, "select index'" + indexes.toString() + "' on " + toHTML(), false);
160        findElement();
161        for (int i = 0; i < indexes.length; i++) {
162            WebElement option = options.get(indexes[i]);
163            setSelected(option);
164        }
165    }
166    /**
167     * Select standard select by attribute text, and select fake select with ul and li by attribute title
168     * @param text
169     */
170    public void selectByText(String text) {
171        Logging.logWebStep(null, "select text'" + text + "' on " + toHTML(), false);
172        findElement();
173        if(options!=null) {
174            for (WebElement option : options) {
175                if(option.getAttribute("title").equals(text)) {
176                    setSelected(option);
177                }
178            }
179        }
180        /**
181         * Select standard select by attribute text, and select fake select with ul and li by attribute title
182         * @param text
183         */
184        public void selectByAttribute(String attribute, String value) {
185            Logging.logWebStep(null, "select attribute'" + attribute + "' value'" + value + "' on " + toHTML(), false);
186            findElement();
187            if(options!=null) {
188                for (WebElement option : options) {
189                    if(option.getAttribute(attribute).equals(value)) {
190                        setSelected(option);
191                    }
192                }
193            }
194        }
195    }
196    /**
197     * Click on the selected option
198     */
199    public void click() {
200        findElement();
201        String selectedText = null;
202        if (options != null) {
203            selectedText = options.get(0).getAttribute("text");
204        }
205        for (WebElement option : options) {
206            if (selectedText.equals(option.getAttribute("text"))) {
207                setSelected(option);
208            }
209        }
210    }
211 }
```

Read-Only Smart Insert 1:1

Java EE Debug

Quick Access

CheckBox.class Form.class HtmlElement.class Image.class IPage.class Label.class Link.class LocatorSwitch.c RadioButton.class SelectList.clas SelectList

```

147     valueList.addOption(element.getAttribute("value"));
148 }
149 String[] values = new String[valueList.size()];
150 return valueList.toArray(values);
151 }
152 }
153
154 public void init() {
155     super.init();
156     try {
157         select = getNewSelectElement(element);
158         options = select.getOptions();
159     } catch (UnexpectedTagNameException e) {
160         if (element.getTagName().equalsIgnoreCase("ul")) {
161             options = element.findElements(By.tagName("li"));
162         }
163     }
164 }
165
166 public boolean isMultiple() {
167     findElement();
168     String value = element.getAttribute("multiple");
169     return value != null && !value.equals("false");
170 }
171
172 public void selectByIndex(int index) {
173     Logging.logWebStep(null, "select index'" + index + "' on " + toHTML(), false);
174     findElement();
175     WebElement option = options.get(index);
176     setSelected(option);
177 }
178
179 public void selectByIndex(int[] indexes) {
180     Logging.logWebStep(null, "select index'" + indexes.toString() + "' on " + toHTML(), false);
181     findElement();
182     for (int i = 0; i < indexes.length; i++) {
183         WebElement option = options.get(indexes[i]);
184         setSelected(option);
185     }
186 }
187
188 /**
189  * Select standard select by attribute text, and select fake select with ul and li by attribute title
190  * @param text
191  */
192 public void selectByText(String text) {
193     Logging.logWebStep(null, "select text'" + text + "' on " + toHTML(), false);
194     findElement();
195     if(options!=null) {
196         for (WebElement option : options) {
197             if(option.getAttribute("title").equals(text)) {
198                 setSelected(option);
199             }
200         }
201     }
202     if (options != null) {
203         for (WebElement option : options) {
204             if(option.getAttribute("text").equals(text)) {
205                 setSelected(option);
206             }
207         }
208     }
209 }
210
211 /**
212  * Click on the selected option
213  */
214 public void click() {
215     findElement();
216     String selectedText = null;
217     if (options != null) {
218         selectedText = options.get(0).getAttribute("text");
219     }
220     for (WebElement option : options) {
221         if (selectedText.equals(option.getAttribute("text"))) {
222             setSelected(option);
223         }
224     }
225 }
```

Read-Only Smart Insert 1:1

The screenshot shows the Java code for the `SelectList` class within the `com.ebay.maui.driver.web.element` package. The code handles various methods for interacting with dropdown menus, including selecting by text, value, or values, and setting the selected option.

```
197     driver.findElement(By.xpath("//li[text()='"+text+"']")).click();
198 }
199 
200     for (WebElement option : options) {
201         String selectedText = null;
202         if (option.getAttribute("label") != null)
203             selectedText = option.getAttribute("label");
204         else
205             selectedText = option.getAttribute("text");
206         if (selectedText.equals(text)) {
207             setSelected(option);
208             break;
209         }
210     }
211 }
212 
213 public void selectByText(String[] texts) {
214     Logging.logWebStep(null, "select texts:" + texts + " on " + toHTML(), false);
215     findElement();
216     for (int i = 0; i < texts.length; i++) {
217         for (WebElement option : options) {
218             if (option.getAttribute("text").equals(texts[i])) {
219                 setSelected(option);
220                 break;
221             }
222         }
223     }
224 }
225 
226 public void selectByValue(String value) {
227     Logging.logWebStep(null, "select value:" + value + " on " + toHTML(), false);
228     findElement();
229     for (WebElement option : options) {
230         if (option.getAttribute("value").equals(value)) {
231             setSelected(option);
232             break;
233         }
234     }
235 }
236 
237 public void selectByValues(String[] values) {
238     Logging.logWebStep(null, "select values:" + values + " on " + toHTML(), false);
239     findElement();
240     for (int i = 0; i < values.length; i++) {
241         for (WebElement option : options) {
242             if (option.getAttribute("value").equals(values[i])) {
243                 setSelected(option);
244                 break;
245             }
246         }
247     }
248 }
249 
250 private void setSelected(WebElement option) {
251     if (!option.isSelected())
252         option.click();
253     try {
254         Thread.sleep(1000);
255     } catch (InterruptedException e) {
256         e.printStackTrace();
257     }
258 }
259 }
260 }
```

The screenshot shows the Java code for the `Table` class within the `com.ebay.maui.driver.web.element` package. The code implements the `HtmlElement` interface and provides methods for finding rows and columns, and getting cell content.

```
8 public class Table extends HtmlElement {
9     private List<WebElement> rows = null;
10    private List<WebElement> columns = null;
11 
12    public Table(String label, By by) {
13        super(label, by);
14    }
15 
16    public Table(String label, String locator) {
17        super(label, locator);
18    }
19 
20    public void findElement() {
21        super.findElement();
22        try {
23            rows = element.findElements(By.tagName("tr"));
24        } catch (NotFoundException e) {
25        }
26    }
27 
28    public int getColumnCount() {
29        if (columns == null)
30            findElement();
31 
32        // Need to check whether rows is null AND whether or not the list of rows is empty
33        if (rows != null && !rows.isEmpty()) {
34            try {
35                columns = rows.get(0).findElements(By.tagName("td"));
36            } catch (NotFoundException e) {
37            }
38        }
39        if (columns == null || columns.size() == 0) {
40            try {
41                if (rows.size() > 1)
42                    columns = rows.get(1).findElements(By.tagName("td"));
43                else
44                    columns = rows.get(0).findElements(By.tagName("th"));
45            } catch (NotFoundException e) {
46            }
47        }
48    }
49 
50    if (columns != null)
51        return columns.size();
52    return 0;
53 }
54 
55 public List<WebElement> getColumns() {
56     return columns;
57 }
58 
59 /**
60  * Get table cell content
61  * @param row Starts from 1
62  * @param column Starts from 1
63  */
64 public String getContent(int row, int column) {
65     if (rows == null)
66         findElement();
67     if (rows != null && !rows.isEmpty()) {
68         try {
69             WebElement rowElement = rows.get(row - 1);
70             if (rowElement != null && !rowElement.isEmpty())
71                 try {
72                     WebElement columnElement = rowElement.findElements(By.tagName("td"))
73                         .get(column - 1);
74                     if (columnElement != null)
75                         return columnElement.getText();
76                 } catch (Exception e) {
77                     e.printStackTrace();
78                 }
79             }
80         } catch (Exception e) {
81             e.printStackTrace();
82         }
83     }
84     return null;
85 }
```

```

61 /**
62  * Get table cell content
63  * @param row Starts from 1
64  * @param column Starts from 1
65  */
66 public String getContent(int row, int column) {
67     if (rows == null)
68         findElement();
69
70     if (rows != null && !rows.isEmpty()) {
71         try {
72             columns = rows.get(row - 1).findElements(By.tagName("td"));
73         } catch (NotFoundException e) {
74             e.printStackTrace();
75         }
76         if (columns == null || columns.size() == 0) {
77             try {
78                 columns = rows.get(row - 1).findElements(By.tagName("th"));
79             } catch (NotFoundException e) {
80                 e.printStackTrace();
81             }
82         }
83         return columns.get(column - 1).getText();
84     }
85     return null;
86 }
87
88 // TODO MM: In this method we're looking for /tbody/tr OR /tr -- this differs from how the rows property is being set
89 // in the constructor and the findElement() method. Why do we call findElements() immediately after the
90 // findElement(), which basically does the same thing? If we expect the answer to be different when we call /tbody/tr,
91 // that means we don't trust this.findElement(), which means the getCount might be different from rows.size(). That could
92 // lead to some very tricky debugging
93 public int getRowCount() {
94     if (rows == null)
95         findElement();
96     else
97         return rows.size();
98     int count = element.findElements(By.xpath("//tbody/tr")).size();
99     if (count == 0) {
100         count = element.findElements(By.xpath("//tr")).size();
101     }
102     return count;
103 }
104
105 public List<WebElement> getRows() {
106     return rows;
107 }
108
109 public boolean isHasBody() {
110     return getRows().size() > 0;
111 }
112
113 // public void setColumns(List<WebElement> columns) {
114 //     this.columns = columns;
115 // }
116
117 // public void setHasBody(boolean hasBody) {
118 //     this.hasBody = hasBody;
119 // }
120
121 // public void setRows(List<WebElement> rows) {
122 //     this.rows = rows;
123 // }
124 }
125

```

```

1 package com.ebay.maul.driver.web.element;
2
3 import org.openqa.selenium.By;
4 import org.openqa.selenium.WebDriverException;
5
6 import com.ebay.maul.controller.Logging;
7
8 public class TextField extends HtmlElement {
9     public TextField(String label, By by) {
10         super(label, by);
11     }
12
13     public TextField(String label, String locator) {
14         super(label, locator);
15     }
16
17     public void clear() {
18         Logging.logWebStep(null, "clear value on " + toHTML(), false);
19         findElement();
20         // clear for file input, will get Exception:Element must be
21         // user-editable in order to clear it.
22         if (!element.getAttribute("type").equals(ignoreCase("file")))
23             element.clear();
24     }
25
26     public void sendKeys(String keysToSend) {
27         Logging.logWebStep(null, "sendKeys '" + keysToSend + "' on " +
28                             toHTML(), false);
29         findElement();
30         element.sendKeys(keysToSend);
31     }
32
33     public void type(String keysToSend) {
34         try {
35             clear();
36         } catch (WebDriverException e) {
37             Logging.logWebStep(null, "Exception:clear value on " + toHTML(),
38                               true);
39         }
40         sendKeys(keysToSend);
41     }
42 }
43

```

The screenshot shows the Java code for the `WebPage` class in an IDE. The code is annotated with Javadoc-style comments and annotations like `@SuppressWarnings("deprecation")`, `@Param`, and `@throws`. The class extends `BaseUIPage` and implements `IPage`. It contains methods for handling page redirection, opening new windows, and managing URLs. The code is heavily commented and includes several imports at the top.

```
/* SuppressWarnings("deprecation") */
public class WebPage extends BaseUIPage implements IPage {
    ...
    private static final Logger logger = Logger.getLogger(WebPage.class);
    private static final int MAX_WAIT_TIME_FOR_REDIRECTION = 3;
    private boolean popupFlag = false;
    private String url;
    private HTMLElement pageIdentifierElement = null;
    private String popupWindowName = null;
    private String windowHandle = null;
    private String pageTitle = null;
    private String location = null;
    private String htmlSavedPath = null;
    private String htmlFilePath = null;
    private String rId = null;
    private String pageName = null;
    private String testMethodSignature = null;
    private String testMethodSignaturePath = null;
    private String imageFilePath = null;
    ...
    /**
     * Constructor for non-entry point page
     */
    @Param("pageIdentifierElement")
    @throws Exception
    public WebPage() throws Exception {
        this(false, null, null, null, true, false);
    }
    ...
    public WebPage(boolean isPopup, String popupWindowName) throws Exception {
        this(isPopup, popupWindowName, null, null, true, false);
    }
    ...
    /**
     * Base Constructor
     */
    @Param("url")
    @Param("convert")
    @throws Exception
    private WebPage(boolean isPopup, HTMLElement pageIdentifierElement, String url, boolean convert, booleanwaitForRedirection) throws Exception {
        Calendar start = Calendar.getInstance();
        start.setTime(new Date());
        ...
        if (ContextManager.getGlobalContext() != null) {
            ContextManager.getGlobalContext().getTestingContext();
            suiteName = ContextManager.getGlobalContext().getTestingContext()
                .getSuite().getName();
            outputDirectory = ContextManager.getGlobalContext()
                .getTestingContext().getOutputDirectory();
            testMethodSignature = ContextManager.getGlobalContext().getTestMethodSignature();
            // testMethodSignature = ContextManager.getThreadContext().getTestMethodSignature();
        }
        ...
        this.pageIdentifierElement = pageIdentifierElement;
        this.popupFlag = isPopup;
        this.popupWindowName = popupWindowName;
    }
    ...
}
```

The screenshot shows the `mauccomponent-ebayindia/mauccomponent-ebayindia` code in an IDE. The code is annotated with Javadoc-style comments and annotations like `@Param` and `@throws`. It includes a `driver` field and various methods for handling web pages, such as `open`, `waitForPageLoad`, and `selectWindow`. The code also handles redirection logic and waits for specific conditions to be met. The code is heavily commented and includes several imports at the top.

```
driver = WebUDriver.getWebDriver();
...
if (!popupFlag) {
    ...
    if (url != null) {
        open(url, convert);
    }
    ...
    // Wait For Page Load
    waitForPageToLoad();
}
else {
    Logging.logWarning(null, "wait for popup page to load.", false);
    waitForPopup(popupWindowName);
}
Windows windows = new Windows(driver);
windows.selectWindow(driver, "name=" + popupWindowName);
...
// populate page info
populateAndCapturePageSnapshot();
}
try {
    this.windowHandle = driver.getWindowHandle();
} catch (Exception ex) {
    // Ignore for OperoDriver
}
...
// Take care of redirecting pages like SignIn
int count = 0;
boolean pageNotCurrent = true;
do {
    count++;
    try {
        // Check for Page errors
        assertNoPageErrors();
        pageNotCurrent = false;
    } catch (PageNotCurrentException e) {
        if (!waitForRedirection)
            throw e;
        else
            waitForSeconds(1);
        waitForPageLoad();
    }
} while (waitForRedirection && pageNotCurrent
    && count < MAX_WAIT_TIME_FOR_REDIRECTION);
AbstractPageListener.notifyPageLoad(this);
Calendar end = Calendar.getInstance();
start.setTime(new Date());
long startTime = start.getTimeInMillis();
long endTime = end.getTimeInMillis();
if ((endTime - startTime) / 1000 > 0)
    Logging.log("Open web page in :" + (endTime - startTime) / 1000
    + "seconds",true);
}
...
/**
 * Constructor for non-entry point page
 */
@Param("pageIdentifierElement")
@throws Exception
*/
```

Screenshot of two Eclipse IDE windows showing Java code for the `WebPage` class.

**Top Window:**

```

176    public WebPage(HtmlElement pageIdentifierElement) throws Exception {
177        this(false, null, pageIdentifierElement, null, true, false);
178    }
179
180    /**
181     * Constructor for non-entry point page after redirection
182     *
183     * @param pageIdentifierElement
184     * @param urlForRedirection
185     * @throws Exception
186     */
187    public WebPage(HtmlElement pageIdentifierElement, boolean waitForRedirection)
188        throws Exception {
189        this(false, null, pageIdentifierElement, null, true, waitForRedirection);
190    }
191
192    /**
193     * Constructor for Pop-up Pages
194     *
195     * @param pageIdentifierElement
196     * @param isPopup
197     * @param popupWindowName
198     * @throws Exception
199     */
200    public WebPage(HtmlElement pageIdentifierElement, boolean isPopup,
201        String popupWindowName) throws Exception {
202        this(isPopup, popupWindowName, pageIdentifierElement, null, true, false);
203    }
204
205    /**
206     * Constructor for Entry point page. URLs will be converted based on feature
207     * pool.
208     *
209     * @param pageIdentifierElement
210     * @param url
211     * @throws Exception
212     */
213    public WebPage(HtmlElement pageIdentifierElement, String url)
214        throws Exception {
215        this(false, null, pageIdentifierElement, url, true, false);
216    }
217
218    /**
219     * Constructor for entry point page after redirection
220     *
221     * @param pageIdentifierElement
222     * @param url
223     * @param waitForRedirection
224     * @throws Exception
225     */
226    public WebPage(HtmlElement pageIdentifierElement, String url,
227        boolean waitForRedirection) throws Exception {
228        this(false, null, pageIdentifierElement, url, true, waitForRedirection);
229    }
230
231    // ////////////////// Getters & Setters End
232    // /////////////////////////////////
233
234    /**
235     * Constructor for Entry point page.
236     *
237     * @pre>
238     * WARNING!!! WARNING!!! WARNING!!!
239     * Should only be used in following cases
240     */
241    public WebPage(String url) throws Exception {
242        this(false, null, null, url, true, false);
243    }
244
245    /**
246     * Should only be used in following cases
247     */
248    public WebPage(HtmlElement pageIdentifierElement, String url,
249        boolean isAbsoluteURL, boolean waitForRedirection)
250        throws Exception {
251        this(false, null, pageIdentifierElement, url, isAbsoluteURL,
252            waitForRedirection);
253    }
254
255    /**
256     * Constructor for Entry point page. Null PageIdentifier Element, always
257     * convert
258     *
259     * @param url
260     * @throws Exception
261     */
262    public WebPage(String url) throws Exception {
263        this(false, null, null, url, true, false);
264    }
265
266    /**
267     * Constructor for Entry point page. Null PageIdentifier Element, not
268     * convert if convert=false
269     *
270     * @param url
271     * @param convert
272     * @throws Exception
273     */
274    public WebPage(String url, boolean convert) throws Exception {
275        this(false, null, null, url, convert, false);
276    }
277
278    /**
279     * assertCookiePresent(String name)
280     */
281    protected void assertCookiePresent(String name) {
282        Logging.logWebStep(null, "assert cookie " + name + " is present.");
283        assertFalse(getCookieByName(name) != null, "Cookie: {" + name
284            + " not found.");
285    }
286
287    @Override
288    protected void assertCurrentPage(boolean log)
289        throws PageNotCurrentException {
290
291        //if (pageIdentifierElement != null){
292        //    try {
293        //        pageIdentifierElement.init();
294        //    } catch (Exception e) {
295        //        e.printStackTrace();
296        //    }
297        //}
298
299        if (pageIdentifierElement == null) {
300
301        } else if (this.isElementPresent(pageIdentifierElement.getBy())) {
302            try {
303                ...
304            }
305        }
306    }

```

**Bottom Window:**

```

239    /**
240     * assertCookiePresent(String name)
241     */
242    protected void assertCookiePresent(String name) {
243        Logging.logWebStep(null, "assert cookie " + name + " is present.");
244        assertFalse(getCookieByName(name) != null, "Cookie: {" + name
245            + " not found.");
246    }
247
248    @Override
249    protected void assertCurrentPage(boolean log)
250        throws PageNotCurrentException {
251
252        //if (pageIdentifierElement != null){
253        //    try {
254        //        pageIdentifierElement.init();
255        //    } catch (Exception e) {
256        //        e.printStackTrace();
257        //}
258
259        if (pageIdentifierElement == null) {
260
261        } else if (this.isElementPresent(pageIdentifierElement.getBy())) {
262            try {
263                ...
264            }
265        }
266    }

```

The screenshot shows an IDE interface with a code editor displaying Java code. The code is part of a class named `WebPage`. The code includes several methods for asserting page elements and their presence. It uses annotations like `@Ignore` and `@Test`, and imports from packages such as `com.ebay.maui.driver.web.element` and `com.ebay.maui.platform`. The code is annotated with Javadoc-style comments explaining its purpose.

```
...  
363     try {  
364         if (!contextHelper.getThreadContext().isCaptureSnapshot())  
365             new ScreenShotUtil(driver).capturePageSnapshotOnException();  
366     } catch (Exception e) {  
367         // ignore all exceptions  
368         e.printStackTrace();  
369     }  
370     throw new PageNotCurrentException(getClass().getCanonicalName()  
371             + " is not the current page. @pageIdentifierElement "  
372             + pageIdentifierElement.toString() + " is not found.");  
373 }  
374  
375 if (log)  
376     Logging.logWebStep(  
377         null,  
378         "assert " +  
379             + getClass().getSimpleName()  
380             + " is the current page."  
381             + "(pageIdentifierElement == null ? \" assert PageIdentifierElement "  
382             + pageIdentifierElement.toHTML()  
383             + " is present.\")"  
384             + ":"), false);  
385 }  
386  
387 // ////////////////// assertion methods ///////////////////  
388  
389 public void assertHtmlSource(String text) {  
390     Logging.logWebStep(null, "assert text '" + text  
391             + "' is present in page source.", false);  
392     assertHTML(getHTMLSource().contains(text)), "Text: (" + text  
393             + ") not found on page source.");  
394 }  
395  
396 public void assertKeywordInPagePresent(String text) {  
397     Logging.logWebStep(null, "assert text '" + text  
398             + "' is present in page source.", false);  
399     Assert.assertFalse(getHTMLSource().contains(text)), "Text: (" + text  
400             + ") not found on page source.");  
401 }  
402  
403 public void assertLocation(String urlPattern) {  
404     Logging.logWebStep(null, "assert location '" + urlPattern + "\\".  
405             false);  
406     assertHTML(getLocation().contains(urlPattern)), "Pattern: ("  
407             + urlPattern + ") not found on page location.");  
408 }  
409  
410 /**  
411 * Check HTTP Status Code Errors. This is going to be tricky. Selenium does not  
412 * provide us with HTTP Response codes. So have to rely on the DOM content  
413 * to check for Errors. For example: QA PHR Page If page error, take error  
414 * snapshot and throw exception  
415 * @throws PageNotCurrentException  
416 */  
417 public final void assertNoPageErrors() throws PageNotCurrentException {  
418     for (HTTPStatusCode statusCode : HTTPStatusCode.values()) {  
419         if (getBodyText() != null  
420             && getBodyText().toLowerCase().contains  
421                 statusCode.getTitle().toLowerCase()) {  
422             throw new WebResponseException(statusCode.toString()  
423                     + " Found.", null, this);  
424         }  
425     }  
426 }
```

This screenshot continues the same IDE session, showing the same Java code for the `WebPage` class. The code includes methods for capturing screenshots, handling file paths, and closing the browser. It also contains logic for dealing with multiple windows and ignoring specific exceptions.

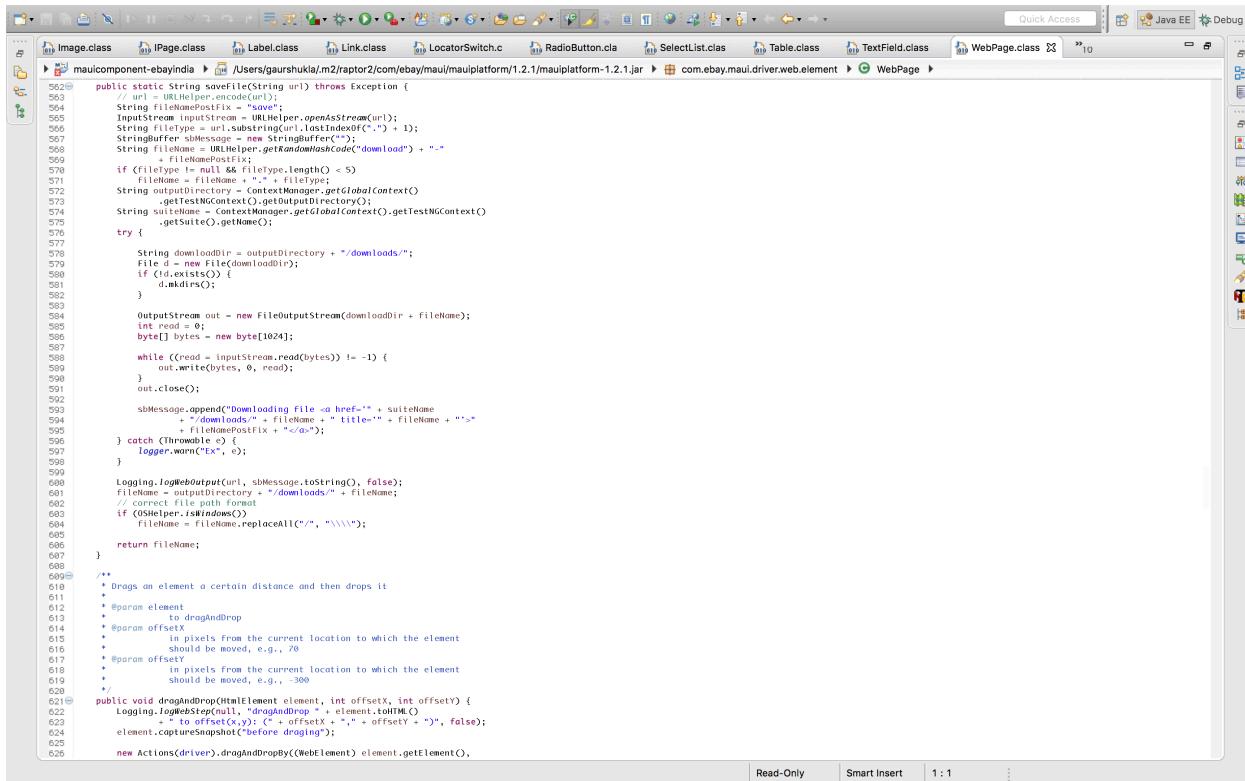
```
...  
378     // Check whether it's the expected page.  
379     assertCurrentPage(true);  
380 }  
381  
382 public void assertPageSectionPresent(WebPageSection pageSection) {  
383     Logging.logWebStep(null,  
384         "assert pageSection '" + pageSection.getName()  
385             + "' is present.", false);  
386     assertElementPresent(new HTMLElement(pageSection.getName()),  
387             pageSection.getBy());  
388 }  
389  
390 public void assertTitleString(String text) {  
391     Logging.logWebStep(null, "assert text '" + text  
392             + "' is present in title.", false);  
393     assertHTML(getTitle().contains(text)), "Text: (" + text  
394             + ") not found on page title.");  
395 }  
396  
397 public void capturePageSnapshot() {  
398     ScreenShot screenShot = new ScreenShotUtil(driver)  
399         .captureWebPageSnapshot();  
400     this.title = screenShot.getTitle();  
401     this.pageId = screenShot.getPageId();  
402     this.pageId = screenShot.getRegId();  
403  
404     if (screenShot.getHTMLSourcePath() == null) {  
405         htmlFilePath = page.getHTMLSourcePath().replace(suiteName,  
406             outputDirectory);  
407         htmlSavedFilePath = screenShot.getHTMLSourcePath();  
408     }  
409     if (screenShot.getImagePath() != null) {  
410         imagePath = screenShot.getImagePath().replace(suiteName,  
411             outputDirectory);  
412     }  
413     Logging.logWebOutput(null,  
414         title + " (" + Logging.buildScreenshotLog(screenShot) + ")",  
415         false);  
416 }  
417  
418 public final void close() throws PageNotCurrentException {  
419     if (WebElementDriver.getWebDriver() == null) {  
420         return;  
421     }  
422     Alert alertListener = AlertPageUpload.this;  
423     Logging.log("close web page");  
424     boolean isMultipleWindow = false;  
425     if (driver.getWindowHandles().size() > 1)  
426         isMultipleWindow = true;  
427     try {  
428         driver.close();  
429     } catch (WebDriverException ignore) {}  
430     if (WebElementDriver.getWebDriver().getMode()  
431         .equals(ignoreClose("LocallyOnMC"))){  
432         try {  
433             Thread.sleep(2000 * 2);  
434         } catch (InterruptedException e) {}  
435     } // ignore  
436 }  
437  
438 try {  
439     if (isMultipleWindow)  
440         ...  
441     ...  
442 }  
443 ...  
444 ...  
445 ...  
446 ...  
447 ...  
448 ...  
449 ...  
450 ...  
451 ...  
452 ...  
453 ...  
454 ...  
455 ...  
456 ...  
457 ...  
458 ...  
459 ...  
460 ...  
461 ...  
462 ...  
463 ...  
464 ...  
465 ...  
466 ...  
467 ...  
468 ...  
469 ...  
470 ...  
471 ...  
472 ...  
473 ...  
474 ...  
475 ...  
476 ...  
477 ...  
478 ...  
479 ...  
480 ...  
481 ...  
482 ...  
483 ...  
484 ...  
485 ...  
486 ...  
487 ...  
488 ...  
489 ...  
490 ...  
491 ...  
492 ...  
493 ...  
494 ...  
495 ...  
496 ...  
497 ...  
498 ...  
499 ...  
500 ...  
501 ...  
502 ...  
503 ...  
504 ...  
505 ...  
506 ...  
507 ...  
508 ...  
509 ...  
510 ...  
511 ...  
512 ...  
513 ...  
514 ...  
515 ...  
516 ...  
517 ...  
518 ...  
519 ...  
520 ...  
521 ...  
522 ...  
523 ...  
524 ...  
525 ...  
526 ...  
527 ...  
528 ...  
529 ...  
530 ...  
531 ...  
532 ...  
533 ...  
534 ...  
535 ...  
536 ...  
537 ...  
538 ...  
539 ...  
540 ...  
541 ...  
542 ...  
543 ...  
544 ...  
545 ...  
546 ...  
547 ...  
548 ...  
549 ...  
550 ...  
551 ...  
552 ...  
553 ...  
554 ...  
555 ...  
556 ...  
557 ...  
558 ...  
559 ...  
560 ...  
561 ...  
562 ...  
563 ...  
564 ...  
565 ...  
566 ...  
567 ...  
568 ...  
569 ...  
570 ...  
571 ...  
572 ...  
573 ...  
574 ...  
575 ...  
576 ...  
577 ...  
578 ...  
579 ...  
580 ...  
581 ...  
582 ...  
583 ...  
584 ...  
585 ...  
586 ...  
587 ...  
588 ...  
589 ...  
590 ...  
591 ...  
592 ...  
593 ...  
594 ...  
595 ...  
596 ...  
597 ...  
598 ...  
599 ...  
600 ...  
601 ...  
602 ...  
603 ...  
604 ...  
605 ...  
606 ...  
607 ...  
608 ...  
609 ...  
610 ...  
611 ...  
612 ...  
613 ...  
614 ...  
615 ...  
616 ...  
617 ...  
618 ...  
619 ...  
620 ...  
621 ...  
622 ...  
623 ...  
624 ...  
625 ...  
626 ...  
627 ...  
628 ...  
629 ...  
630 ...  
631 ...  
632 ...  
633 ...  
634 ...  
635 ...  
636 ...  
637 ...  
638 ...  
639 ...  
640 ...  
641 ...  
642 ...  
643 ...  
644 ...  
645 ...  
646 ...  
647 ...  
648 ...  
649 ...  
650 ...  
651 ...  
652 ...  
653 ...  
654 ...  
655 ...  
656 ...  
657 ...  
658 ...  
659 ...  
660 ...  
661 ...  
662 ...  
663 ...  
664 ...  
665 ...  
666 ...  
667 ...  
668 ...  
669 ...  
670 ...  
671 ...  
672 ...  
673 ...  
674 ...  
675 ...  
676 ...  
677 ...  
678 ...  
679 ...  
680 ...  
681 ...  
682 ...  
683 ...  
684 ...  
685 ...  
686 ...  
687 ...  
688 ...  
689 ...  
690 ...  
691 ...  
692 ...  
693 ...  
694 ...  
695 ...  
696 ...  
697 ...  
698 ...  
699 ...  
700 ...  
701 ...  
702 ...  
703 ...  
704 ...  
705 ...  
706 ...  
707 ...  
708 ...  
709 ...  
710 ...  
711 ...  
712 ...  
713 ...  
714 ...  
715 ...  
716 ...  
717 ...  
718 ...  
719 ...  
720 ...  
721 ...  
722 ...  
723 ...  
724 ...  
725 ...  
726 ...  
727 ...  
728 ...  
729 ...  
730 ...  
731 ...  
732 ...  
733 ...  
734 ...  
735 ...  
736 ...  
737 ...  
738 ...  
739 ...  
740 ...  
741 ...  
742 ...  
743 ...  
744 ...  
745 ...  
746 ...  
747 ...  
748 ...  
749 ...  
750 ...  
751 ...  
752 ...  
753 ...  
754 ...  
755 ...  
756 ...  
757 ...  
758 ...  
759 ...  
760 ...  
761 ...  
762 ...  
763 ...  
764 ...  
765 ...  
766 ...  
767 ...  
768 ...  
769 ...  
770 ...  
771 ...  
772 ...  
773 ...  
774 ...  
775 ...  
776 ...  
777 ...  
778 ...  
779 ...  
780 ...  
781 ...  
782 ...  
783 ...  
784 ...  
785 ...  
786 ...  
787 ...  
788 ...  
789 ...  
790 ...  
791 ...  
792 ...  
793 ...  
794 ...  
795 ...  
796 ...  
797 ...  
798 ...  
799 ...  
800 ...  
801 ...  
802 ...  
803 ...  
804 ...  
805 ...  
806 ...  
807 ...  
808 ...  
809 ...  
810 ...  
811 ...  
812 ...  
813 ...  
814 ...  
815 ...  
816 ...  
817 ...  
818 ...  
819 ...  
820 ...  
821 ...  
822 ...  
823 ...  
824 ...  
825 ...  
826 ...  
827 ...  
828 ...  
829 ...  
830 ...  
831 ...  
832 ...  
833 ...  
834 ...  
835 ...  
836 ...  
837 ...  
838 ...  
839 ...  
840 ...  
841 ...  
842 ...  
843 ...  
844 ...  
845 ...  
846 ...  
847 ...  
848 ...  
849 ...  
850 ...  
851 ...  
852 ...  
853 ...  
854 ...  
855 ...  
856 ...  
857 ...  
858 ...  
859 ...  
860 ...  
861 ...  
862 ...  
863 ...  
864 ...  
865 ...  
866 ...  
867 ...  
868 ...  
869 ...  
870 ...  
871 ...  
872 ...  
873 ...  
874 ...  
875 ...  
876 ...  
877 ...  
878 ...  
879 ...  
880 ...  
881 ...  
882 ...  
883 ...  
884 ...  
885 ...  
886 ...  
887 ...  
888 ...  
889 ...  
890 ...  
891 ...  
892 ...  
893 ...  
894 ...  
895 ...  
896 ...  
897 ...  
898 ...  
899 ...  
900 ...  
901 ...  
902 ...  
903 ...  
904 ...  
905 ...  
906 ...  
907 ...  
908 ...  
909 ...  
910 ...  
911 ...  
912 ...  
913 ...  
914 ...  
915 ...  
916 ...  
917 ...  
918 ...  
919 ...  
920 ...  
921 ...  
922 ...  
923 ...  
924 ...  
925 ...  
926 ...  
927 ...  
928 ...  
929 ...  
930 ...  
931 ...  
932 ...  
933 ...  
934 ...  
935 ...  
936 ...  
937 ...  
938 ...  
939 ...  
940 ...  
941 ...  
942 ...  
943 ...  
944 ...  
945 ...  
946 ...  
947 ...  
948 ...  
949 ...  
950 ...  
951 ...  
952 ...  
953 ...  
954 ...  
955 ...  
956 ...  
957 ...  
958 ...  
959 ...  
960 ...  
961 ...  
962 ...  
963 ...  
964 ...  
965 ...  
966 ...  
967 ...  
968 ...  
969 ...  
970 ...  
971 ...  
972 ...  
973 ...  
974 ...  
975 ...  
976 ...  
977 ...  
978 ...  
979 ...  
980 ...  
981 ...  
982 ...  
983 ...  
984 ...  
985 ...  
986 ...  
987 ...  
988 ...  
989 ...  
990 ...  
991 ...  
992 ...  
993 ...  
994 ...  
995 ...  
996 ...  
997 ...  
998 ...  
999 ...  
1000 ...  
1001 ...  
1002 ...  
1003 ...  
1004 ...  
1005 ...  
1006 ...  
1007 ...  
1008 ...  
1009 ...  
1010 ...  
1011 ...  
1012 ...  
1013 ...  
1014 ...  
1015 ...  
1016 ...  
1017 ...  
1018 ...  
1019 ...  
1020 ...  
1021 ...  
1022 ...  
1023 ...  
1024 ...  
1025 ...  
1026 ...  
1027 ...  
1028 ...  
1029 ...  
1030 ...  
1031 ...  
1032 ...  
1033 ...  
1034 ...  
1035 ...  
1036 ...  
1037 ...  
1038 ...  
1039 ...  
1040 ...  
1041 ...  
1042 ...  
1043 ...  
1044 ...  
1045 ...  
1046 ...  
1047 ...  
1048 ...  
1049 ...  
1050 ...  
1051 ...  
1052 ...  
1053 ...  
1054 ...  
1055 ...  
1056 ...  
1057 ...  
1058 ...  
1059 ...  
1060 ...  
1061 ...  
1062 ...  
1063 ...  
1064 ...  
1065 ...  
1066 ...  
1067 ...  
1068 ...  
1069 ...  
1070 ...  
1071 ...  
1072 ...  
1073 ...  
1074 ...  
1075 ...  
1076 ...  
1077 ...  
1078 ...  
1079 ...  
1080 ...  
1081 ...  
1082 ...  
1083 ...  
1084 ...  
1085 ...  
1086 ...  
1087 ...  
1088 ...  
1089 ...  
1090 ...  
1091 ...  
1092 ...  
1093 ...  
1094 ...  
1095 ...  
1096 ...  
1097 ...  
1098 ...  
1099 ...  
1100 ...  
1101 ...  
1102 ...  
1103 ...  
1104 ...  
1105 ...  
1106 ...  
1107 ...  
1108 ...  
1109 ...  
1110 ...  
1111 ...  
1112 ...  
1113 ...  
1114 ...  
1115 ...  
1116 ...  
1117 ...  
1118 ...  
1119 ...  
1120 ...  
1121 ...  
1122 ...  
1123 ...  
1124 ...  
1125 ...  
1126 ...  
1127 ...  
1128 ...  
1129 ...  
1130 ...  
1131 ...  
1132 ...  
1133 ...  
1134 ...  
1135 ...  
1136 ...  
1137 ...  
1138 ...  
1139 ...  
1140 ...  
1141 ...  
1142 ...  
1143 ...  
1144 ...  
1145 ...  
1146 ...  
1147 ...  
1148 ...  
1149 ...  
1150 ...  
1151 ...  
1152 ...  
1153 ...  
1154 ...  
1155 ...  
1156 ...  
1157 ...  
1158 ...  
1159 ...  
1160 ...  
1161 ...  
1162 ...  
1163 ...  
1164 ...  
1165 ...  
1166 ...  
1167 ...  
1168 ...  
1169 ...  
1170 ...  
1171 ...  
1172 ...  
1173 ...  
1174 ...  
1175 ...  
1176 ...  
1177 ...  
1178 ...  
1179 ...  
1180 ...  
1181 ...  
1182 ...  
1183 ...  
1184 ...  
1185 ...  
1186 ...  
1187 ...  
1188 ...  
1189 ...  
1190 ...  
1191 ...  
1192 ...  
1193 ...  
1194 ...  
1195 ...  
1196 ...  
1197 ...  
1198 ...  
1199 ...  
1200 ...  
1201 ...  
1202 ...  
1203 ...  
1204 ...  
1205 ...  
1206 ...  
1207 ...  
1208 ...  
1209 ...  
1210 ...  
1211 ...  
1212 ...  
1213 ...  
1214 ...  
1215 ...  
1216 ...  
1217 ...  
1218 ...  
1219 ...  
1220 ...  
1221 ...  
1222 ...  
1223 ...  
1224 ...  
1225 ...  
1226 ...  
1227 ...  
1228 ...  
1229 ...  
1230 ...  
1231 ...  
1232 ...  
1233 ...  
1234 ...  
1235 ...  
1236 ...  
1237 ...  
1238 ...  
1239 ...  
1240 ...  
1241 ...  
1242 ...  
1243 ...  
1244 ...  
1245 ...  
1246 ...  
1247 ...  
1248 ...  
1249 ...  
1250 ...  
1251 ...  
1252 ...  
1253 ...  
1254 ...  
1255 ...  
1256 ...  
1257 ...  
1258 ...  
1259 ...  
1260 ...  
1261 ...  
1262 ...  
1263 ...  
1264 ...  
1265 ...  
1266 ...  
1267 ...  
1268 ...  
1269 ...  
1270 ...  
1271 ...  
1272 ...  
1273 ...  
1274 ...  
1275 ...  
1276 ...  
1277 ...  
1278 ...  
1279 ...  
1280 ...  
1281 ...  
1282 ...  
1283 ...  
1284 ...  
1285 ...  
1286 ...  
1287 ...  
1288 ...  
1289 ...  
1290 ...  
1291 ...  
1292 ...  
1293 ...  
1294 ...  
1295 ...  
1296 ...  
1297 ...  
1298 ...  
1299 ...  
1300 ...  
1301 ...  
1302 ...  
1303 ...  
1304 ...  
1305 ...  
1306 ...  
1307 ...  
1308 ...  
1309 ...  
1310 ...  
1311 ...  
1312 ...  
1313 ...  
1314 ...  
1315 ...  
1316 ...  
1317 ...  
1318 ...  
1319 ...  
1320 ...  
1321 ...  
1322 ...  
1323 ...  
1324 ...  
1325 ...  
1326 ...  
1327 ...  
1328 ...  
1329 ...  
1330 ...  
1331 ...  
1332 ...  
1333 ...  
1334 ...  
1335 ...  
1336 ...  
1337 ...  
1338 ...  
1339 ...  
1340 ...  
1341 ...  
1342 ...  
1343 ...  
1344 ...  
1345 ...  
1346 ...  
1347 ...  
1348 ...  
1349 ...  
1350 ...  
1351 ...  
1352 ...  
1353 ...  
1354 ...  
1355 ...  
1356 ...  
1357 ...  
1358 ...  
1359 ...  
1360 ...  
1361 ...  
1362 ...  
1363 ...  
1364 ...  
1365 ...  
1366 ...  
1367 ...  
1368 ...  
1369 ...  
1370 ...  
1371 ...  
1372 ...  
1373 ...  
1374 ...  
1375 ...  
1376 ...  
1377 ...  
1378 ...  
1379 ...  
1380 ...  
1381 ...  
1382 ...  
1383 ...  
1384 ...  
1385 ...  
1386 ...  
1387 ...  
1388 ...  
1389 ...  
1390 ...  
1391 ...  
1392 ...  
1393 ...  
1394 ...  
1395 ...  
1396 ...  
1397 ...  
1398 ...  
1399 ...  
1400 ...  
1401 ...  
1402 ...  
1403 ...  
1404 ...  
1405 ...  
1406 ...  
1407 ...  
1408 ...  
1409 ...  
1410 ...  
1411 ...  
1412 ...  
1413 ...  
1414 ...  
1415 ...  
1416 ...  
1417 ...  
1418 ...  
1419 ...  
1420 ...  
1421 ...  
1422 ...  
1423 ...  
1424 ...  
1425 ...  
1426 ...  
1427 ...  
1428 ...  
1429 ...  
1430 ...  
1431 ...  
1432 ...  
1433 ...  
1434 ...  
1435 ...  
1436 ...  
1437 ...  
1438 ...  
1439 ...  
1440 ...  
1441 ...  
1442 ...  
1443 ...  
1444 ...  
1445 ...  
1446 ...  
1447 ...  
1448 ...  
1449 ...  
1450 ...  
1451 ...  
1452 ...  
1453 ...  
1454 ...  
1455 ...  
1456 ...  
1457 ...  
1458 ...  
1459 ...  
1460 ...  
1461 ...  
1462 ...  
1463 ...  
1464 ...  
1465 ...  
1466 ...  
1467 ...  
1468 ...  
1469 ...  
1470 ...  
1471 ...  
1472 ...  
1473 ...  
1474 ...  
1475 ...  
1476 ...  
1477 ...  
1478 ...  
1479 ...  
1480 ...  
1481 ...  
1482 ...  
1483 ...  
1484 ...  
1485 ...  
1486 ...  
1487 ...  
1488 ...  
1489 ...  
1490 ...  
1491 ...  
1492 ...  
1493 ...  
1494 ...  
1495 ...  
1496 ...  
1497 ...  
1498 ...  
1499 ...  
1500 ...  
1501 ...  
1502 ...  
1503 ...  
1504 ...  
1505 ...  
1506 ...  
1507 ...  
1508 ...  
1509 ...  
1510 ...  
1511 ...  
1512 ...  
1513 ...  
1514 ...  
1515 ...  
1516 ...  
1517 ...  
1518 ...  
1519 ...  
1520 ...  
1521 ...  
1522 ...  
1523 ...  
1524 ...  
1525 ...  
1526 ...  
1527 ...  
1528 ...  
1529 ...  
1530 ...  
1531 ...  
1532 ...  
1533 ...  
1534 ...  
1535 ...  
1536 ...  
1537 ...  
1538 ...  
1539 ...  
1540 ...  
1541 ...  
1542 ...  
1543 ...  
1544 ...  
1545 ...  
1546 ...  
1547 ...  
1548 ...  
1549 ...  
1550 ...  
1551 ...  
1552 ...  
1553 ...  
1554 ...  
1555 ...  
1556 ...  
1557 ...  
1558 ...  
1559 ...  
1560 ...  
1561 ...  
1562 ...  
1563 ...  
1564 ...  
1565 ...  
1566 ...  
1567 ...  
1568 ...  
1569 ...  
1570 ...  
1571 ...  
1572 ...  
1573 ...  
1574 ...  
1575 ...  
1576 ...  
1577 ...  
1578 ...  
1579 ...  
1580 ...  
1581 ...  
1582 ...  
1583 ...  
1584 ...  
1585 ...  
1586 ...  
1587 ...  
1588 ...  
1589 ...  
1590 ...  
1591 ...  
1592 ...  
1593 ...  
1594 ...  
1595 ...  
1596 ...  
1597 ...  
1598 ...  
1599 ...  
1600 ...  
1601 ...  
1602 ...  
1603 ...  
1604 ...  
1605 ...  
1606 ...  
1607 ...  
1608 ...  
1609 ...  
1610 ...  
1611 ...  
1612 ...  
1613 ...  
1614 ...  
1615 ...  
1616 ...  
1617 ...  
1618 ...  
1619 ...  
1620 ...  
1621 ...  
1622 ...  
1623 ...  
1624 ...  
1625 ...  
1626 ...  
1627 ...  
1628 ...  
1629 ...  
1630 ...  
1631 ...  
1632 ...  
1633 ...  
1634 ...  
1635 ...  
1636 ...  
1637 ...  
1638 ...  
1639 ...  
1640 ...  
1641 ...  
1642 ...  
1643 ...  
1644 ...  
1645 ...  
1646 ...  
1647 ...  
1648 ...  
1649 ...  
1650 ...  
1651 ...  
1652 ...  
1653 ...  
1654 ...  
1655 ...  
1656 ...  
1657 ...  
1658 ...  
1659 ...  
1660 ...  
1661 ...  
1662 ...  
1663 ...  
1664 ...  
1665 ...  
1666 ...  
1667 ...  
1668 ...  
1669 ...  
1670 ...  
1671 ...  
1672 ...  
1673 ...  
1674 ...  
1675 ...  
1676 ...  
1677 ...  
1678 ...  
1679 ...  
1680 ...  
1681 ...  
1682 ...  
1683 ...  
1684 ...  
1685 ...  
1686 ...  
1687 ...  
1688 ...  
1689 ...  
1690 ...  
1691 ...  
1692 ...  
1693 ...  
1694 ...  
169
```

The screenshot shows the Java IDE interface with the file `WebPage.java` open. The code is annotated with Javadoc comments explaining various methods and parameters. The methods include `convert`, `downloadfileBy`, and `downloadfile`. The code uses reflection to handle different URL converters and handles exceptions related to WebDriver and MauilException.

```
422     }
423     try {
424         if (!isMultipleWindow)
425             this.selectWindow();
426         else
427             WebKitDriver.setWebDriver(null);
428     } catch (UnreachableBrowserException ex) {
429         WebKitDriver.setWebDriver(null);
430     }
431 }
432 
433 /**
434  * Convert string url
435  * @param url
436  * @return String
437  */
438 public String convert(String url) {
439     if (ContextManager.getThreadContext() != null) {
440         String urlConvertClass = ContextManager.getThreadContext()
441             .getURLConvertClass();
442         if (urlConvertClass != null) {
443             try {
444                 Class<?> converterClass = Class.forName(urlConvertClass);
445                 Object converter = converterClass.newInstance();
446                 Method convertMethod = converterClass.getMethod("convertURL",
447                     String.class);
448                 return (String) convertMethod.invoke(converter, url);
449             } catch (Exception e) {
450                 logger.warn("Convert URL failed", e);
451             }
452         }
453     }
454     return url;
455 }
456 
457 /**
458  * Download a file to test-output/MauilTestSuite/downloads\ folder regardless
459  * of local or GRID
460  *
461  * @param locator
462  * @param fileName
463  * @return full path of download file
464  * @throws MauilException
465  */
466 public final String downloadfileBy(String fileName)
467     throws MauilException {
468     return downloadfile(driver.findElement(By.getAttribute("href")),
469                         fileName);
470 }
471 
472 /**
473  * Download a file to test-output/MauilTestSuite/downloads\ folder regardless
474  * of local or GRID
475  *
476  * @param locator
477  * @param fileName
478  * @return full path of download file
479  * @throws MauilException
480  */
481 public final String downloadfile(String locatorOrDownloadUrl,
482     String fileNamePostFix) throws MauilException {
483     if (locatorOrDownloadUrl.startsWith("Http://"))
484         locatorOrDownloadUrl.startsWith("Https://"));
485     // direct download
486     try {
487         locatorOrDownloadUrl = convert(locatorOrDownloadUrl);
488     } catch (Exception e) {
489     }
490 }
491 
```

The screenshot shows the Java IDE interface with the file `WebPage.java` open. The code is annotated with Javadoc comments explaining various methods and parameters. The methods include `savefile` and `downloadfile`. The code handles file streams, URL prefixes, and output directory logic to save files from URLs.

```
498     /*
499      * HtmlUnitDriver htmlUnitDriver = new HtmlUnitDriver();
500      * htmlUnitDriver.get(locatorOrDownloadUrl); String fileAsString =
501      * htmlUnitDriver.getPageSource();
502      */
503     // String fileAsString =
504     // session.downloadfileToString(locatorOrDownloadUrl, fileName);
505     // fileAsString = trimPrefix(fileAsString); byte[] byteArray =
506     // fileAsString.getBytes(); return saveDownloadedFile(byteArray,
507     // fileName);
508     // 
509     InputStream inputStream;
510     StringBuffer fileName;
511     String fileAsBuffer;
512     String fileName = URLHelper.getRandomHashCode("download") + "-" +
513         fileAsBufferPostFix;
514     try {
515         inputStream = URLHelper.openAsStream(locatorOrDownloadUrl);
516         String filetype = url.substring(url.lastIndexOf(".") + 1);
517         if (filetype.length() < 5 & filetype.charAt(0) == ".")
518             fileName = fileName + "-" + filetype;
519         String outputDirectory = ContextManager.getGlobal(context)
520             .getTestNGContext().getOutputDirectory();
521         String suiteName = ContextManager.getGlobal(context)
522             .getTestNGContext().getSuite().getName();
523         String downloadDir = outputDirectory + "/downloads/";
524         File d = new File(downloadDir);
525         if (!d.exists())
526             d.mkdirs();
527         OutputStream out = new FileOutputStream(downloadDir + fileName);
528         int read = 0;
529         byte[] bytes = new byte[1024];
530         while ((read = inputStream.read(bytes)) != -1) {
531             out.write(bytes, 0, read);
532         }
533         out.close();
534         sbMessage.append("Downloading file <a href=\"" + suiteName
535             + ".html\" " + fileName + " title=\"" + fileName
536             + "\">" + fileAsBufferPostFix + "</a>");
537     } catch (Throwable e) {
538         logger.warn("Ex", e);
539     }
540     Logging.logWebOutput(url, sbMessage.toString(), false);
541     fileName = outputDirectory + "/downloads/" + fileName;
542     // correct file path format
543     if (OSHelper.isWindows())
544         fileName = fileName.replaceAll("\\/", "\\\\");
545     return fileName;
546 }
547 
548 /**
549  * Save file
550  * @param url
551  * @return String
552  */
553 public static String savefile(String url) throws Exception {
554 }
```

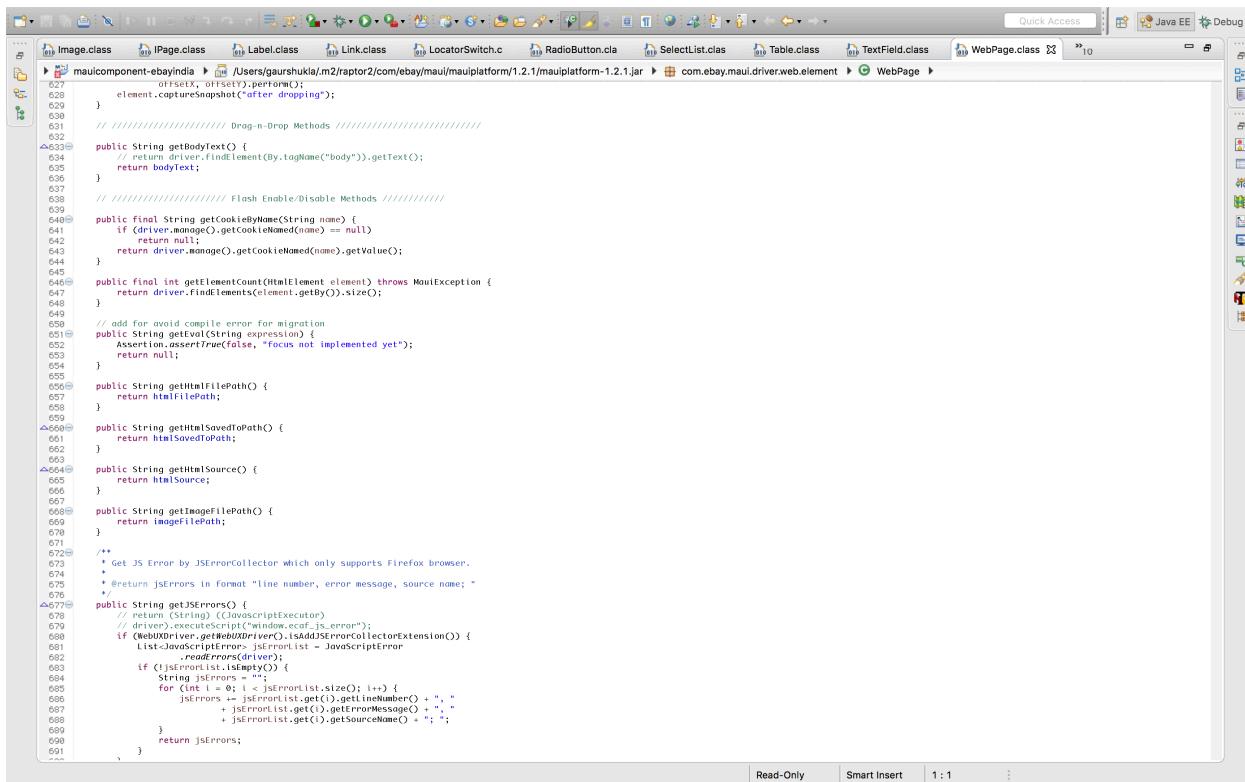


This screenshot shows a Java code editor with the file `com.ebay.mau.driver.web.element.WebPage.java` open. The code implements a static method `saveFile` to download files from a URL. It uses `URLHelper` to get the URL, opens a stream, and then writes the content to a local file in a download directory. The code handles exceptions and logs messages using `Logging`. It also includes a `dragAndDrop` method for interacting with web elements.

```
public static String saveFile(String url) throws Exception {
    URL url = URLHelper.getAbsoluteUrl(url);
    String fileNamePostfix = "";
    InputStream inputStream = URLHelper.openAsStream(url);
    String filetype = url.substring(url.lastIndexOf(".") + 1);
    String stringBuffer_sbMessage = new String(stringBuffer);
    stringBuffer_sbMessage.append("[").append(getRandomStringCode("download") + "-");
    stringBuffer_sbMessage.append(fileNamePostfix);
    if (filetype != null && filetype.length() < 5)
        fileName = fileName + "." + filetype;
    String suiteName = ContextManager.getGlobalContext()
        .getTestNGContext().getOutputDirectory();
    String suiteName = ContextManager.getGlobalContext().getTestNGContext()
        .getSuite().getName();
    try {
        String downloadDir = outputDirectory + "/downloads/";
        File d = new File(downloadDir);
        if (!d.exists()) {
            d.mkdir();
        }
        FileOutputStream out = new FileOutputStream(downloadDir + fileName);
        int read = 0;
        byte[] bytes = new byte[1024];
        while ((read = inputStream.read(bytes)) != -1) {
            out.write(bytes, 0, read);
        }
        out.close();
        sbMessage.append("Downloading file <a href=\"" + suiteName
            + "/downloads/" + fileName + " title=\"" + fileName + "\">");
        sbMessage.append(fileName);
    } catch (Throwable e) {
        logger.warn("Ex", e);
    }
    Logging.logWebOutput(url, sbMessage.toString(), false);
    fileName = outputDirectory + "/downloads/" + fileName;
    // correct file path format
    if (OSHelper.isWindows())
        fileName = fileName.replaceAll("\\/", "\\\\");
}
return fileName;
}

/** 
 * Drags an element a certain distance and then drops it
 */
public void dragAndDrop(WebElement element, int offsetX, int offsetY) {
    Logging.logWebOutput(element, "dragAndDrop", element.getAttribute("id"));
    element.captureSnapshot("before dragging");
    new Actions(driver).dragAndDropBy((WebElement) element.getElement(),
        offsetX, offsetY);
}

public void dragAndDrop(WebElement element, int offset) {
    Logging.logWebOutput(element, "dragAndDrop", element.getAttribute("id"));
    element.captureSnapshot("before dragging");
    new Actions(driver).dragAndDropBy((WebElement) element.getElement(),
        offset);
}
```



This screenshot shows a Java code editor with the file `com.ebay.mau.driver.web.element.WebPage.java` open. The code contains several utility methods: `getByText`, `getCookieByName`, `getElementsCount`, `getHTMLFilePath`, `getHTMLSavedPath`, `getHTMLSource`, `getImageFilePath`, `getJSErrors`, and `getJavaScriptExecutor`. The `getJSErrors` method specifically handles JavaScript errors by executing a script to get error details from the browser's error collector.

```
public String getByText() {
    // return driver.findElement(By.tagName("body")).getText();
    return bodyText;
}

public final String getCookieByName(String name) {
    if (driver.manage().getCookieNamed(name) == null)
        return null;
    return driver.manage().getCookieNamed(name).getValue();
}

public final int getElementsCount(HtmlElement element) throws MauException {
    return driver.findElements(element.getByQ()).size();
}

// add for visual compile error for migration
public String getEval(String expression) {
    Assertion.assertIfFalse(false, "focus not implemented yet");
    return null;
}

public String getHTMLFilePath() {
    return htmlFilePath;
}

public String getHTMLSavedPath() {
    return htmlSavedPath;
}

public String getHTMLSource() {
    return htmlSource;
}

public String getImageFilePath() {
    return imageFilePath;
}

/**
 * Get JS Error by JSErrorCollector which only supports Firefox browser.
 * @return jsErrors in format "line number, error message, source name"
 */
public String getJSErrors() {
    String jsErrors = "";
    if (driver instanceof JavaScriptExecutor)
        driver.executeScript("window.ecf_is_error");
    if (WebDriver.getWebDriver().isAddJSerrorCollectorExtension())
        List<JavaScriptError> jsErrorList = JavaScriptError
            .getJSErrorCollectorExtension();
    if (jsErrorList.isEmpty())
        String jsErrors = "";
    for (int i = 0; i < jsErrorList.size(); i++) {
        jsErrors += jsErrorList.get(i).getLineNumber() + ": "
            + jsErrorList.get(i).getErrorMessage() + ", "
            + jsErrorList.get(i).getSourceName() + ": ";
    }
    return jsErrors;
}
```

The screenshot shows the Java code for the `WebPage` class. The code includes methods for getting current URL, page ID, and window handle, setting timeout, getting title, URL, and frame ID, navigating back and forward, and initializing web elements. It also handles exceptions and logs actions.

```
692     }
693     return null;
694   }
695 
696   ▲696 public String getLocation() {
697     return driver.getCurrentUrl();
698   }
699 
700   public String getPageId() {
701     return pageId;
702   }
703 
704   public String getPopupWindowName() {
705     return popupWindowName;
706   }
707 
708   ▲708 public String getRqid() {
709     return rqid;
710   }
711 
712   public int getTimeout() {
713     return ContextManager.getThreadContext().getWebSessionTimeout();
714   }
715 
716   ▲716 public String getTitle() {
717     return driver.getTitle();
718   }
719 
720   public String getUrl() {
721     return url;
722   }
723 
724   public String getWindowHandle() {
725     return windowHandle;
726   }
727 
728   ▲728 public final void goBack() {
729     Logging.logWebStep(null, "goBack", false);
730     driver.navigate().back();
731     frameFlag = false;
732   }
733 
734   ▲734 public final void goForward() {
735     Logging.logWebStep(null, "goForward", false);
736     driver.navigate().forward();
737     frameFlag = false;
738   }
739 
740   ▲740 public void initWebElements() {
741     // Element will be initialized right before using, so no need additional
742     // steps
743 
744     * try {PageFactory.initElements(driver, this); }catch(Exception e) {
745     //ignore exception } driver = WebUDriver.getWebDriver(); Field[]
746     * fields = this.getClass().getDeclaredFields(); for (Field field :
747     * fields) { //System.out.println(field.getName());}
748     *
749     * try { if (field.get(this) instanceof HTMLElement) { Method method =
750     * HTMLElement.class.getMethod("init"); method.invoke(field.get(this));}
751     * else if (field.get(this) instanceof SelectList) { Method method =
752     * SelectList.class.getMethod("init"); method.invoke(Field.get(this));}
753     * catch (Exception e) {} }
754     * }
755   }
756 }
```

The screenshot continues the Java code for the `WebPage` class. It includes methods for checking cookie presence, getting frame status, maximizing windows, opening URLs, and handling unsupported command exceptions. The code uses logging and handles various browser types and exceptions.

```
757   }
758 
759   public final boolean isCookiePresent(String name) {
760     return getCookieByName(name) != null;
761   }
762 
763   ▲763 public boolean isFrame() {
764     return frameFlag;
765   }
766 
767   ▲767 public final void maximizeWindow() {
768     new BaseUtil(driver).maximizeWindow();
769   }
770 
771   ▲771 public void open(String url) throws Exception {
772     open(url, true);
773   }
774 
775   ▲775 private void open(String url, boolean convert) throws Exception {
776     if (this.getDriver() == null) {
777       Logging.logWebStep(url, "Open browser", false);
778       driver = webUDriver.createWebDriver();
779       // maximizeWindow();
780     }
781 
782     if (convert) {
783       String tempURL = convert(url);
784       url = tempURL;
785     }
786     setUrl(url);
787     Logging.logWebStep(url, "Open url <a href=\"" + url + "\">" + url
788     + "</a>", false);
789 
790     try {
791       System.out.println("Navigate to " + url);
792       driver.navigate().to(url);
793     } catch (UnsupportedOperationException e) {
794       handleIfTheLastWindowIsClosed();
795       Logging.logWebStep(url, "Open browser", false);
796       if (webUDriver.getConfig().getBrowser() != BrowserType.Android) {
797         driver = webUDriver.createWebDriver();
798         maximizeWindow();
799         driver.navigate().to(url);
800       } else {
801         driver.navigate().to(url);
802       }
803     }
804     catch (Exception e) {
805       throw e;
806     }
807     catch (UnsupportedCommandException e) {
808       Logging.log("get UnSupportedCommandException, retry");
809       driver = webUDriver.createWebDriver();
810       maximizeWindow();
811       driver.navigate().to(url);
812     }
813     catch (org.openqa.selenium.TimeoutException e) {
814       Logging.log("got time out when loading " + url + ", ignored");
815     }
816     catch (org.openqa.selenium.UnhandledAlertException e) {
817       Logging.log("got unhandledAlertException, retry");
818       driver.navigate().to(url);
819     }
820     catch (Throwable e) {
821       e.printStackTrace();
822       throw new WebDriverException(e);
823     }
824     switchToDefaultContent();
825     System.out.println("Done Navigate to " + url);
826   }
827 }
```

```

242     private void populateAndCapturePageSnapshot() {
243         try {
244             setTitle(driver.getTitle());
245             htmlSource = driver.getPageSource();
246         }
247         // Handle staleElementReferenceException for IE and Opera
248         catch (StaleElementReferenceException ignore) {
249             bodyText = driver.findElement(By.tagName("body")).getText();
250         }
251         catch (UnreachableBrowserException e) {
252             throw new WebDriverException(e);
253         }
254         catch (WebDriverException e) {
255             throw e;
256         }
257         capturePageSnapshot();
258     }
259
260     public final void refresh() throws PageNotCurrentException {
261         Logging.logWebStep(null, "refresh", false);
262         try {
263             driver.navigate().refresh();
264         }
265         catch (org.openqa.selenium.TimeoutException ex) {
266             Logging.log("got time out exception, ignore");
267         }
268
269         frameFlag = false;
270
271         try {
272             waitForPageToLoad();
273         }
274         catch (Exception e) {
275             throw new RuntimeException(e);
276         }
277
278         assertNoPageErrors();
279     }
280
281     public final void resizeTo(int width, int height) {
282         new BaseUtil(driver).resizeWindow(width, height);
283     }
284
285     public final void selectFrame(By by) {
286         Logging.logWebStep(null, "select frame, locator={" + by.toString()
287             + "}", false);
288         driver.switchTo().frame(driver.findElement(by));
289         frameFlag = true;
290     }
291
292     public final void selectFrame(String locator) {
293         Logging.logWebStep(null, "select frame, locator={" + locator + "}",
294             false);
295         driver.switchTo().frame(locator);
296         frameFlag = true;
297     }
298
299     public final void selectWindow() throws PageNotCurrentException {
300         Logging.logWebStep(null, "select window, locator=" + "window"
301             + getPopUpWindowName() + "\\", false);
302         // selectWindow(getPopUpWindowName());
303         driver.switchTo().window(
304             (String) driver.getWindowHandles().toArray()[0]);
305         waitForSeconds(1);
306     }
307
308     protected void setWindowSavedPath(String htmlSavedPath) {
309         this.htmlSavedPath = htmlSavedPath;
310     }
311
312     // protected void setHtmlSource(String htmlSource) {
313     // this.htmlSource = htmlSource;
314     // }
315
316     // protected void setLocation(String location) {
317     // this.location = location;
318     // }
319
320     protected void setTitle(String title) {
321         this.title = title;
322     }
323
324     protected void setUrl(String openUrl) {
325         this.url = openUrl;
326     }
327
328     public void switchToDefaultContent() {
329         try {
330             driver.switchTo().defaultContent();
331         }
332         catch (UnhandledAlertException e) {
333         }
334     }
335
336     private final void waitForPageToLoad() throws Exception {
337         try {
338             new WebDriverWait()
339                 .@Override
340                 public boolean until() {
341                     try {
342                         driver.switchTo().defaultContent();
343                         return true;
344                     }
345                     catch (NoSuchElementException ex) {
346                         ThreadHelper.waitForSeconds(2);
347                     }
348                     catch (WebDriverException e) {
349                         return false;
350                     }
351                 }
352             ).until(new ExpectedCondition<WebElement>("I need to wait for page to load",
353                 webDriver -> webDriver.getCurrentUrl().getWebSessionTimeout());
354         }
355         catch (WaitTimeOutException ex) {
356         }
357
358         // populate page info
359     }

```

The screenshot shows the Java code for the `WebPage` class. The code handles setting URLs, switching to default content, waiting for page load, and finding elements by various locators. It also includes methods for getting element URLs, texts, and sources.

```
923     protected void setUrl(String openurl) {
924         this.url = openurl;
925     }
926
927     public void switchToDefaultContent() {
928         try {
929             driver.switchTo().defaultContent();
930         } catch (UnhandledAlertException e) {
931             e.printStackTrace();
932         }
933     }
934
935     private final void waitForPageToLoad() throws Exception {
936         try {
937             new WebDriverWait(driver, 10).until(ExpectedConditions.presenceOfElementLocated(By.id("content")));
938         } catch (TimeoutException ex) {
939             ThreadHelper.waitForSeconds(2);
940         }
941     }
942
943     public boolean until() {
944         try {
945             driver.switchTo().defaultContent();
946             return true;
947         } catch (UnhandledAlertException ex) {
948             ThreadHelper.waitForSeconds(2);
949         } catch (WebDriverException e) {
950             e.printStackTrace();
951         }
952         return false;
953     }
954
955     // populate page info
956     try {
957         populateAndCapturePageSnapshot();
958     } catch (Exception ex) {
959         // ex.printStackTrace();
960         throw ex;
961     }
962
963     public WebElement getElement(By by, String elementName) {
964         WebElement element = null;
965         try {
966             element = driver.findElement(by);
967         } catch (ElementNotVisibleException e) {
968             Logging.error(elementName + " is not found with locator - " + by.toString());
969             throw e;
970         }
971         return element;
972     }
973
974     public String getElementUrl(By by, String name) {
975         return getElement(by, name).getAttribute("href");
976     }
977
978     public String getElementText(By by, String name) {
979         return getElement(by, name).getText();
980     }
981
982     public String getElementSrc(By by, String name) {
983         return getElement(by, name).getAttribute("src");
984     }
985 }
986
987
```

The screenshot shows the Java code for the `WebPageSection` class. It extends `BaseHtmlPage` and provides methods for creating sections, capturing snapshots, and getting locator information.

```
5 import com.ebay.mau.controller.Logging;
6 import com.ebay.mau.driver.web.ScreenshotUtil;
7 import com.ebay.mau.driver.web.WebElementUtil;
8 import com.ebay.mau.exception.MauException;
9
10 public abstract class WebPageSection extends BaseHtmlPage {
11
12     private String name = null;
13     private String locator = null;
14     protected WebElement element = null;
15     private By by = null;
16
17     public WebPageSection(String name) {
18         super();
19         this.name = name;
20     }
21
22     public WebPageSection(String name, By by) {
23         super();
24         this.name = name;
25         this.by = by;
26     }
27
28     public WebPageSection(String name, String locator) throws MauException {
29         super();
30         if (!locator.startsWith("//xpath=")) {
31             locator = locator.substring(6);
32         } else if (!locator.startsWith("//") && !locator.startsWith("//")) {
33             locator = "//[" + locator.substring(1) + " or @name=" + locator.substring(1)];
34         }
35
36         this.name = name;
37         this.locator = locator;
38         this.by = By.xpath(locator);
39     }
40
41     /**
42      * Captures page snapshot.
43     */
44     public void capturePageSnapshot() {
45         Screenshot screenShot = new ScreenshotUtil(driver).captureWebPageSnapshot();
46         String title = screenShot.getTitle();
47         String url = screenShot.getLocation();
48
49         Logging.logWebOutput(url, title + " " + Logging.buildScreenshotLog(screenShot) + ">", false);
50     }
51
52     public String getLocator() {
53         return locator;
54     }
55
56     public String getName() {
57         return name;
58     }
59
60     public By getBy() {
61         return by;
62     }
63
64     public boolean isPageSectionPresent() {
65         return isElementPresent(by);
66     }
67 }
68
```

```

11  public abstract class AbstractWebDriverFactory {
12      protected WebDriverConfig cfg;
13
14      protected WebDriver driver;
15
16      public AbstractWebDriverFactory(WebDriverConfig cfg) {
17          this.cfg = cfg;
18      }
19
20      public void cleanup() {
21          try {
22              if (driver != null) {
23                  try {
24                      System.out.println("quitting webdriver...."+Thread.currentThread().getId());
25                      driver.quit();
26                  } catch (WebDriverException ex) {
27                      if (WebBrowser.getWebBrowser().getConfig().getBrowser() != BrowserType.InternetExplore)
28                          System.out.println("Quit exception....");
29                      + WebBrowser.getWebBrowser().getConfig()
30                      .getBrowser().getName() + "!";
31                  }
32                  + ex.getMessage();
33              }
34              driver = null;
35          } catch (Exception ex) {
36              // ignore all exceptions
37          }
38      }
39
40      public WebDriver createWebDriver() throws Exception {
41          return null;
42      }
43
44      public WebDriver getWebDriver() {
45          return driver;
46      }
47
48      public WebDriverConfig getWebDriverConfig() {
49          return cfg;
50      }
51
52      public void setImplicitWaitTimeout(double timeout) {
53          if (timeout < 1)
54              driver.manage()
55                  .timeouts()
56                  .implicitlyWait((long) (timeout * 1000),
57                                  TimeUnit.MILLISECONDS);
58          else {
59              try {
60                  driver.manage()
61                  .timeouts()
62                  .implicitlyWait(new Double(timeout).intValue(),
63                                  TimeUnit.SECONDS);
64              } catch (Exception ex) {
65                  ex.printStackTrace();
66              }
67          }
68      }
69
70      public void setWebDriver(WebDriver driver) {
71          this.driver = driver;
72      }
73  }

```

Read-Only | Smart Insert | 1 : 1

```

11  public abstract class AbstractWebDriverFactory {
12      protected WebDriverConfig cfg;
13
14      protected WebDriver driver;
15
16      public AbstractWebDriverFactory(WebDriverConfig cfg) {
17          this.cfg = cfg;
18      }
19
20      public void cleanup() {
21          try {
22              if (driver != null) {
23                  try {
24                      System.out.println("quitting webdriver...."+Thread.currentThread().getId());
25                      driver.quit();
26                  } catch (WebDriverException ex) {
27                      if (WebBrowser.getWebBrowser().getConfig().getBrowser() != BrowserType.InternetExplore)
28                          System.out.println("Quit exception....");
29                      + WebBrowser.getWebBrowser().getConfig()
30                      .getBrowser().getName() + "!";
31                  }
32                  + ex.getMessage();
33              }
34              driver = null;
35          } catch (Exception ex) {
36              // ignore all exceptions
37          }
38      }
39
40      public WebDriver createWebDriver() throws Exception {
41          return null;
42      }
43
44      public WebDriver getWebDriver() {
45          return driver;
46      }
47
48      public WebDriverConfig getWebDriverConfig() {
49          return cfg;
50      }
51
52      public void setImplicitWaitTimeout(double timeout) {
53          if (timeout < 1)
54              driver.manage()
55                  .timeouts()
56                  .implicitlyWait((long) (timeout * 1000),
57                                  TimeUnit.MILLISECONDS);
58          else {
59              try {
60                  driver.manage()
61                  .timeouts()
62                  .implicitlyWait(new Double(timeout).intValue(),
63                                  TimeUnit.SECONDS);
64              } catch (Exception ex) {
65                  ex.printStackTrace();
66              }
67          }
68      }
69
70      public void setWebDriver(WebDriver driver) {
71          this.driver = driver;
72      }
73
74      public void setWebDriverConfig(WebDriverConfig cfg) {
75          this.cfg = cfg;
76      }
77  }
78

```

Read-Only | Smart Insert | 1 : 1

The screenshot shows the AndroidCapabilitiesFactory.java file in an IDE. The code implements the ICapabilitiesFactory interface and creates DesiredCapabilities objects for Android devices. It handles various configuration parameters like proxy, browser version, platform, resolution, and capabilities like takesScreenshot and acceptSSLCerts.

```
3 import org.openqa.selenium.Proxy;
4 import org.openqa.selenium.remote.CapabilityType;
5 import org.openqa.selenium.remote.DesiredCapabilities;
6
7 import com.ebay.maui.controller.ContextManager;
8 import com.ebay.maui.driver.web.WebDriverConfig;
9 import com.ebay.qa.autobot.listeners.BeaconListener;
10
11 public class AndroidCapabilitiesfactory implements ICapabilitiesFactory {
12
13     public DesiredCapabilities createCapabilities(WebDriverConfig cfg) {
14         DesiredCapabilities capability = DesiredCapabilities.android();
15
16         if (cfg.getDeviceTarget() != null) {
17             capability = SelendroidCapabilities
18                 .android(getDeviceTargetPlatform(cfg.getAndroidTarget()));
19         } else {
20             capability = SelendroidCapabilities.android();
21         }
22
23         capability.setCapability(CapabilityType.ENABLED, true);
24         if (cfg.isEnableBrowser()) {
25             capability.setJavaScriptEnabled(true);
26         } else {
27             capability.setJavaScriptEnabled(false);
28         }
29         capability.setCapability(CapabilityType.TAKES_SCREENSHOT, true);
30         capability.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);
31
32         if (cfg.getBrowserVersion() != null) {
33             capability.setVersion(cfg.getBrowserVersion());
34         }
35
36         if (cfg.getPlatform() != null) {
37             capability.setPlatform(cfg.getPlatform());
38         }
39
40         if (cfg.getResolution() != null) {
41             capability.setCapability("screensize", cfg.getResolution());
42         }
43
44         if (cfg.getProxyHost() != null) {
45             Proxy proxy = cfg.getProxy();
46             capability.setCapability(CapabilityType.PROXY, proxy);
47         }
48
49         if (ContextManager.getThreadObjectContext().isBeaconListenerEnabled()) {
50             BeaconListener.setCapabilityMetrics(capability);
51         }
52
53         return capability;
54     }
55
56     /* private DeviceTargetPlatform getDeviceTargetPlatform(String deviceTarget) {
57     *     if (deviceTarget != null) {
58     *         for (DeviceTargetPlatform b : DeviceTargetPlatform.values()) {
59     *             if (b.getName().equalsIgnoreCase(deviceTarget)) {
60     *                 return b;
61     *             }
62     *         }
63     *     }
64     *     return DeviceTargetPlatform.ANDROID17;
65     */
66 }
67 }
```

The screenshot shows the ChromeCapabilitiesFactory.java file in an IDE. This class implements the ICapabilitiesFactory interface and creates DesiredCapabilities objects for Chrome. It handles options like userAgentOverride, chromeExtensionPath, experimentalOptions, proxyHost, platform, resolution, and various capabilities.

```
21
22 public class ChromeCapabilitiesfactory implements ICapabilitiesFactory {
23
24     public DesiredCapabilities createCapabilities(WebDriverConfig cfg) {
25         DesiredCapabilities capability = DesiredCapabilities.chrome();
26
27         ChromeOptions options = new ChromeOptions();
28         if (cfg.getUserAgentOverride() != null) {
29             options.addArguments("--user-agent=" + cfg.getUserAgentOverride());
30         }
31         capability.setCapability("chrome.switches",
32             Arrays.asList("--user-agent=" + cfg.getUserAgentOverride()));
33
34     }
35
36     if (cfg.getChromeExtensionPath() != null) {
37         options.addExtensions(new File(cfg.getChromeExtensionPath()));
38     }
39
40     if (cfg.isEnableCookie()) {
41         Map<String, Integer> prefs = new HashMap<String, Integer>();
42         prefs.put("profile.default_content_settings.cookies", 2);
43         options.setExperimentalOption("prefs", prefs);
44     }
45
46     capability.setCapability(ChromeOptions.CAPABILITY, options);
47
48     if (cfg.isEnableJavaScript()) {
49         capability.setJavaScriptEnabled(true);
50     } else {
51         capability.setJavaScriptEnabled(false);
52     }
53     capability.setCapability(CapabilityType.TAKES_SCREENSHOT, true);
54     capability.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);
55
56     if (cfg.getBrowserVersion() != null) {
57         capability.setVersion(cfg.getBrowserVersion());
58     }
59
60     if (cfg.getPlatform() != null) {
61         capability.setPlatform(cfg.getPlatform());
62     }
63
64     if (cfg.getResolution() != null) {
65         capability.setCapability("resolution", cfg.getResolution());
66     }
67
68     if (cfg.getProxyHost() != null) {
69         Proxy proxy = cfg.getProxy();
70         capability.setCapability(CapabilityType.PROXY, proxy);
71     }
72
73     if (cfg.getChromeBinPath() != null) {
74         capability.setCapability("chrome.binary", cfg.getChromeBinPath());
75     }
76
77     // Set ChromeDriver for local mode
78     if (cfg.getDriver() == WebDriverConfig.LocallyOnMC) {
79         String driverVersion = System.getProperty("webdriver.chrome.driver");
80         if (chromedriverPath == null) {
81             try {
82                 if (System.getProperty("webdriver.chrome.driver") != null) {
83                     System.out.println("Chrome driver get from property:");
84                     + System.getProperty("webdriver.chrome.driver"));
85                     System.setProperty("webdriver.chrome.driver",
86                         System.getProperty("webdriver.chrome.driver")));
87             } catch (Exception e) {
88                 e.printStackTrace();
89             }
90         }
91     }
92 }
```

The screenshot shows the Eclipse IDE interface with the Java EE perspective selected. The code editor displays the `ChromeCapabilitiesFactory` class from the `com.ebay.maui.driver.web.factory` package. The code handles the creation of WebDriver capabilities for Chrome, including setting proxy, resolution, and chromedriver paths based on configuration parameters.

```
1  package com.ebay.maui.driver.web.factory;
2
3  import java.io.IOException;
4  import java.util.concurrent.TimeUnit;
5
6  import org.openqa.selenium.UnsupportedCommandException;
7  import org.openqa.selenium.WebDriver;
8  import org.openqa.selenium.chrome.ChromeDriver;
9
10 import com.ebay.maui.driver.web.WebDriverConfig;
11
12 public class ChromeDriverFactory extends AbstractWebDriverFactory implements
13     IWebDriverFactory {
14
15     public ChromeDriverFactory(WebDriverConfig cfg) {
16         super(cfg);
17     }
18
19     /**
20      * create native driver instance, designed for unit testing
21      *
22      * @return
23      */
24     protected WebDriver createNativeDriver() {
25         return new ChromeDriver();
26     }
27
28     @Override
29     public WebDriver createWebDriver() throws IOException {
30         WebDriverConfig cfg = this.getWebDriverConfig();
31
32         driver = createNativeDriver();
33
34         setImplicitWaitTimeout(cfg.getImplicitWaitTimeout());
35         if (cfg.getPageLoadTimeout() > 0) {
36             setPageLoadTimeout(cfg.getPageLoadTimeout());
37         }
38         this.setWebDriver(driver);
39         return driver;
40     }
41
42     protected void setPageLoadTimeout(long timeout) {
43         try {
44             driver.manage().timeouts()
45                 .pageLoadTimeout(timeout, TimeUnit.SECONDS);
46         } catch (UnsupportedCommandException e) {
47             // chromedriver does not support pageLoadTimeout
48         }
49     }
50
51
52 }
```

The screenshot shows the Eclipse IDE interface with the Java EE perspective selected. The code editor displays the `WebPage` class from the `com.ebay.maui.driver.web.factory` package. The class contains methods for handling resources and creating web drivers, specifically for Chrome.

```
1  package com.ebay.maui.driver.web.factory;
2
3  import java.io.IOException;
4  import java.util.concurrent.TimeUnit;
5
6  import org.openqa.selenium.UnsupportedCommandException;
7  import org.openqa.selenium.WebDriver;
8  import org.openqa.selenium.chrome.ChromeDriver;
9
10 import com.ebay.maui.driver.web.WebDriverConfig;
11
12 public class WebPage {
13
14     public void handleExtractResources() throws IOException {
15         String dir = this.getClass().getResource("/").getPath()
16             + "moxiextern";
17         dir = FileHelper.decodePath(dir);
18
19         if (!new File(dir).exists()) {
20             System.out.println("Handling chrome resources in " + dir);
21             FileHelper.extractJar(dir, WebDriverExternalResources.class);
22         }
23
24         if (new File(dir + OSHelper.getSlash() + "chromedriver.exe").exists()) {
25             FileHelper.extractJar(dir, WebDriverExternalResources.class);
26         }
27
28         if (OSHelper.isWindows()) {
29             System.setProperty("webdriver.chrome.driver", dir
30                 + "\\chromedriver.exe");
31         } else {
32             System.setProperty("webdriver.chrome.driver", dir + "/chromedriver");
33             new File(dir + "/chromedriver").setExecutable(true);
34         }
35     }
36
37 }
```

The image shows two side-by-side Java code editors, likely from an IDE like Eclipse or IntelliJ IDEA. Both editors are displaying the same class, `FirefoxCapabilitiesFactory`, with minor differences in the code content.

**Editor 1 (Top):**

```
1 package com.ebay.mau.driver.web.factory;
2
3 import custom.profileDirCustFF.FPProfileWorker;
4
5 public class FirefoxCapabilitiesFactory implements ICapabilitiesFactory {
6     static boolean isProfileCreated = false;
7     private static Object lockProfile = new Object();
8     private static Object lockNewProfile = new Object();
9
10    protected void configProfile(FirefoxProfile profile, WebDriverConfig cfg) {
11        profile.setAcceptUntrustedCertificates(cfg
12            .isSetAcceptUntrustedCertificates());
13        profile.setAssumeUntrustedCertificateIssuer(cfg
14            .isSetAssumeUntrustedCertificateIssuer());
15
16        if (cfg.getFfbInPath() != null)
17            System.setProperty("webdriver.firefox.bin", cfg.getFfbInPath());
18
19        if (cfg.getUserAgentOverride() != null)
20            profile.setPreference("general.useragent.override",
21                cfg.getUserAgentOverride());
22
23        if (cfg.getNtLMAuthTrustedURIs() != null)
24            profile.setPreference("network.automatic-ntlm-auth.trusted-uris",
25                "http://127.0.0.1/*");
26        if (cfg.getProxyDownloadDir() != null) {
27            profile.setPreference("browser.download.dir",
28                cfg.getProxyDownloadDir());
29            profile.setPreference("browser.download.folderList", 2);
30            profile.setPreference("browser.download.manager.showWhenStarting",
31                false);
32            profile.setPreference("browser.helperApps.neverAsk.saveToDisk",
33                "application/etext-stream;text/plain,application/pdf,application/zip,text/csv,text/html");
34
35        if (!cfg.isEnableJavaScript())
36            profile.setPreference("javascript.enabled", false);
37        else {
38            Add Firefox extension to collect JS Error
39            if (cfg.isAddJSSErrorCollectorExtension()) {
40                try {
41                    JavaScriptError.addExtension(profile);
42                } catch (IOException e) {
43                }
44            }
45        }
46
47        if (!cfg.isEnableCookie())
48            profile.setPreference("network.cookie.cookieBehavior", 2);
49
50        // fix permission denied problem
51        profile.setPreference("capability.policy.Window.QueryInterface", "allAccess");
52        profile.setPreference("capability.policy.default.Window.QueryInterface", "allAccess");
53        profile.setPreference("capability.policy.Window.frameElement.get",
54            "allAccess");
55        profile.setPreference("capability.policy.HTMLDocument.compatMode.get",
56            "allAccess");
57        profile.setPreference("capability.policy.Document.compatMode.get",
58            "allAccess");
59        profile.setPreference("capability.policy.default.Document.compatMode.get",
60            "allAccess");
61        profile.setEnableNativeEvents(false);
62    }
63
64    public DesiredCapabilities createCapabilities(WebDriverConfig cfg) {
65        DesiredCapabilities capability = DesiredCapabilities.firefox();
66
67        FirefoxProfile profile = getFirefoxProfile(cfg);
68        configProfile(profile, cfg);
69        capability.setCapability(FirefoxDriver.PROFILE, profile);
70
71        if (cfg.isEnableJavaScript())
72            capability.setJavaScriptEnabled(true);
73        else
74            capability.setJavaScriptEnabled(false);
75
76        capability.setCapability(CapabilityType.TAKES_SCREENSHOT, true);
77        capability.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);
78
79        if (cfg.getBrowserVersion() != null) {
80            capability.setVersion(cfg.getBrowserVersion());
81
82        if (cfg.getPlatform() != null) {
83            capability.setPlatform(cfg.getPlatform());
84
85        if (cfg.getResolution() != null) {
86            capability.setCapability("resolution", cfg.getResolution());
87
88        if (cfg.getProxyHost() != null)
89            capability.setCapability(CapabilityType.PROXY, cfg.getProxy());
90
91        if (ContextManager.currentThreadContext().isDebugEnabled())
92            DebugListener.setCapabilityMetrics(capability);
93
94        return capability;
95
96    }
97
98    protected FirefoxProfile createFirefoxProfile(String path) {
99        synchronized (lockNewProfile) {
100            if (path != null)
101                return new FirefoxProfile(new File(path));
102            else
103                return new FirefoxProfile();
104        }
105
106    }
107
108    /**
109     * extractDefaultProfile to a folder
110     * @param profilePath
111     *          The folder to store the profile
112     * @throws IOException
113     */
114    protected void extractDefaultProfile(String profilePath) throws IOException {
115        synchronized (lockProfile) {
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
213
```

The screenshot shows an IDE interface with multiple tabs open. The current tab displays Java code for extracting a Firefox profile. The code includes several protected methods: `extractDefaultProfile`, `getFirefoxProfile`, and `getFirefoxProfilePath`. It handles exceptions and prints logs to `System.out`. The code uses `FileHelper` to delete and extract files from a jar. It also checks if a profile path is null and if a file exists at that path. The code is annotated with Javadoc comments and includes imports for `java.io.IOException`, `java.util.logging.Logger`, and `java.util.logging.Level`.

```
146     protected void extractDefaultProfile(String profilePath) throws IOException {
147         synchronized (lockProfile) {
148             try {
149                 if (!isProfileCreated) {
150                     System.out.println("start create profile");
151                     FileHelper.deleteDirectory(profilePath);
152                     FileHelper.extractJar(profilePath, FFProfileWorker.class);
153                 }
154             } catch (Exception ex) {
155                 ex.printStackTrace();
156             }
157         }
158         isProfileCreated = true;
159     }
160 
161     protected synchronized FirefoxProfile getFirefoxProfile(WebDriverConfig cfg) {
162         String profilePath = cfg.getProfilePath();
163         FirefoxProfile profile = null;
164         String realPath = null;
165         if (cfg.isUseFirefoxDefaultProfile())
166             realPath = getFirefoxProfilePath(path);
167         else
168             realPath = null;
169         profile = createFirefoxProfile(realPath);
170         return profile;
171     }
172 
173     protected String getFirefoxProfilePath(String path) {
174         String realPath = null;
175         if (path != null && !new File(path).exists()) {
176             Logging.log("Firefox profile path:" + path
177                         + " can't be found, use default");
178             path = null;
179         }
180         if (path != null) {
181             realPath = path;
182         } else {
183             try {
184                 // String slash = "\\";;
185                 // String profilePath = "C:\\grid\\profile";
186                 String profilePath = this.getClass().getResource("/").getPath()
187                     + "/profile";
188                 profilePath = URLDecoder.decode(profilePath);
189                 // if (!OSHelper.isWindows()) {
190                 //     slash = "/";
191                 //     profilePath = "/tmp/mozilla";
192                 // }
193                 // extractDefaultProfile(profilePath);
194                 realPath = profilePath + OSHelper.getSlash()
195                           + "customProfileDirCUSTFF";
196             } catch (Exception e) {
197                 e.printStackTrace();
198             }
199             realPath = null;
200         }
201         System.out.println("FirefoxProfile: " + realPath);
202         return realPath;
203     }
204 }
```

The screenshot shows an IDE interface with multiple tabs open. The current tab displays Java code for the `FirefoxDriverFactory` class, which implements `AbstractWebDriverFactory`. The code includes methods for creating native drivers and web drivers. It handles timeouts and socket exceptions. The code is annotated with Javadoc comments and includes imports for `java.util.concurrent.TimeUnit`, `java.util.logging.Logger`, and `java.util.logging.Level`.

```
11 public class FirefoxDriverFactory extends AbstractWebDriverFactory implements
12     IWebDriverFactory {
13     private long timeout = 60;
14 
15     /**
16      * @param cfg
17      *          the configuration of the FirefoxDriver
18      */
19     public FirefoxDriverFactory(WebDriverConfig cfg) {
20         super(cfg);
21     }
22 
23     /**
24      * Create native driver instance, designed for unit testing
25      */
26     @Override
27     protected WebDriver createNativeDriver() {
28         return new FirefoxDriver();
29     }
30 
31     @Override
32     public WebDriver createWebDriver() {
33         WebDriverConfig cfg = this.getWebDriverConfig();
34 
35         System.out.println("start create firefox");
36         driver = createWebDriverWithTimeout();
37 
38         System.out.println("end create firefox");
39 
40         // Implicit Waits to handle dynamic element. The default value is 5
41         // seconds.
42         setImplicitWaitTimeout(cfg.getImplicitWaitTimeout());
43         if (cfg.getPageLoadTimeout() > 0) {
44             setPageLoadTimeout(cfg.getPageLoadTimeout());
45         }
46 
47         this.setWebDriver(driver);
48         return driver;
49     }
50 
51     /**
52      * Create webDriver, capture socket exception and retry with timeout
53      */
54     protected WebDriver createWebDriverWithTimeout() {
55         long time = 0;
56         while (time < getTimeout()) {
57             try {
58                 driver = createNativeDriver();
59                 return driver;
60             } catch (WebDriverException ex) {
61                 if (ex.getMessage().contains("SocketException"))
62                     if (!ex.getMessage().contains("Failed to connect to binary FirefoxBinary"))
63                         if (ex.getMessage().contains("Unable to bind to locking port 7054 within 45000 ms"))
64                             try {
65                                 Thread.sleep(1000);
66                             } catch (InterruptedException e) {
67 
68 
69 
70 
71 
72 
73 
74 
```

The screenshot shows a Java code editor with the following code:

```

    39     driver = createWebDriverWithTimeout();
    40
    41     System.out.println("end create Firefox");
    42
    43     // Implicit Waits to handle dynamic element. The default value is 5
    44     // seconds.
    45     setImplicitWaitTimeout(cfg.getImplicitWaitTimeout());
    46     if (cfg.getPageLoadTimeout() >= 0) {
    47         setPageLoadTimeout(cfg.getPageLoadTimeout());
    48     }
    49
    50     this.setWebDriver(driver);
    51     return driver;
    52 }
    53
    54 /**
    55 * Create webDriver, capture socket exception and retry with timeout
    56 */
    57 @return WebDriver
    58
    59 protected WebDriver createWebDriverWithTimeout() {
    60     long time = 0;
    61     while (time < getTimeout()) {
    62         try {
    63             driver = createNativeDriver();
    64         } catch (WebDriverException ex) {
    65             if (ex.getMessage().contains("SocketException"))
    66                 || ex.getMessage().contains("Failed to connect to binary FirefoxBinary")
    67                 || ex.getMessage()
    68                     .contains(
    69                         "Unable to bind to locking port 7054 within 45000 ms"));
    70             try {
    71                 Thread.sleep(1000);
    72             } catch (InterruptedException e) {
    73                 time++;
    74             }
    75         } else
    76             throw new RuntimeException(ex);
    77     }
    78 }
    79
    80     throw new RuntimeException(
    81         "Got exception when creating webDriver with socket timeout 1 minute");
    82 }
    83
    84 /**
    85 * It's designed for shorten timeout in unit testing
    86 */
    87 @return timeout
    88
    89
    90 protected long getTimeout() {
    91     return timeout;
    92 }
    93
    94 // protected void setImplicitWaitTimeout(long timeout)
    95 // {
    96 //     driver.manage().timeouts().implicitlyWait(timeout, TimeUnit.SECONDS);
    97 // }
    98
    99 protected void setPageLoadTimeout(long timeout) {
    100     driver.manage().timeouts().pageLoadTimeout(timeout, TimeUnit.SECONDS);
    101 }
    102 }

```

The code is annotated with Javadoc-style comments and includes exception handling for socket errors and timeouts.

The screenshot shows a Java code editor and a Project Explorer side-by-side.

**Code Editor Content:**

```

1 package com.ebay.maul.driver.web.factory;
2
3 import org.openqa.selenium.Proxy;
4 import org.openqa.selenium.remote.CapabilityType;
5 import org.openqa.selenium.remote.DesiredCapabilities;
6
7 import com.ebay.maul.driver.web.WebDriverConfig;
8
9 public class HtmlUnitCapabilitiesFactory implements ICapabilitiesFactory{
10
11     public DesiredCapabilities createCapabilities(WebDriverConfig cfg) {
12         DesiredCapabilities capability = DesiredCapabilities.htmlUnit();
13
14         if (cfg.isEnableJavascript())
15             capability.setJavascriptEnabled(true);
16         else
17             capability.setJavascriptEnabled(false);
18
19         capability.setCapability(CapabilityType.TAKES_SCREENSHOT, true);
20
21         if (cfg.getBrowserVersion() != null) {
22             capability.setVersion(cfg.getBrowserVersion());
23
24         if (cfg.getPlatform() != null) {
25             capability.setPlatform(cfg.getPlatform());
26
27         if (cfg.getProxyHost() != null) {
28             Proxy proxy = cfg.getProxy();
29             capability.setCapability(CapabilityType.PROXY, proxy);
30
31         }
32
33         }
34
35         return capability;
36
37     }
38
39 }
40

```

**Project Explorer Content:**

- com.ebay.maul.driver.api
- com.ebay.maul.driver.db
- com.ebay.maul.driver.email
- com.ebay.maul.driver.ssh
- com.ebay.maul.driver.web
- com.ebay.maul.driver.web.element
  - BaseHtmlPage.class
  - BaseHtmlPage
  - BasePageObject.class
  - Button.class
  - CheckBox.class
  - Form.class
  - GenericWebPage.class
  - HtmlElement.class
  - Image.class
  - iPage.class
  - Label.class
  - Link.class
  - LocatorSwitch.class
  - LocatorType.class
  - RadioButton.class
  - SelectList.class
  - Table.class
  - TextField.class
  - WebPage.class
  - WebPageSection.class
  - WebPageSection
- com.ebay.maul.driver.web.factory
  - AbstractWebDriverFactory.class
  - AbstractWebDriverFactory
  - AndroidCapabilitiesFactory.class
  - ChromeCapabilitiesFactory.class
  - ChromeDriverFactory.class
  - FirefoxCapabilitiesFactory.class
  - FirefoxDriverFactory.class
  - HtmlUnitCapabilitiesFactory.class
  - HtmlUnitDriverFactory.class
  - ICapabilitiesFactory.class
  - IECapabilitiesFactory.class
  - IDriverFactory.class
  - IWebDriverFactory.class
  - OperaCapabilitiesFactory.class
  - OperaDriverFactory.class
  - PhantomJSCapabilitiesFactory.class
  - RemoteDriverFactory.class
  - SafariCapabilitiesFactory.class

**Call Hierarchy:**

Members calling constructors of 'WebPageSection' - in workspace

- WebPageSection(String, By) - com.ebay.maul.driver.web.element.WebPageSection
- WebPageSection(String, String) - com.ebay.maul.driver.web.element.WebPageSection
- WebPageSection(String) - com.ebay.maul.driver.web.element.WebPageSection

Project Explorer   Navigator   Type Hierarchy

ChromeDriverFac FirefoxCapabili FirefoxDriverFa HtmlUnitCapabil HtmlUnitDriverF

```

1 package com.ebay.maul.driver.web.factory;
2
3 import org.openqa.selenium.WebDriver;
4 import org.openqa.selenium.htmlunit.HtmlUnitDriver;
5
6 import com.ebay.maul.controller.Logging;
7 import com.ebay.maul.driver.web.WebDriverConfig;
8
9 public class HtmlUnitDriverFactory extends AbstractWebDriverFactory implements IWebDriverFactory{
10
11     public HtmlUnitDriverFactory(IWebDriverConfig cfg) {
12         super(cfg);
13     }
14
15
16
17     @Override
18     public WebDriver createWebDriver() {
19         WebDriverConfig cfg = this.getWebDriverConfig();
20
21         driver = new HtmlUnitDriver((IHtmlUnitCapabilitiesFactory) createCapabilities(cfg));
22
23         //Implicit wait to handle dynamic elements. The default value is 5 seconds.
24         setImplicitWaitTimeout(cfg.getImplicitWaitTimeout());
25         if(cfg.getPageLoadTimeout() > 0)
26             Logging.log("htmlunit doesn't support pageLoadTimeout, ignoring...");
27         //java.lang.RuntimeException: java.lang.UnsupportedOperationException: pageLoadTimeout
28         //driver.manage().timeouts().pageLoadTimeout((cfg.getPageLoadTimeout()), TimeUnit.SECONDS);
29
30         this.setWebDriver(driver);
31
32         return driver;
33     }
34
35 }
36
37
38

```

Markers Properties Servers Data Source Explorer Snippets Console Progress Results of running class RaiseClass Package Explorer Call Hierar

Members calling constructors of 'WebPageSection' - in workspace

- WebPageSection(String, By) - com.ebay.maul.driver.web.element.WebPageSection
- WebPageSection(String, String) - com.ebay.maul.driver.web.element.WebPageSection
- WebPageSection(String) - com.ebay.maul.driver.web.element.WebPageSection

Line Call

Read-Only Smart Insert 1:1

Project Explorer   Navigator   Type Hierarchy

FirefoxCapabili FirefoxDriverFa HtmlUnitCapabil HtmlUnitDriverF ICapabilitiesFa IECapabilitiesF

```

1 package com.ebay.maul.driver.web.factory;
2
3 import org.openqa.selenium.remote.DesiredCapabilities;
4
5 import com.ebay.maul.driver.web.WebDriverConfig;
6
7
8 public interface ICapabilitiesFactory {
9
10     public DesiredCapabilities createCapabilities(IWebDriverConfig cfg);
11
12 }
13

```

Markers Properties Servers Data Source Explorer Snippets Console Progress Results of running class RaiseClass Package Explorer Call Hierar

Members calling constructors of 'WebPageSection' - in workspace

- WebPageSection(String, By) - com.ebay.maul.driver.web.element.WebPageSection
- WebPageSection(String, String) - com.ebay.maul.driver.web.element.WebPageSection
- WebPageSection(String) - com.ebay.maul.driver.web.element.WebPageSection

Line Call

Read-Only Smart Insert 1:1

```

19 public class IECapabilitiesFactory implements ICapabilitiesFactory {
20     ...
21     private void handleExtractResources() throws IOException {
22         String dir = this.getClass().getResource(">").getPath();
23         ...
24         FileHelper.extractJar(dir, WebDriverExternalResources.class);
25         if (new File(dir + "\\" + "IEDriverServer.exe").exists()) {
26             if (OSHelper.getIEVersion() < 10) {
27                 FileHelper.copyFile(dir + "\\" + "IEDriverServer_x64.exe", dir
28                     + "\\" + "IEDriverServer.exe");
29             } else {
30                 FileHelper.copyFile(dir + "\\" + "IEDriverServer_Win32.exe", dir
31                     + "\\" + "IEDriverServer.exe");
32             }
33             System.setProperty("webdriver.ie.driver", dir + "\\" + "IEDriverServer.exe");
34             System.out.println(dir + "\\" + "IEDriverServer.exe");
35         }
36     }
37     ...
38     public DesiredCapabilities createCapabilities(WebDriverConfig cfg) {
39         ...
40         // IE driver for Local mode
41         if (cfg.getMode() == WebDriverMode.LocalyOnRC) {
42             if (cfg.getDriverPath() != null)
43                 System.setProperty("webdriver.ie.driver", cfg.getDriverPath());
44             else if (System.getenv("webdriver.ie.driver") != null)
45                 System.setProperty("webdriver.ie.driver", System.getenv("webdriver.ie.driver"));
46             else{
47                 try {
48                     handleExtractResources();
49                 } catch (IOException ex) {
50                 }
51             }
52         }
53     }
54     ...
55     DesiredCapabilities capability = DesiredCapabilities.internetExplorer();
56     if (cfg.isEnableJavascipt())
57         capability.setJavaScriptEnabled(true);
58     else{
59         capability.setJavaScriptEnabled(false);
60         capability.setCapability(CapabilityType.TAKES_SCREENSHOT, true);
61         capability.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);
62         capability.setCapability(InternetExplorerDriver.IGNORE_ZOOM_SETTING, true);
63     }
64     if (cfg.getBrowserVersion() != null) {
65         capability.setVersion(cfg.getBrowserVersion());
66     }
67     if (cfg.getPlatform() != null) {
68         capability.setPlatform(cfg.getPlatform());
69     }
70     if (cfg.getResolution() != null) {
71         capability.setCapability("resolution", cfg.getResolution());
72     }
73     ...
74     if (cfg.getProxyHost() != null) {
75         Proxy proxy = cfg.getProxy();
76         capability.setCapability(CapabilityType.PROXY, proxy);
77     }
78     // To force protected mode transition, but users should set the browser
79     // proxy to reduce the strange behavior
80     // If a Protected Mode boundary is crossed, very unexpected behavior
81     // including hangs, element location not working, and clicks not being
82     // propagated
83     // This might not be a big problem in our testing since almost we should
84     // test in the same zone. added in 6/18/2013
85     capability
86         .setCapability(
87             InternetExplorerDriver.INTRODUCE_FLAKINESS_BY_IGNORING_SECURITY_DOMAINS,
88             true);
89     if (ContextManager.getThreadContext().isBeBoshiListenerEnabled()) {
90         BeBoshiListener.setCapabilityMetrics(capability);
91     }
92     ...
93     return capability;
94 }
95 ...
96 }
97 ...
98 }
99 ...
100 }
101 ...
102 }
103 ...
104 }
105 ...
106 }
107 ...
108 }
109 ...
110 }
111 ...
112 }
113 ...
114 }
115 ...
116 }
117 ...
118 }
119 ...
120 }
121 ...
122 }
123 ...
124 }
125 ...
126 }
127 ...
128 }
129 ...
130 }
131 ...
132 }
133 ...
134 }
135 ...
136 }
137 ...
138 }
139 ...
140 }
141 ...
142 }
143 ...
144 }
145 ...
146 }
147 ...
148 }
149 ...
150 }
151 ...
152 }
153 ...
154 }
155 ...
156 }
157 ...
158 }
159 ...
160 }
161 ...
162 }
163 ...
164 }
165 ...
166 }
167 ...
168 }
169 ...
170 }
171 ...
172 }
173 ...
174 }
175 ...
176 }
177 ...
178 }
179 ...
180 }
181 ...
182 }
183 ...
184 }
185 ...
186 }
187 ...
188 }
189 ...
190 }
191 ...
192 }
193 ...
194 }
195 ...
196 }
197 ...
198 }
199 ...
200 }
201 ...
202 }
203 ...
204 }
205 ...
206 }
207 ...
208 }
209 ...
210 }
211 ...
212 }
213 ...
214 }
215 ...
216 }
217 ...
218 }
219 ...
220 }
221 ...
222 }
223 ...
224 }
225 ...
226 }
227 ...
228 }
229 ...
230 }
231 ...
232 }
233 ...
234 }
235 ...
236 }
237 ...
238 }
239 ...
240 }
241 ...
242 }
243 ...
244 }
245 ...
246 }
247 ...
248 }
249 ...
250 }
251 ...
252 }
253 ...
254 }
255 ...
256 }
257 ...
258 }
259 ...
260 }
261 ...
262 }
263 ...
264 }
265 ...
266 }
267 ...
268 }
269 ...
270 }
271 ...
272 }
273 ...
274 }
275 ...
276 }
277 ...
278 }
279 ...
280 }
281 ...
282 }
283 ...
284 }
285 ...
286 }
287 ...
288 }
289 ...
290 }
291 ...
292 }
293 ...
294 }
295 ...
296 }
297 ...
298 }
299 ...
299 }
300 ...
301 ...
302 ...
303 ...
304 ...
305 ...
306 ...
307 ...
308 ...
309 ...
310 ...
311 ...
312 ...
313 ...
314 ...
315 ...
316 ...
317 ...
318 ...
319 ...
320 ...
321 ...
322 ...
323 ...
324 ...
325 ...
326 ...
327 ...
328 ...
329 ...
330 ...
331 ...
332 ...
333 ...
334 ...
335 ...
336 ...
337 ...
338 ...
339 ...
340 ...
341 ...
342 ...
343 ...
344 ...
345 ...
346 ...
347 ...
348 ...
349 ...
350 ...
351 ...
352 ...
353 ...
354 ...
355 ...
356 ...
357 ...
358 ...
359 ...
360 ...
361 ...
362 ...
363 ...
364 ...
365 ...
366 ...
367 ...
368 ...
369 ...
370 ...
371 ...
372 ...
373 ...
374 ...
375 ...
376 ...
377 ...
378 ...
379 ...
380 ...
381 ...
382 ...
383 ...
384 ...
385 ...
386 ...
387 ...
388 ...
389 ...
390 ...
391 ...
392 ...
393 ...
394 ...
395 ...
396 ...
397 ...
398 ...
399 ...
399 }
400 ...
401 ...
402 ...
403 ...
404 ...
405 ...
406 ...
407 ...
408 ...
409 ...
410 ...
411 ...
412 ...
413 ...
414 ...
415 ...
416 ...
417 ...
418 ...
419 ...
420 ...
421 ...
422 ...
423 ...
424 ...
425 ...
426 ...
427 ...
428 ...
429 ...
430 ...
431 ...
432 ...
433 ...
434 ...
435 ...
436 ...
437 ...
438 ...
439 ...
439 }
440 ...
441 ...
442 ...
443 ...
444 ...
445 ...
446 ...
447 ...
448 ...
449 ...
450 ...
451 ...
452 ...
453 ...
454 ...
455 ...
456 ...
457 ...
458 ...
459 ...
459 }
460 ...
461 ...
462 ...
463 ...
464 ...
465 ...
466 ...
467 ...
468 ...
469 ...
469 }
470 ...
471 ...
472 ...
473 ...
474 ...
475 ...
476 ...
477 ...
478 ...
479 ...
479 }
480 ...
481 ...
482 ...
483 ...
484 ...
485 ...
486 ...
487 ...
488 ...
489 ...
489 }
490 ...
491 ...
492 ...
493 ...
494 ...
495 ...
496 ...
497 ...
498 ...
499 ...
499 }
500 ...
501 ...
502 ...
503 ...
504 ...
505 ...
506 ...
507 ...
508 ...
509 ...
509 }
510 ...
511 ...
512 ...
513 ...
514 ...
515 ...
516 ...
517 ...
518 ...
519 ...
519 }
520 ...
521 ...
522 ...
523 ...
524 ...
525 ...
526 ...
527 ...
528 ...
529 ...
529 }
530 ...
531 ...
532 ...
533 ...
534 ...
535 ...
536 ...
537 ...
538 ...
539 ...
539 }
540 ...
541 ...
542 ...
543 ...
544 ...
545 ...
546 ...
547 ...
548 ...
549 ...
549 }
550 ...
551 ...
552 ...
553 ...
554 ...
555 ...
556 ...
557 ...
558 ...
559 ...
559 }
560 ...
561 ...
562 ...
563 ...
564 ...
565 ...
566 ...
567 ...
568 ...
569 ...
569 }
570 ...
571 ...
572 ...
573 ...
574 ...
575 ...
576 ...
577 ...
578 ...
579 ...
579 }
580 ...
581 ...
582 ...
583 ...
584 ...
585 ...
586 ...
587 ...
588 ...
589 ...
589 }
590 ...
591 ...
592 ...
593 ...
594 ...
595 ...
596 ...
597 ...
598 ...
599 ...
599 }
599 }
600 ...
601 ...
602 ...
603 ...
604 ...
605 ...
606 ...
607 ...
608 ...
609 ...
609 }
610 ...
611 ...
612 ...
613 ...
614 ...
615 ...
616 ...
617 ...
618 ...
619 ...
619 }
620 ...
621 ...
622 ...
623 ...
624 ...
625 ...
626 ...
627 ...
628 ...
629 ...
629 }
630 ...
631 ...
632 ...
633 ...
634 ...
635 ...
636 ...
637 ...
638 ...
639 ...
639 }
640 ...
641 ...
642 ...
643 ...
644 ...
645 ...
646 ...
647 ...
648 ...
649 ...
649 }
650 ...
651 ...
652 ...
653 ...
654 ...
655 ...
656 ...
657 ...
658 ...
659 ...
659 }
660 ...
661 ...
662 ...
663 ...
664 ...
665 ...
666 ...
667 ...
668 ...
669 ...
669 }
670 ...
671 ...
672 ...
673 ...
674 ...
675 ...
676 ...
677 ...
678 ...
679 ...
679 }
680 ...
681 ...
682 ...
683 ...
684 ...
685 ...
686 ...
687 ...
688 ...
689 ...
689 }
690 ...
691 ...
692 ...
693 ...
694 ...
695 ...
696 ...
697 ...
698 ...
699 ...
699 }
699 }
700 ...
701 ...
702 ...
703 ...
704 ...
705 ...
706 ...
707 ...
708 ...
709 ...
709 }
710 ...
711 ...
712 ...
713 ...
714 ...
715 ...
716 ...
717 ...
718 ...
719 ...
719 }
720 ...
721 ...
722 ...
723 ...
724 ...
725 ...
726 ...
727 ...
728 ...
729 ...
729 }
729 }
730 ...
731 ...
732 ...
733 ...
734 ...
735 ...
736 ...
737 ...
738 ...
739 ...
739 }
739 }
740 ...
741 ...
742 ...
743 ...
744 ...
745 ...
746 ...
747 ...
748 ...
749 ...
749 }
749 }
750 ...
751 ...
752 ...
753 ...
754 ...
755 ...
756 ...
757 ...
758 ...
759 ...
759 }
759 }
760 ...
761 ...
762 ...
763 ...
764 ...
765 ...
766 ...
767 ...
768 ...
769 ...
769 }
769 }
770 ...
771 ...
772 ...
773 ...
774 ...
775 ...
776 ...
777 ...
778 ...
779 ...
779 }
779 }
780 ...
781 ...
782 ...
783 ...
784 ...
785 ...
786 ...
787 ...
788 ...
789 ...
789 }
789 }
790 ...
791 ...
792 ...
793 ...
794 ...
795 ...
796 ...
797 ...
798 ...
799 ...
799 }
799 }
800 ...
801 ...
802 ...
803 ...
804 ...
805 ...
806 ...
807 ...
808 ...
809 ...
809 }
809 }
810 ...
811 ...
812 ...
813 ...
814 ...
815 ...
816 ...
817 ...
818 ...
819 ...
819 }
819 }
820 ...
821 ...
822 ...
823 ...
824 ...
825 ...
826 ...
827 ...
828 ...
829 ...
829 }
829 }
830 ...
831 ...
832 ...
833 ...
834 ...
835 ...
836 ...
837 ...
838 ...
839 ...
839 }
839 }
840 ...
841 ...
842 ...
843 ...
844 ...
845 ...
846 ...
847 ...
848 ...
849 ...
849 }
849 }
850 ...
851 ...
852 ...
853 ...
854 ...
855 ...
856 ...
857 ...
858 ...
859 ...
859 }
859 }
860 ...
861 ...
862 ...
863 ...
864 ...
865 ...
866 ...
867 ...
868 ...
869 ...
869 }
869 }
870 ...
871 ...
872 ...
873 ...
874 ...
875 ...
876 ...
877 ...
878 ...
879 ...
879 }
879 }
880 ...
881 ...
882 ...
883 ...
884 ...
885 ...
886 ...
887 ...
888 ...
889 ...
889 }
889 }
890 ...
891 ...
892 ...
893 ...
894 ...
895 ...
896 ...
897 ...
898 ...
899 ...
899 }
899 }
900 ...
901 ...
902 ...
903 ...
904 ...
905 ...
906 ...
907 ...
908 ...
909 ...
909 }
909 }
910 ...
911 ...
912 ...
913 ...
914 ...
915 ...
916 ...
917 ...
918 ...
919 ...
919 }
919 }
920 ...
921 ...
922 ...
923 ...
924 ...
925 ...
926 ...
927 ...
928 ...
929 ...
929 }
929 }
930 ...
931 ...
932 ...
933 ...
934 ...
935 ...
936 ...
937 ...
938 ...
939 ...
939 }
939 }
940 ...
941 ...
942 ...
943 ...
944 ...
945 ...
946 ...
947 ...
948 ...
949 ...
949 }
949 }
950 ...
951 ...
952 ...
953 ...
954 ...
955 ...
956 ...
957 ...
958 ...
959 ...
959 }
959 }
960 ...
961 ...
962 ...
963 ...
964 ...
965 ...
966 ...
967 ...
968 ...
969 ...
969 }
969 }
970 ...
971 ...
972 ...
973 ...
974 ...
975 ...
976 ...
977 ...
978 ...
979 ...
979 }
979 }
980 ...
981 ...
982 ...
983 ...
984 ...
985 ...
986 ...
987 ...
988 ...
989 ...
989 }
989 }
990 ...
991 ...
992 ...
993 ...
994 ...
995 ...
996 ...
997 ...
998 ...
999 ...
999 }
999 }
1000 ...
1001 ...
1002 ...
1003 ...
1004 ...
1005 ...
1006 ...
1007 ...
1008 ...
1009 ...
1009 }
1009 }
1010 ...
1011 ...
1012 ...
1013 ...
1014 ...
1015 ...
1016 ...
1017 ...
1018 ...
1019 ...
1019 }
1019 }
1020 ...
1021 ...
1022 ...
1023 ...
1024 ...
1025 ...
1026 ...
1027 ...
1028 ...
1029 ...
1029 }
1029 }
1030 ...
1031 ...
1032 ...
1033 ...
1034 ...
1035 ...
1036 ...
1037 ...
1038 ...
1039 ...
1039 }
1039 }
1040 ...
1041 ...
1042 ...
1043 ...
1044 ...
1045 ...
1046 ...
1047 ...
1048 ...
1049 ...
1049 }
1049 }
1050 ...
1051 ...
1052 ...
1053 ...
1054 ...
1055 ...
1056 ...
1057 ...
1058 ...
1059 ...
1059 }
1059 }
1060 ...
1061 ...
1062 ...
1063 ...
1064 ...
1065 ...
1066 ...
1067 ...
1068 ...
1069 ...
1069 }
1069 }
1070 ...
1071 ...
1072 ...
1073 ...
1074 ...
1075 ...
1076 ...
1077 ...
1078 ...
1079 ...
1079 }
1079 }
1080 ...
1081 ...
1082 ...
1083 ...
1084 ...
1085 ...
1086 ...
1087 ...
1088 ...
1089 ...
1089 }
1089 }
1090 ...
1091 ...
1092 ...
1093 ...
1094 ...
1095 ...
1096 ...
1097 ...
1098 ...
1099 ...
1099 }
1099 }
1100 ...
1101 ...
1102 ...
1103 ...
1104 ...
1105 ...
1106 ...
1107 ...
1108 ...
1109 ...
1109 }
1109 }
1110 ...
1111 ...
1112 ...
1113 ...
1114 ...
1115 ...
1116 ...
1117 ...
1118 ...
1119 ...
1119 }
1119 }
1120 ...
1121 ...
1122 ...
1123 ...
1124 ...
1125 ...
1126 ...
1127 ...
1128 ...
1129 ...
1129 }
1129 }
1130 ...
1131 ...
1132 ...
1133 ...
1134 ...
1135 ...
1136 ...
1137 ...
1138 ...
1139 ...
1139 }
1139 }
1140 ...
1141 ...
1142 ...
1143 ...
1144 ...
1145 ...
1146 ...
1147 ...
1148 ...
1149 ...
1149 }
1149 }
1150 ...
1151 ...
1152 ...
1153 ...
1154 ...
1155 ...
1156 ...
1157 ...
1158 ...
1159 ...
1159 }
1159 }
1160 ...
1161 ...
1162 ...
1163 ...
1164 ...
1165 ...
1166 ...
1167 ...
1168 ...
1169 ...
1169 }
1169 }
1170 ...
1171 ...
1172 ...
1173 ...
1174 ...
1175 ...
1176 ...
1177 ...
1178 ...
1179 ...
1179 }
1179 }
1180 ...
1181 ...
1182 ...
1183 ...
1184 ...
1185 ...
1186 ...
1187 ...
1188 ...
1189 ...
1189 }
1189 }
1190 ...
1191 ...
1192 ...
1193 ...
1194 ...
1195 ...
1196 ...
1197 ...
1198 ...
1199 ...
1199 }
1199 }
1200 ...
1201 ...
1202 ...
1203 ...
1204 ...
1205 ...
1206 ...
1207 ...
1208 ...
1209 ...
1209 }
1209 }
1210 ...
1211 ...
1212 ...
1213 ...
1214 ...
1215 ...
1216 ...
1217 ...
1218 ...
1219 ...
1219 }
1219 }
1220 ...
1221 ...
1222 ...
1223 ...
1224 ...
1225 ...
1226 ...
1227 ...
1228 ...
1229 ...
1229 }
1229 }
1230 ...
1231 ...
1232 ...
1233 ...
1234 ...
1235 ...
1236 ...
1237 ...
1238 ...
1239 ...
1239 }
1239 }
1240 ...
1241 ...
1242 ...
1243 ...
1244 ...
1245 ...
1246 ...
1247 ...
1248 ...
1249 ...
1249 }
1249 }
1250 ...
1251 ...
1252 ...
1253 ...
1254 ...
1255 ...
1256 ...
1257 ...
1258 ...
1259 ...
1259 }
1259 }
1260 ...
1261 ...
1262 ...
1263 ...
1264 ...
1265 ...
1266 ...
1267 ...
1268 ...
1269 ...
1269 }
1269 }
1270 ...
1271 ...
1272 ...
1273 ...
1274 ...
1275 ...
1276 ...
1277 ...
1278 ...
1279 ...
1279 }
1279 }
1280 ...
1281 ...
1282 ...
1283 ...
1284 ...
1285 ...
1286 ...
1287 ...
1288 ...
1289 ...
1289 }
1289 }
1290 ...
1291 ...
1292 ...
1293 ...
1294 ...
1295 ...
1296 ...
1297 ...
1298 ...
1299 ...
1299 }
1299 }
1300 ...
1301 ...
1302 ...
1303 ...
1304 ...
1305 ...
1306 ...
1307 ...
1308 ...
1309 ...
1309 }
1309 }
1310 ...
1311 ...
1312 ...
1313 ...
1314 ...
1315 ...
1316 ...
1317 ...
1318 ...
1319 ...
1319 }
1319 }
1320 ...
1321 ...
1322 ...
1323 ...
1324 ...
1325 ...
1326 ...
1327 ...
1328 ...
1329 ...
1329 }
1329 }
1330 ...
1331 ...
1332 ...
1333 ...
1334 ...
1335 ...
1336 ...
1337 ...
1338 ...
1339 ...
1339 }
1339 }
1340 ...
1341 ...
1342 ...
1343 ...
1344 ...
1345 ...
1346 ...
1347 ...
1348 ...
1349 ...
1349 }
1349 }
1350 ...
1351 ...
1352 ...
1353 ...
1354 ...
1355 ...
1356 ...
1357 ...
1358 ...
1359 ...
1359 }
1359 }
1360 ...
1361 ...
1362 ...
1363 ...
1364 ...
1365 ...
1366 ...
1367 ...
1368 ...
1369 ...
1369 }
1369 }
1370 ...
1371 ...
1372 ...
1373 ...
1374 ...
1375 ...
1376 ...
1377 ...
1378 ...
1379 ...
1379 }
1379 }
1380 ...
1381 ...
1382 ...
1383 ...
1384 ...
1385 ...
1386 ...
1387 ...
1388 ...
1389 ...
1389 }
1389 }
1390 ...
1391 ...
1392 ...
1393 ...
1394 ...
1395 ...
1396 ...
1397 ...
1398 ...
1399 ...
1399 }
1399 }
1400 ...
1401 ...
1402 ...
1403 ...
1404 ...
1405 ...
1406 ...
1407 ...
1408 ...
1409 ...
1409 }
1409 }
1410 ...
1411 ...
1412 ...
1413 ...
1414 ...
1415 ...
1416 ...
1417 ...
1418 ...
1419 ...
1419 }
1419 }
1420 ...
1421 ...
1422 ...
1423 ...
1424 ...
1425 ...
1426 ...
1427 ...
1428 ...
1429 ...
1429 }
1429 }
1430 ...
1431 ...
1432 ...
1433 ...
1434 ...
1435 ...
1436 ...
1437 ...
1438 ...
1439 ...
1439 }
1439 }
1440 ...
1441 ...
1442 ...
1443 ...
1444 ...
1445 ...
1446 ...
1447 ...
1448 ...
1449 ...
1449 }
1449 }
1450 ...
1451 ...
1452 ...
1453 ...
1454 ...
1455 ...
1456 ...
1457 ...
1458 ...
1459 ...
1459 }
1459 }
1460 ...
1461 ...
1462 ...
1463 ...
1464 ...
1465 ...
1466 ...
1467 ...
1468 ...
1469 ...
1469 }
1469 }
1470 ...
1471 ...
1472 ...
1473 ...
1474 ...
1475 ...
1476 ...
1477 ...
1478 ...
1479 ...
1479 }
1479 }
1480 ...
1481 ...
1482 ...
1483 ...
1484 ...
1485 ...
1486 ...
1487 ...
1488 ...
1489 ...
1489 }
1489 }
1490 ...
1491 ...
1492 ...
1493 ...
1494 ...
1495 ...
1496 ...
1497 ...
1498 ...
1499 ...
1499 }
1499 }
1500 ...
1501 ...
1502 ...
1503 ...
1504 ...
1505 ...
1506 ...
1507 ...
1508 ...
1509 ...
1509 }
1509 }
1510 ...
1511 ...
1512 ...
1513 ...
1514 ...
1515 ...
1516 ...
1517 ...
1518 ...
1519 ...
1519 }
1519 }
1520 ...
1521 ...
1522 ...
1523 ...
1524 ...
1525 ...
1526 ...
1527 ...
1528 ...
1529 ...
1529 }
1529 }
1530 ...
1531 ...
1532 ...
1533 ...
1534 ...
1535 ...
1536 ...
1537 ...
1538 ...
1539 ...
1539 }
1539 }
1540 ...
1541 ...
1542 ...
1543 ...
1544 ...
1545 ...
1546 ...
1547 ...
1548 ...
1549 ...
1549 }
1549 }
1550 ...
1551 ...
1552 ...
1553 ...
1554 ...
1555 ...
1556 ...
1557 ...
1558 ...
1559 ...
1559 }
1559 }
1560 ...
1561 ...
1562 ...
1563 ...
1564 ...
1565 ...
1566 ...
1567 ...
1568 ...
1569 ...
1569 }
1569 }
1570 ...
1571 ...
1572 ...
1573 ...
1574 ...
1575 ...
1576 ...
1577 ...
1578 ...
1579 ...
1579 }
1579 }
1580 ...
1581 ...
1582 ...
1583 ...
1584 ...
1585 ...
1586 ...
1587 ...
1588 ...
1589 ...
1589 }
1589 }
1590 ...
1591 ...
1592 ...
1593 ...
1594 ...
1595 ...
1596 ...
1597 ...
1598 ...
1599 ...
1599 }
1599 }
1600 ...
1601 ...
1602 ...
1603 ...
1604 ...
1605 ...
1606 ...
1607 ...
1608 ...
1609 ...
1609 }
1609 }
1610 ...
1611 ...
1612 ...
1613 ...
1614 ...
1615 ...
1616 ...
1617 ...
1618 ...
1619 ...
1619 }
1619 }
1620 ...
1621 ...
1622 ...
1623 ...
1624 ...
1625 ...
1626 ...
1627 ...
1628 ...
1629 ...
1629 }
1629 }
1630 ...
1631 ...
1632 ...
1633 ...
1634 ...
1635 ...
1636 ...
1637 ...
1638 ...
1639 ...
1639 }
1639 }
1640 ...
1641 ...
1642 ...
1643 ...
1644 ...
1645 ...
1646 ...
1647 ...
1648 ...
1649 ...
1649 }
1649 }
1650 ...
1651 ...
1652 ...
1653 ...
1654 ...
1655 ...
1656 ...
1657 ...
1658 ...
1659 ...
1659 }
1659 }
1660 ...
1661 ...
1662 ...
1663 ...
1664 ...
1665 ...
1666 ...
1667 ...
1668 ...
1669 ...
1669 }
1669 }
1670 ...
1671 ...
1672 ...
1673 ...
1674 ...
1675 ...
1676 ...
1677 ...
1678 ...
1679 ...
1679 }
1679 }
1680 ...
1681 ...
1682 ...
1683 ...
1684 ...
1685 ...
1686 ...
1687 ...
1688 ...
1689 ...
1689 }
1689 }
1690 ...
1691 ...
1692 ...
1693 ...
1694 ...
1695 ...
1696 ...
1697 ...
1698 ...
1699 ...
1699 }
1699 }
1700 ...
1701 ...
1702 ...
1703 ...
1704 ...
1705 ...
1706 ...
1707 ...
1708 ...
1709 ...
1709 }
1709 }
1710 ...
1711 ...
1712 ...
1713 ...
1714 ...
1715 ...
1716 ...
1717 ...
1718 ...
1719 ...
1719 }
1719 }
1720 ...
1721 ...
1722 ...
1723 ...
1724 ...
1725 ...
1726 ...
1727 ...
1728 ...
1729 ...
1729 }
1729 }
1730 ...
1731 ...
1732 ...
1733 ...
1734 ...
1735 ...
1736 ...
1737 ...
1738 ...
1739 ...
1739 }
1739 }
1740 ...
1741 ...
1742 ...
1743 ...
1744 ...
1745 ...
1746 ...
1747 ...
1748 ...
1749 ...
1749 }
1749 }
1750 ...
1751 ...
1752 ...
1753 ...
1754 ...
1755 ...
1756 ...
1757 ...
1758 ...
1759 ...
1759 }
1759 }
1760 ...
1761 ...
1762 ...
1763 ...
1764 ...
1765 ...
1766 ...
1767 ...
1768 ...
1769 ...
1769 }
1769 }
1770 ...
1771 ...
1772 ...
1773 ...
1774 ...
1775 ...
1776 ...
1777 ...
1778 ...
1779 ...
1779 }
1779 }
1780 ...
1781 ...
1782 ...
1783 ...
1784 ...
1785 ...
1786 ...
1787 ...
1788 ...
1789 ...
1789 }
1789 }
1790 ...
1791 ...
1792 ...
1793 ...
1794 ...
1795 ...
1796 ...
1797 ...
1798 ...
1799 ...
1799 }
1799 }
1800 ...
1801 ...
1802 ...
1803 ...
1804 ...
1805 ...
1806 ...
1807 ...
1808 ...
1809 ...
1809 }
1809 }
1810 ...
1811 ...
1812 ...
1813 ...
1814 ...
1815 ...
1816 ...
1817 ...
1818 ...
1819 ...
1819 }
1819 }
1820 ...
1821 ...
1822 ...
1823 ...
1824 ...
1825 ...
1826 ...
1827 ...
1828 ...
1829 ...
1829 }
1829 }
1830 ...
1831 ...
1832 ...
1833 ...
1834 ...
1835 ...
1836 ...
1837 ...
1838 ...
1839 ...
1839 }
1839 }
1840 ...
1841 ...
1842 ...
1843 ...
1844 ...
1845 ...
1846 ...
1847 ...
1848 ...
1849 ...
1849 }
1849 }
1850 ...
1851 ...
1852 ...
1853 ...
1854 ...
1855 ...
1856 ...
1857 ...
1858 ...
1859 ...
1859 }
1859 }
1860 ...
1861 ...
1862 ...
1863 ...
1864 ...
1865 ...
1866 ...
1867 ...
1868 ...
1869 ...
1869 }
1869 }
1870 ...
1871 ...
1872 ...
1873 ...
1874 ...
1875 ...
1876 ...
1877 ...
1878 ...
1879 ...
1879 }
1879 }
1880 ...
1881 ...
1882 ...
1883 ...
1884 ...
1885 ...
1886 ...
1887 ...
1888 ...
1889 ...
1889 }
1889 }
1890 ...
1891 ...
1892 ...
1893 ...
1894 ...
1895 ...
1896 ...
1897 ...
1898 ...
1899 ...
1899 }
1899 }
1900 ...
1901 ...
1902 ...
1903 ...
1904 ...
1905 ...
1906 ...
1907 ...
1908 ...
1909 ...
1909 }
1909 }
1910 ...
1911 ...
1912 ...
1913 ...
1914 ...
1915 ...
1916 ...
1917 ...
1918 ...
1919 ...
1919 }
1919 }
1920 ...
1921 ...
1922 ...
1923 ...
1924 ...
1925 ...
1926 ...
1927 ...
1928 ...
1929 ...
1929 }
1929 }
1930 ...
1931 ...
1932 ...
1933 ...
1934 ...
1935 ...
1936 ...
1937 ...
1938 ...
1939 ...
1939 }
1939 }
1940 ...
1941 ...
1942 ...
1943 ...
1944 ...
1945 ...
1946 ...
1947 ...
1948 ...
1949 ...
1949 }
1949 }
1950 ...
1951 ...
1952 ...
1953 ...
1954 ...
1955 ...
1956 ...
1957 ...
1958 ...
1959 ...
1959 }
1959 }
1960 ...
1961 ...
1962 ...
1963 ...
1964 ...
1965 ...
1966 ...
1967 ...
1968 ...
1969 ...
1969 }
1969 }
1970 ...
1971 ...
1972 ...
1973 ...
1974 ...
1975 ...
1976 ...
1977 ...
1978 ...
1979 ...
1979 }
1979 }
1980 ...
1981 ...
1982 ...
1983 ...
1984 ...
1985 ...
1986 ...
1987 ...
1988 ...
1989 ...
1989 }
1989 }
1990 ...
1991 ...
1992 ...
1993 ...
1994 ...
1995 ...
1996 ...
1997 ...
1998 ...
1999 ...
1999 }
1999 }
2000 ...
2001 ...
2002 ...
2003 ...
2004 ...
2005 ...
2006 ...
2007 ...
2008 ...
2009 ...
2009 }
2009 }
2010 ...
2011 ...
2012 ...
2013 ...
2014 ...
2015 ...
2016 ...
2017 ...
2018 ...
2019 ...
2019 }
2019 }
2020 ...
2021 ...
2022 ...
2023 ...
2024 ...
2025 ...
2026 ...
2027 ...
2028 ...
2029 ...
2029 }
2029 }
2030 ...
2031 ...
2032 ...
2033 ...
2034 ...
2035 ...
2036 ...
2037 ...
2038 ...
2039 ...
2039 }
2039 }
2040 ...
2041 ...
2042 ...
2043 ...
2044 ...
2045 ...
2046 ...
2047 ...
2048 ...
2049 ...
2049 }
2049 }
2050 ...
2051 ...
2052 ...
2053 ...
2054 ...
2055 ...
2056 ...
2057 ...
2058 ...
2059 ...
2059 }
2059 }
2060 ...
2061 ...
2062 ...
2063 ...
2064 ...
2065 ...
2066 ...
2067 ...
2068 ...
2069 ...
2069 }
2069 }
2070 ...
2071 ...
2072 ...
2073 ...
2074 ...
2075 ...
2076 ...
2077 ...
2078 ...
2079 ...
2079 }
2079 }
2080 ...
2081 ...
2082 ...
2083 ...
2084 ...
2085 ...
2086 ...
2087 ...
2088 ...
2089 ...
2089 }
2089 }
2090 ...
2091 ...
2092 ...
2093 ...
2094 ...
2095 ...
2
```

```

6 import org.openqa.selenium.WebDriver;
7 import org.openqa.selenium.WebDriverException;
8 import org.openqa.selenium.ie.InternetExplorerDriver;
9
10 import com.ebay.maui.webdriver.WebDriverConfig;
11 import com.ebay.maui.helper.ThreadHelper;
12 import com.ebay.maui.helper.ThreadHelperper;
13
14 public class IEDriverFactory extends AbstractWebDriverFactory implements IWebDriverFactory{
15
16     public IEDriverFactory(WebDriverConfig cfg) {
17         super(cfg);
18     }
19
20     @Override
21     public void cleanUp() {
22         try {
23             if (driver != null) {
24                 try {
25                     driver.quit();
26                 } catch (WebDriverException ex) {
27                     ex.printStackTrace();
28                 }
29             }
30             driver = null;
31         } catch (Exception ex) {
32             {
33                 //Ignore all exceptions
34             }
35         }
36     }
37
38     @Override
39     public WebDriver createWebDriver() throws IOException {
40         killprocess();
41         ThreadHelper.waitForSeconds();
42         if(!OSHelper.isWindows())
43         {
44             throw new RuntimeException("IE can only run in windows!");
45         }
46         WebDriverConfig cfg = this.getWebDriverConfig();
47
48         driver = new InternetExplorerDriver(new ICapabilitiesFactory().createCapabilities(cfg));
49
50         //Implicit Waits to handle dynamic element. The default value is 5 seconds.
51         setImplicitWaitTimeout(cfg.getImplicitWaitTimeout());
52         if(cfg.getPageLoadTimeout()>0)
53         {
54             driver.manage().timeouts().pageLoadTimeout(cfg.getPageLoadTimeout(), TimeUnit.SECONDS);
55         }
56
57         this.setWebDriver(driver);
58         return driver;
59     }
60
61     private void killprocess(){
62         //So please make sure only one test case run at a time
63         if(OSHelper.isWindows())
64         {
65             try {
66                 Runtime.getRuntime().exec("taskkill /F /IM IEDriverServer.exe");
67                 Runtime.getRuntime().exec("taskkill /F /IM Iexplore.exe");
68             } catch (IOException e) {
69             }
70         }
71     }
72 }

```

Project Explorer

- com.ebay.maui.driver.web
  - com.ebay.maui.driver.web.element
    - BaseHtmlPage.class
    - BasePageObject.class
    - Button.class
    - CheckBox.class
    - Form.class
    - GenericWebPage.class
    - HtmlElement.class
    - Image.class
    - Page.class
    - Label.class
    - Link.class
    - LocatorSwitch.class
    - LocatorType.class
    - RadioButton.class
    - SelectList.class
    - Table.class
    - TextField.class
    - WebPage.class
    - WebPageSection.class
- com.ebay.maui.driver.web.factory
  - AbstractWebDriverFactory.class
  - AndroidCapabilitiesFactory.class
  - ChromeCapabilitiesFactory.class
  - ChromeDriverFactory.class
  - FirefoxCapabilitiesFactory.class
  - FirefoxDriverFactory.class
  - HTMLUnitCapabilitiesFactory.class
  - OperaDriverFactory.class
  - PhantomJSCapabilitiesFactory.class
  - RemoteDriverFactory.class
  - SafariCapabilitiesFactory.class
  - SafariDriverFactory.class
- com.ebay.maui.exception
  - APIException.class
  - DataCreationException.class

IECapabilitiesFactory.class

IWebDriverFactory.class

IWebDriverFactory

- cleanUp() : void
- createWebDriver() : WebDriver
- getWebDriver() : WebDriver
- getWebDriverConfig() : WebDriverConfig

Members calling constructors of 'WebPageSection' - in workspace

- WebPageSection(String, By) - com.ebay.maui.driver.web.element.WebPageSection
- WebPageSection(String, String) - com.ebay.maui.driver.web.element.WebPageSection
- WebPageSection(String) - com.ebay.maui.driver.web.element.WebPageSection

The screenshot shows an IDE interface with the following details:

- Title Bar:** Quick Access, Java EE, Debug.
- Project Explorer:** Shows files like IECapabilitiesFactory.class, IDEDriverFactory.class, IWebDriverFactory.class, and OperaCapabilitiesFactory.class.
- Code Editor:** Displays the `OperaCapabilitiesFactory` class. The code implements `ICapabilitiesFactory` and overrides the `createCapabilities` method. It handles various configuration parameters such as proxy, browser version, platform, resolution, and context manager settings.
- Status Bar:** Read-Only, Smart Insert, 1:1.

```
1 package com.ebay.maui.driver.web.factory;
2
3 import org.openqa.selenium.Proxy;
4 import org.openqa.selenium.remote.CapabilityType;
5 import org.openqa.selenium.remote.DesiredCapabilities;
6
7 import com.ebay.maui.controller.ContextManager;
8 import com.ebay.maui.driver.web.WebDriverConfig;
9 import com.ebay.ql.autobot.listeners.DeDashListener;
10
11 public class OperaCapabilitiesFactory implements ICapabilitiesFactory {
12
13     public DesiredCapabilities createCapabilities(WebDriverConfig cfg) {
14         DesiredCapabilities capability = DesiredCapabilities.operalink();
15
16         if (cfg.isEnableJavascript())
17             capability.setJavascriptEnabled(true);
18         else
19             capability.setJavascriptEnabled(false);
20         capability.setCapability(CapabilityType.TAKES_SCREENSHOT, true);
21         capability.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);
22
23         if (cfg.getBrowserVersion() != null)
24             capability.setVersion(cfg.getBrowserVersion());
25
26         if (cfg.getPlatform() != null)
27             capability.setPlatform(cfg.getPlatform());
28
29         if (cfg.getResolution() != null)
30             capability.setCapability("resolution", cfg.getResolution());
31
32         if (cfg.getProxyHost() != null) {
33             Proxy proxy = cfg.getProxy();
34             capability.setCapability(CapabilityType.PROXY, proxy);
35
36         if (ContextManager.getContext().isDeDashListenerEnabled())
37             DeDashListener.setCapabilityMetrics(capability);
38
39         return capability;
40     }
41
42 }
43
44
45 }
```

The screenshot shows an IDE interface with the following details:

- Title Bar:** Quick Access, Java EE, Debug.
- Project Explorer:** Shows files like IECapabilitiesFactory.class, IDEDriverFactory.class, IWebDriverFactory.class, OperaCapabilitiesFactory.class, and OperaDriverFactory.class.
- Code Editor:** Displays the `OperaDriverFactory` class. It extends `AbstractWebDriverFactory` and implements `IWebDriverFactory`. The `createWebDriver` method creates a new `OperaDriver` instance using the `OperaCapabilitiesFactory` to set capabilities. It also handles local mode and external resources extraction.
- Status Bar:** Read-Only, Smart Insert, 30 : 29.

```
1 package com.ebay.maui.driver.web.factory;
2
3 import com.ebay.maui.helper.FileHelper;
4
5 public class OperaDriverFactory extends AbstractWebDriverFactory implements
6     IWebDriverFactory {
7
8     public static final String OPERA_DRIVER = OperaDriverService.OPERA_DRIVER_EXE_PROPERTY;
9
10    public OperaDriverFactory(WebDriverConfig cfg) {
11        super(cfg);
12    }
13
14    @Override
15    public WebDriver createWebDriver() throws IOException {
16        WebDriverConfig cfg = this.getWebDriverConfig();
17        getOperaDriver(cfg);
18
19        driver = new OperaDriver(
20            new OperaCapabilitiesFactory().createCapabilities(cfg));
21
22        setupImplicitWait(cfg.getImplicitWaitTimeout());
23        if (cfg.getPageLoadTimeout() >= 0) {
24            driver.manage()
25                .timeouts()
26                    .pageLoadTimeout(cfg.getPageLoadTimeout(), TimeUnit.SECONDS);
27
28        this.setWebDriver(driver);
29        return driver;
30    }
31
32    /**
33     * Get OperaDriver for local run
34     */
35    @Override
36    public void getOperaDriver(WebDriverConfig config) throws IOException {
37        if (config.getMode() == WebDriverMode.LocalOnly) {
38            String operaDriverPath = config.getOperaDriverPath();
39            if (operaDriverPath == null) {
40                try {
41                    if (System.getenv(OPERA_DRIVER) != null) {
42                        System.out.println("Get Opera driver from property:");
43                        System.getenv(OPERA_DRIVER);
44                        System.setProperty(OPERA_DRIVER);
45                        System.getenv(OPERA_DRIVER);
46                    } else {
47                        handleExtractResources();
48                    }
49                } catch (IOException ex) {
50                    ex.printStackTrace();
51                }
52            } else {
53                System.setProperty(OPERA_DRIVER, operaDriverPath);
54            }
55        }
56    }
57
58    public void handleExtractResources() throws IOException {
59        String dir = this.getClass().getResources(".").getPath()
60            + "maultemp";
61        dir = FileHelper.decodePath(dir);
62
63        if (!new File(dir).exists()) {
64            System.out.println("Handling Opera resources in " + dir);
65            FileHelper.extractJarDir( WebdriverExternalResources.class );
66        }
67    }
68
69
70 }
71
72
73    public void handleExtractResources() throws IOException {
74        String dir = this.getClass().getResources(".").getPath()
75            + "maultemp";
76        dir = FileHelper.decodePath(dir);
77
78        if (!new File(dir).exists()) {
79            System.out.println("Handling Opera resources in " + dir);
80            FileHelper.extractJarDir( WebdriverExternalResources.class );
81        }
82
83 }
```

```

public WebDriver createWebDriver() throws IOException {
    WebDriverConfig cfg = this.getWebDriverConfig();
    getOperaDriver(cfg);

    driver = new OperaDriver(
        new OperaCapabilitiesFactory().createCapabilities(cfg));
    setImplicitWaitTimeout(cfg.getImplicitWaitTimeout());
    if (cfg.getPageLoadTimeout() >= 0) {
        driver.manage()
            .timeouts()
            .pageLoadTimeout(cfg.getPageLoadTimeout(), TimeUnit.SECONDS);
    }
    this.setWebDriver(driver);
    return driver;
}

/**
 * Get OperaDriver for local run
 */
throws IOException
{
    public void getOperaDriver(WebDriverConfig config) throws IOException {
        if (config.getMode() == WebdriverMode._LOCALLY_ON_RK) {
            String operaDriverPath = config.getOperaDriverPath();
            if (operaDriverPath == null) {
                try {
                    if (System.getenv(OPERA_DRIVER) != null) {
                        System.out.println("Using system property: " + System.getenv(OPERA_DRIVER));
                        System.setProperty(OPERA_DRIVER, System.getenv(OPERA_DRIVER));
                    } else {
                        handleExternalResources();
                    }
                } catch (IOException ex) {
                    ex.printStackTrace();
                }
            } else {
                System.setProperty(OPERA_DRIVER, operaDriverPath);
            }
        }
    }

    public void handleExternalResources() throws IOException {
        String dir = this.getClass().getResource("/*").getPath();
        dir = FileHelper.decodePath(dir);

        if (!new File(dir).exists()) {
            System.out.println("Handling Opera resources in " + dir);
            FileHelper.extractJarDir(dir, WebdriverExternalResources.class);
        }

        if (new File(dir + OSHelper.getSlash() + "operadriver.exe").exists()) {
            FileHelper.extractJarDir(dir, WebdriverExternalResources.class);
        }

        if (OSHelper.isWindows()) {
            System.setProperty(OPERA_DRIVER, dir + "\\operadriver.exe");
        }
    }
}

```

```

public DesiredCapabilities createCapabilities(WebDriverConfig cfg) {
    DesiredCapabilities capability = DesiredCapabilities.phantomjs();
    if (cfg.isEnableJavaScript()) {
        capability.setJavaScriptEnabled(true);
    } else {
        capability.setJavaScriptEnabled(false);
    }
    capability.setCapability(CapabilityType.TAKES_SCREENSHOT, true);
    capability.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);

    if (cfg.getPlatform() != null) {
        capability.setPlatform(cfg.getPlatform());
    }

    if (cfg.getResolution() != null) {
        capability.setCapability("resolution", cfg.getResolution());
    }

    if (cfg.getProxyHost() != null) {
        Proxy proxy = cfg.getProxy();
        capability.setCapability(CapabilityType.PROXY, proxy);
    }

    if (ContextManager.getThreadContext().isBeDoshiListenerEnabled()) {
        BeDoshiListener.setCapabilityMetrics(capability);
    }
    return capability;
}

```

```
... IECapabilitiesFactory.class IDEDriverFactory.class IWebDriverFactory.class OperaCapabilitiesFactory.class OperaDriverFactory.class PhantomJSCapabilitiesFactory. RemoteDriverFactory.class ...
30
31     public class RemoteDriverFactory extends AbstractWebDriverFactory implements
32         IWebDriverFactory {
33
34     public RemoteDriverFactory(WebDriverConfig cfg) {
35         super(cfg);
36     }
37
38     @Override
39     public WebDriver createWebDriver() throws Exception {
40         WebDriverConfig cfg = this.getWebDriverConfig();
41         DesiredCapabilities capability = null;
42         URL url;
43
44         url = new URL(cfg.getHubUrl());
45
46         switch (cfg.getBrowser()) {
47             case FIREFOX:
48                 capability = new FirefoxCapabilitiesFactory()
49                     .createCapabilities(cfg);
50
51             case INTERNETEXPLORER:
52                 capability = new IECapabilitiesFactory().createCapabilities(cfg);
53
54             case CHROME:
55                 capability = new ChromeCapabilitiesFactory()
56                     .createCapabilities(cfg);
57
58             case HMLUNIT:
59                 capability = new HtmlUnitCapabilitiesFactory()
60                     .createCapabilities(cfg);
61
62             case SAFARI:
63                 capability = new SafariCapabilitiesFactory()
64                     .createCapabilities(cfg);
65
66             case ANDROID:
67                 capability = new AndroidCapabilitiesFactory()
68                     .createCapabilities(cfg);
69
70             case IPHONE:
71                 capability = ((ICapabilitiesFactory) Class
72                             .forName(
73                                 "com.ebay.maui.driver.web.factory.IPhoneCapabilitiesFactory")
74                             .getConstructor().newInstance()).createCapabilities(cfg);
75
76             case IPAD:
77                 capability = ((ICapabilitiesFactory) Class
78                             .forName(
79                                 "com.ebay.maui.driver.web.factory.IPadCapabilitiesFactory")
80                             .getConstructor().newInstance()).createCapabilities(cfg);
81
82             case OPERA:
83                 capability = new OperaCapabilitiesFactory().createCapabilities(cfg);
84
85             case PHANTOMJS:
86                 capability = new PhantomJSCapabilitiesFactory()
87                     .createCapabilities(cfg);
88
89             default:
90                 break;
91         }
92
93         switch (cfg.getBrowser()) {
94             case IPHONE:
95                 ...
96             case IPAD:
97                 driver = (WebDriver) Class
98                     .forName(
99                         "com.ebay.maui.driver.web.factory.RemoteiOSBaseDriver")
100                     .getConstructor(URL.class, DesiredCapabilities.class)
101                     .newInstance(url, capability);
102
103             case ANDROID:
104                 driver = new SelendroidDriver(url, capability);
105
106             default:
107                 try {
108                     driver = createRemoteWebDriver(url, capability);
109                 } catch (Exception ex) {
110                     Logging.log("Create remote WebDriver failed, retry after 5 seconds");
111                     ThreadHelper.waitForSeconds();
112                     driver = createRemoteWebDriver(url, capability);
113                 }
114
115             setImplicitWaitTimeout(cfg.getImplicitWaitTimeout());
116             if (cfg.getPageLoadTimeout() > 0) {
117                 setPageLoadTimeout(cfg.getPageLoadTimeout(), cfg.getBrowser());
118             }
119             this.setWebDriver(driver);
120             String hub = url.getHost();
121             int port = url.getPort();
122             // logging mode in address:
123             try {
124                 HttpHost host = new HttpHost(hub, port);
125                 HttpClient client = HttpClientBuilder.create().build();
126                 //DefaultHttpClient client = new DefaultHttpClient();
127                 String sessionUrl = "http://" + hub + ":" + port
128                     + "/grid/api/testsession/sessionId";
129                 URL session = new URL(sessionUrl);
130                 //constructing the request
131                 BasicHttpPut httpPut = new BasicHttpPut();
132                 httpPut.setURI(session);
133                 httpPut.setEntity(new StringEntity("{" + "sessionId": driver.getSessionId() + "}"));
134                 HttpResponse response = client.execute(host, httpPut);
135                 String responseString = EntityUtils.toString(response.getEntity());
136                 try {
137                     JSONObject object = new JSONObject(responseString);
138                     String proxyId = (String) object.get("proxyId");
139                     String node = (String) proxyId.split("-")[0];
140                     String browserName = ((RemoteWebDriver) driver)
141                         .getCapabilities().getBrowserName();
142                     String browserVersion = ((RemoteWebDriver) driver).getCapabilities()
143                         .getVersion();
144                     System.out.println("WebDriver is running on node " + node
145                         + " " + browserName + " version " + session
146                         + " (" + ((RemoteWebDriver) driver).getSessionId());
147                     Logging.log("WebDriver is running on node " + node + ", "
148                         + browserName + " version " + session +
149                         " (" + ((RemoteWebDriver) driver).getSessionId());
150                 } catch (org.json.JSONException e) {
151
152                 } catch (Exception ex) {
153
154             }
155
156             private WebDriver createRemoteWebDriver(URL url,
157                 DesiredCapabilities capability) throws InterruptedException,
```

```
... IECapabilitiesFactory.class IDEDriverFactory.class IWebDriverFactory.class OperaCapabilitiesFactory.class OperaDriverFactory.class PhantomJSCapabilitiesFactory. RemoteDriverFactory.class ...
94
95         case IPHONE:
96             driver = (WebDriver) Class
97                 .forName(
98                     "com.ebay.maui.driver.web.factory.RemoteiOSBaseDriver")
99                     .getConstructor(URL.class, DesiredCapabilities.class)
100                     .newInstance(url, capability);
101
102         break;
103
104         case ANDROID:
105             driver = new SelendroidDriver(url, capability);
106
107         default:
108             try {
109                 driver = createRemoteWebDriver(url, capability);
110             } catch (Exception ex) {
111                 Logging.log("Create remote WebDriver failed, retry after 5 seconds");
112                 ThreadHelper.waitForSeconds();
113                 driver = createRemoteWebDriver(url, capability);
114             }
115
116         setImplicitWaitTimeout(cfg.getImplicitWaitTimeout());
117         if (cfg.getPageLoadTimeout() > 0) {
118             setPageLoadTimeout(cfg.getPageLoadTimeout(), cfg.getBrowser());
119         }
120         this.setWebDriver(driver);
121         String hub = url.getHost();
122         int port = url.getPort();
123         // logging mode in address:
124         try {
125             HttpHost host = new HttpHost(hub, port);
126             HttpClient client = HttpClientBuilder.create().build();
127             //DefaultHttpClient client = new DefaultHttpClient();
128             String sessionUrl = "http://" + hub + ":" + port
129                 + "/grid/api/testsession/sessionId";
130             URL session = new URL(sessionUrl);
131             //constructing the request
132             BasicHttpPut httpPut = new BasicHttpPut();
133             httpPut.setURI(session);
134             httpPut.setEntity(new StringEntity("{" + "sessionId": driver.getSessionId() + "}"));
135             HttpResponse response = client.execute(host, httpPut);
136             String responseString = EntityUtils.toString(response.getEntity());
137             try {
138                 JSONObject object = new JSONObject(responseString);
139                 String proxyId = (String) object.get("proxyId");
140                 String node = (String) proxyId.split("-")[0];
141                 String browserName = ((RemoteWebDriver) driver)
142                     .getCapabilities().getBrowserName();
143                 String browserVersion = ((RemoteWebDriver) driver).getCapabilities()
144                     .getVersion();
145                 System.out.println("WebDriver is running on node " + node
146                     + " " + browserName + " version " + session
147                     + " (" + ((RemoteWebDriver) driver).getSessionId());
148                 Logging.log("WebDriver is running on node " + node + ", "
149                     + browserName + " version " + session +
150                     " (" + ((RemoteWebDriver) driver).getSessionId());
151             } catch (org.json.JSONException e) {
152
153             } catch (Exception ex) {
154
155         }
156
157         private WebDriver createRemoteWebDriver(URL url,
158             DesiredCapabilities capability) throws InterruptedException,
```

```

138     String proxyId = (String) object.get("proxyId");
139     String nodeIp = proxyId.split(":")[1];
140     String browserName = ((RemoteWebDriver) driver).getBrowserName();
141     String version = ((RemoteWebDriver) driver).getCapabilities().getVersion();
142     System.out.println("Webdriver is running on node " + nodeIp
143                         + " " + browserName + " " + version + " " + sessionId);
144     Logging.log("Webdriver is running on node " + nodeIp + ", "
145                 + browserName + " " + version + " " + sessionId);
146     Logging.log("Session ID: " + sessionId);
147     Logging.log("Driver: " + ((RemoteWebDriver) driver).getSessionId());
148 } catch (org.json.JSONException e) {
149 }
150 } catch (Exception ex) {
151 }
152 }
153 }
154 return driver;
155 }
156
157 private WebDriver createRemoteWebDriver(URL url,
158                                         DesiredCapabilities capability) throws InterruptedException,
159                                         ExecutionException, TimeoutException {
160     ExecutorService exec = Executors.newCachedThreadPool();
161     int timeout = 60 * 25;
162     URL start = url;
163     final DesiredCapabilities capability1 = capability;
164     Callable<WebDriver> task = new Callable<WebDriver>() {
165
166         public WebDriver call() throws Exception {
167             WebDriver driver = new ScreenShotRemoteWebDriver(start,
168                                                               capability1);
169             return driver;
170         }
171     };
172     Future<WebDriver> future = exec.submit(task);
173     WebDriver driver = null;
174     driver = future.get(timeout, TimeUnit.SECONDS);
175
176     return driver;
177 }
178
179 protected void setPageLoadTimeout(long timeout, BrowserType type) {
180     switch (type) {
181     case Chrome:
182         try {
183             driver.manage().timeouts()
184                 .pageLoadTimeout(timeout, TimeUnit.SECONDS);
185         } catch (UnsupportedCommandException e) {
186             // chromedriver does not support pageLoadTimeout
187         }
188         break;
189     case Firefox:
190     case InternetExplorer:
191         driver.manage().timeouts()
192             .pageLoadTimeout(timeout, TimeUnit.SECONDS);
193         break;
194     default:
195         // Safari: java.lang.RuntimeException:
196         // org.openqa.selenium.WebDriverException: Unknown command:
197         // settimeout (WARNING: The server did not provide any stacktrace
198         // information)
199     }
200 }
201 }
202

```

Read-Only Smart Insert 1:1 ...

```

1 package com.ebay.mau.driver.web.factory;
2
3 import org.openqa.selenium.Proxy;
4 import org.openqa.selenium.remote.DesiredCapabilities;
5 import org.openqa.selenium.remote.RemoteDesiredCapabilities;
6 import org.openqa.selenium.safari.SafariOptions;
7
8 import com.ebay.mau.controller.ContextManager;
9 import com.ebay.mau.driver.web.WebDriverConfig;
10 import com.ebay.qa.autobot.listeners.BedashListener;
11
12 public class SafariCapabilitiesFactory implements ICapabilitiesFactory {
13
14     public DesiredCapabilities createCapabilities(WebDriverConfig cfg) {
15         DesiredCapabilities capability = DesiredCapabilities.safari();
16
17         SafariOptions safariOptions = new SafariOptions();
18         safariOptions.setTakesScreenshot(true);
19         capability.setCapability(SafariOptions.CAPABILITY, safariOptions);
20
21         if (cfg.isEnableJavascript()) {
22             capability.setJavascriptEnabled(true);
23         } else {
24             capability.setJavascriptEnabled(false);
25         }
26         capability.setCapability(CapabilityType.TAKES_SCREENSHOT, true);
27         capability.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);
28
29         if (cfg.getPowerVersion() != null) {
30             capability.setVersion(cfg.getBrowserVersion());
31         }
32
33         if (cfg.getPlatform() != null) {
34             capability.setPlatform(cfg.getPlatform());
35         }
36
37         if (cfg.getResolution() != null) {
38             capability.setCapability("resolution", cfg.getResolution());
39         }
40
41         if (cfg.getProxyHost() != null) {
42             Proxy proxy = cfg.getProxy();
43             capability.setCapability(CapabilityType.PROXY, proxy);
44         }
45
46         if (ContextManager.getThreadContext().isBedashListenerEnabled()) {
47             BedashListener.setCapabilityMetrics(capability);
48         }
49
50     }
51
52 }
53

```

Read-Only Smart Insert 1:1 ...

Screenshot of an IDE (likely Eclipse) showing the code editor and project structure.

**Project Explorer:**

- com.ebay.mau.driver.web.element
  - BaseHtmlPage.class
  - BasePageObject.class
  - Button.class
  - CheckBox.class
  - Form.class
  - GenericWebPage.class
  - HtmlElement.class
  - Image.class
  - Page.class
  - Label.class
  - Link.class
  - LocatorSwitch.class
  - LocatorType.class
  - RadioButton.class
  - SelectList.class
  - Table.class
  - TextField.class
  - WebPage.class
  - WebPageSection.class
  - WebPageSection.class
- com.ebay.mau.driver.web.factory
  - AbstractWebDriverFactory.class
  - AndroidCapabilitiesFactory.class
  - ChromeCapabilitiesFactory.class
  - ChromeDriverFactory.class
  - FirefoxCapabilitiesFactory.class
  - FirefoxDriverFactory.class
  - HTMLUnitCapabilitiesFactory.class
  - HTMLUnitDriverFactory.class
  - CapabilitiesFactory.class
  - ICapabilitiesFactory.class
  - IDriverFactory.class
  - WebDriverFactory.class
  - OperaCapabilitiesFactory.class
  - OperaDriverFactory.class
  - PhantomJSCapabilitiesFactory.class
  - RemoteDriverFactory.class
  - SafariCapabilitiesFactory.class
  - SafariDriverFactory.class
- com.ebay.mau.exception
  - APIException.class
  - DataCreationException.class

**Code Editor:**

```

1 package com.ebay.mau.driver.web.factory;
2
3 import org.openqa.selenium.WebDriver;
4 import org.openqa.selenium.remote.DesiredCapabilities;
5 import org.openqa.selenium.safari.SafariDriver;
6
7 import com.ebay.mau.controller.Logging;
8 import com.ebay.mau.driver.web.WebDriverConfig;
9
10 public class SafariDriverFactory extends AbstractWebDriverFactory implements IWebDriverFactory{
11
12     public SafariDriverFactory(@WebDriverConfig cfg) {
13         super(cfg);
14     }
15
16
17     @Override
18     public WebDriver createWebDriver() {
19         //WebDriverConfig cfg = this.getWebDriverConfig();
20
21         DesiredCapabilities cap = new SafariCapabilitiesFactory().createCapabilities(cfg);
22
23         System.out.println("start safari...");
```

**Members calling constructors of 'WebPageSection' - in workspace**

- WebPageSection(String, By) - com.ebay.mau.driver.web.element.WebPageSection
- WebPageSection(String, String) - com.ebay.mau.driver.web.element.WebPageSection
- WebPageSection(String) - com.ebay.mau.driver.web.element.WebPageSection