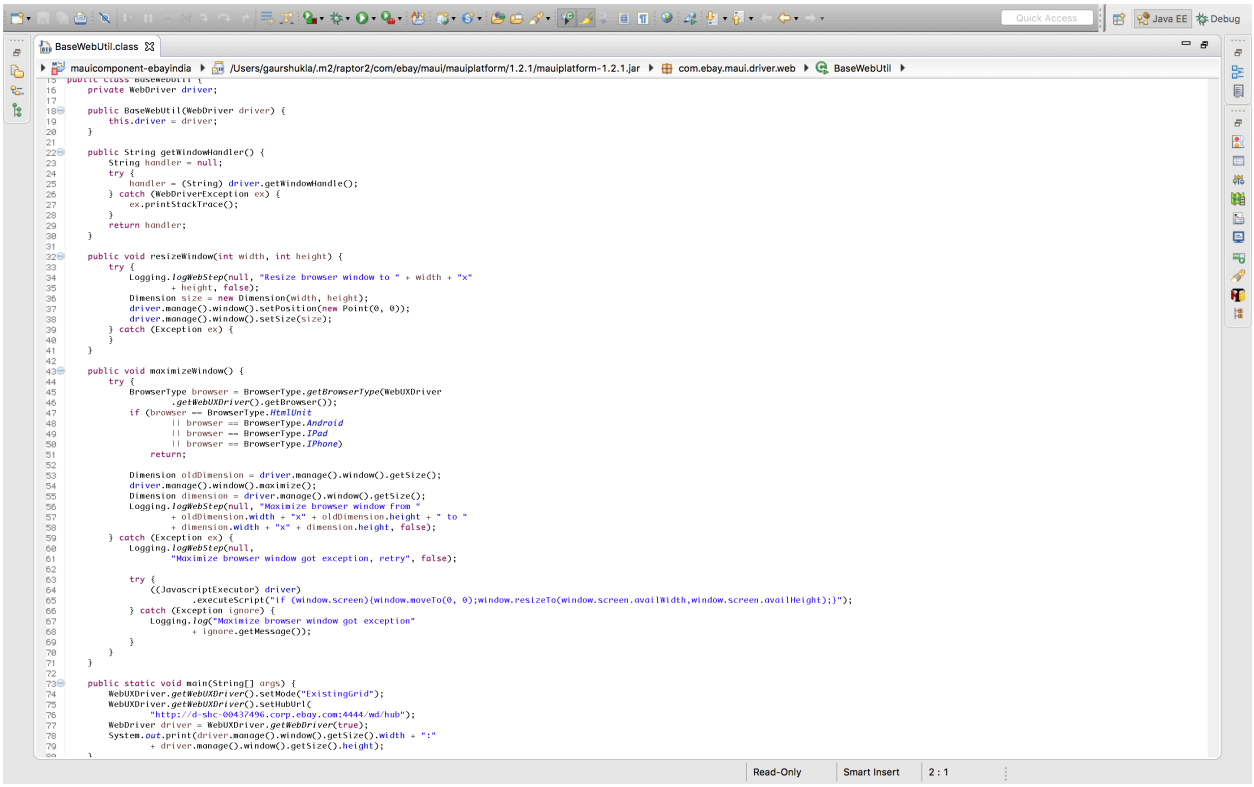
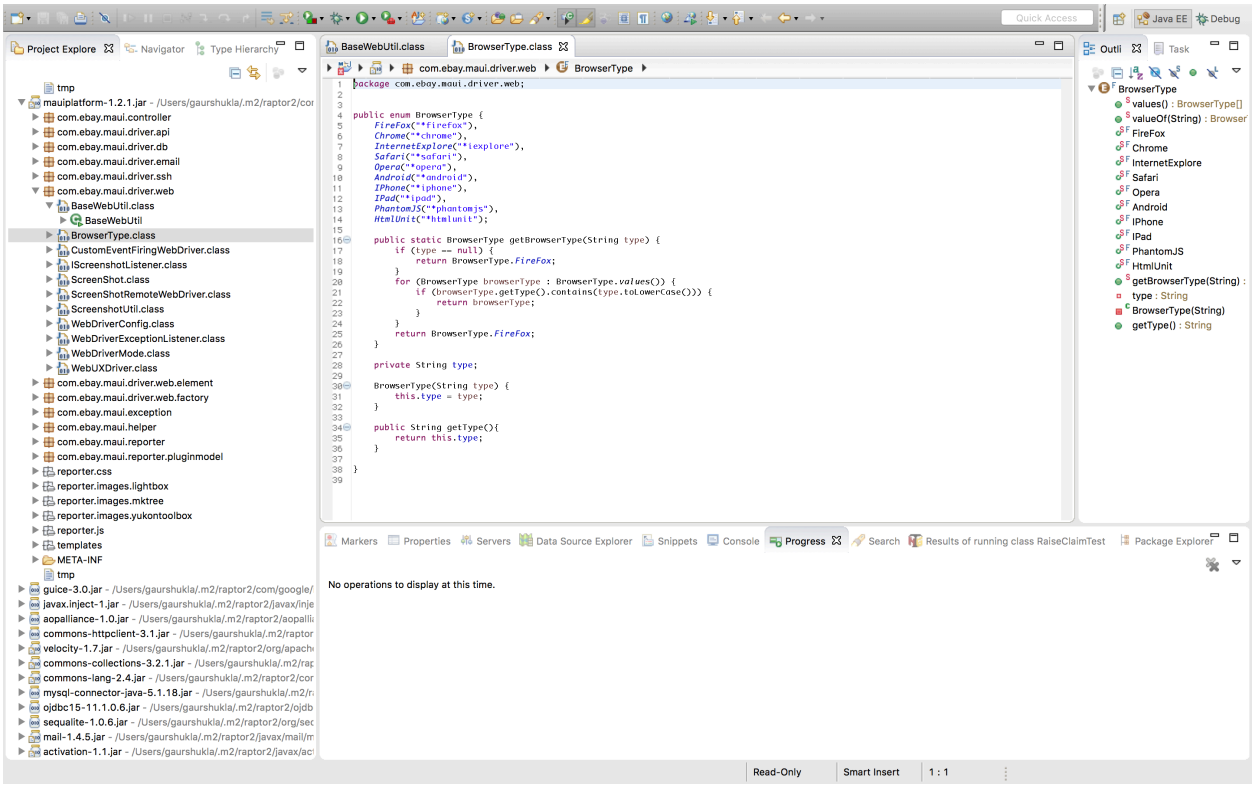
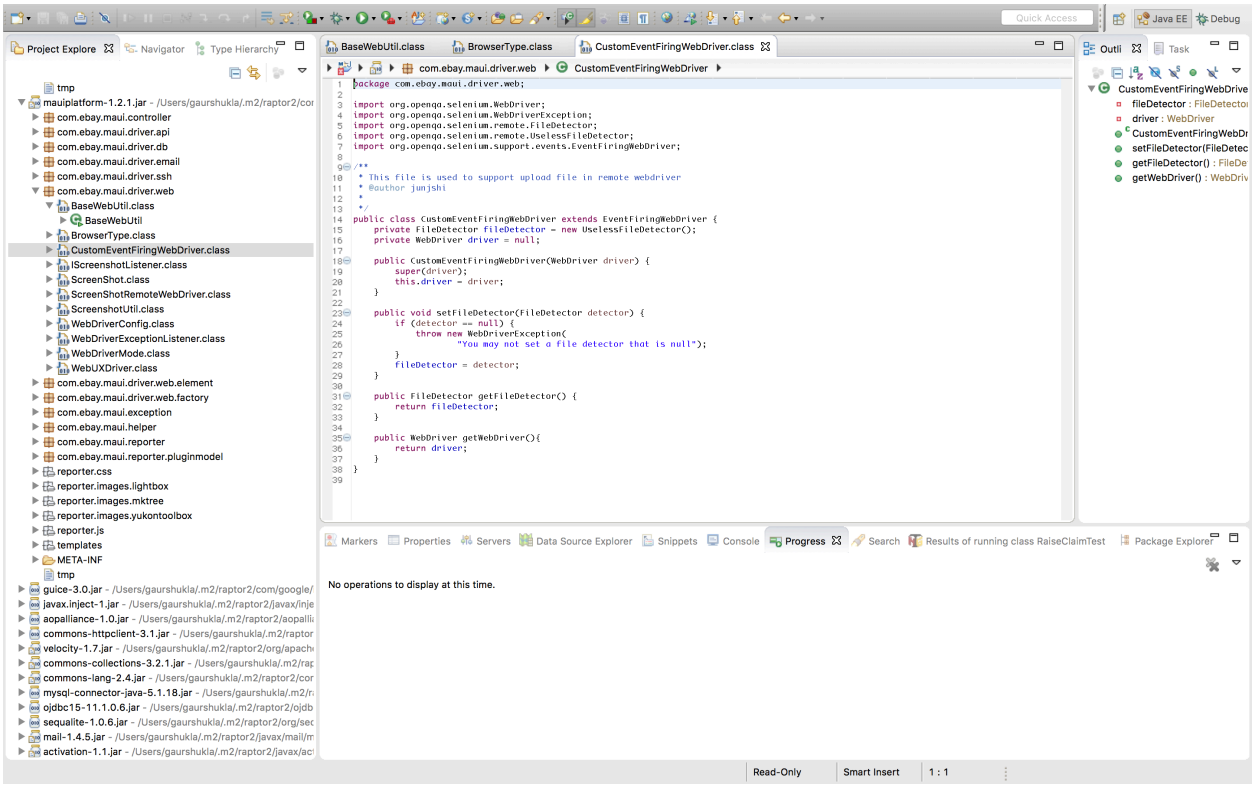
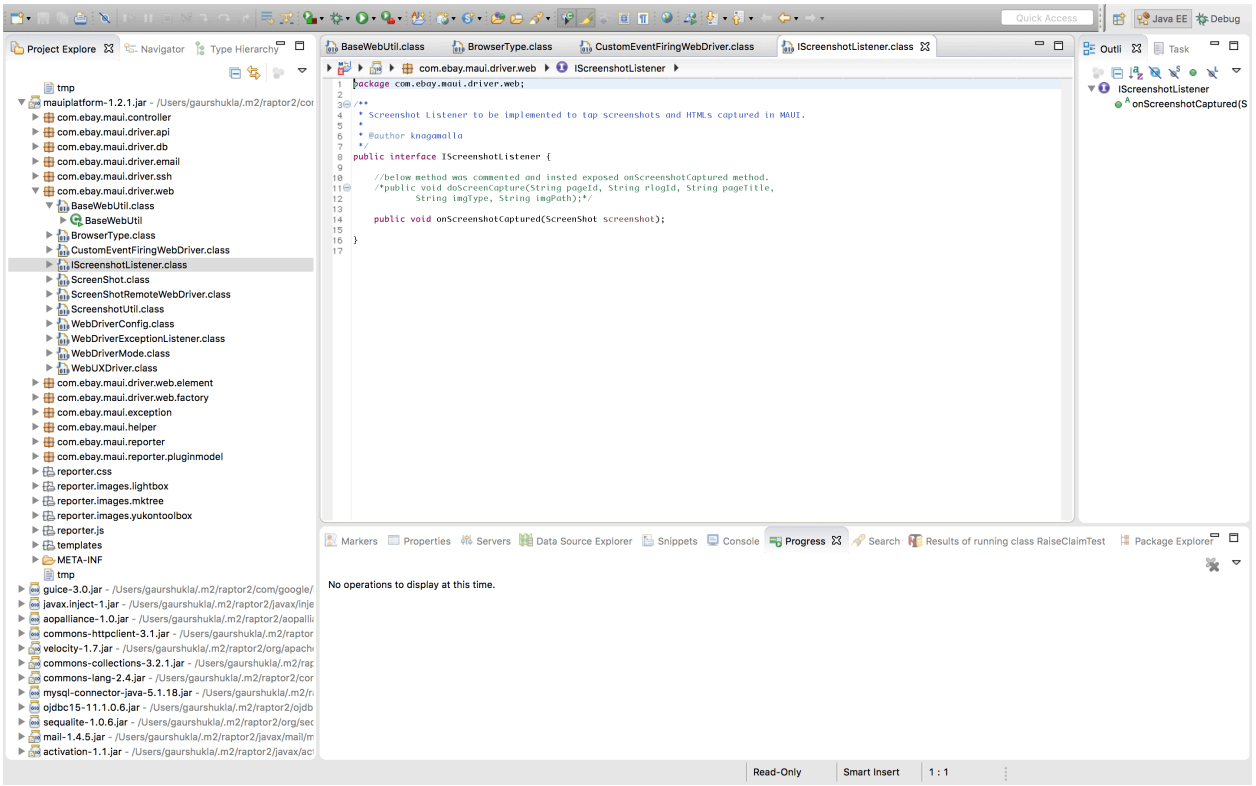
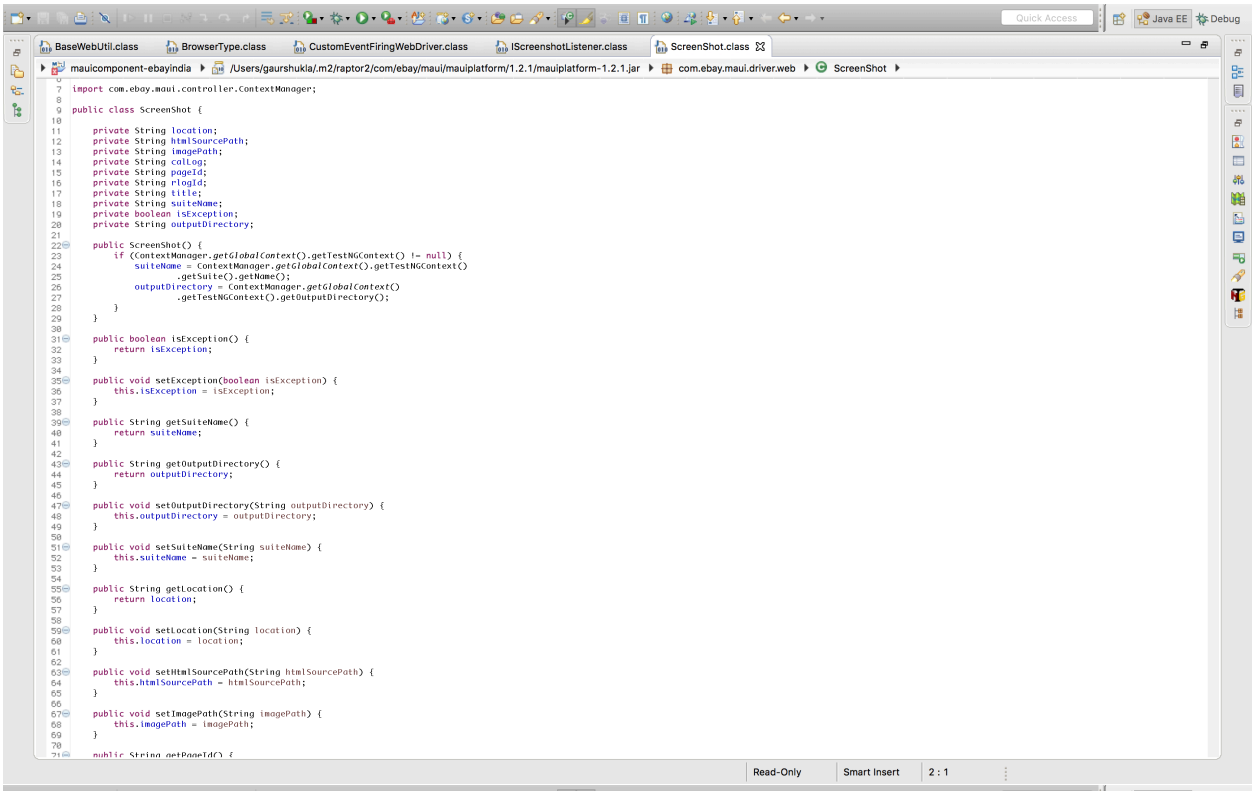
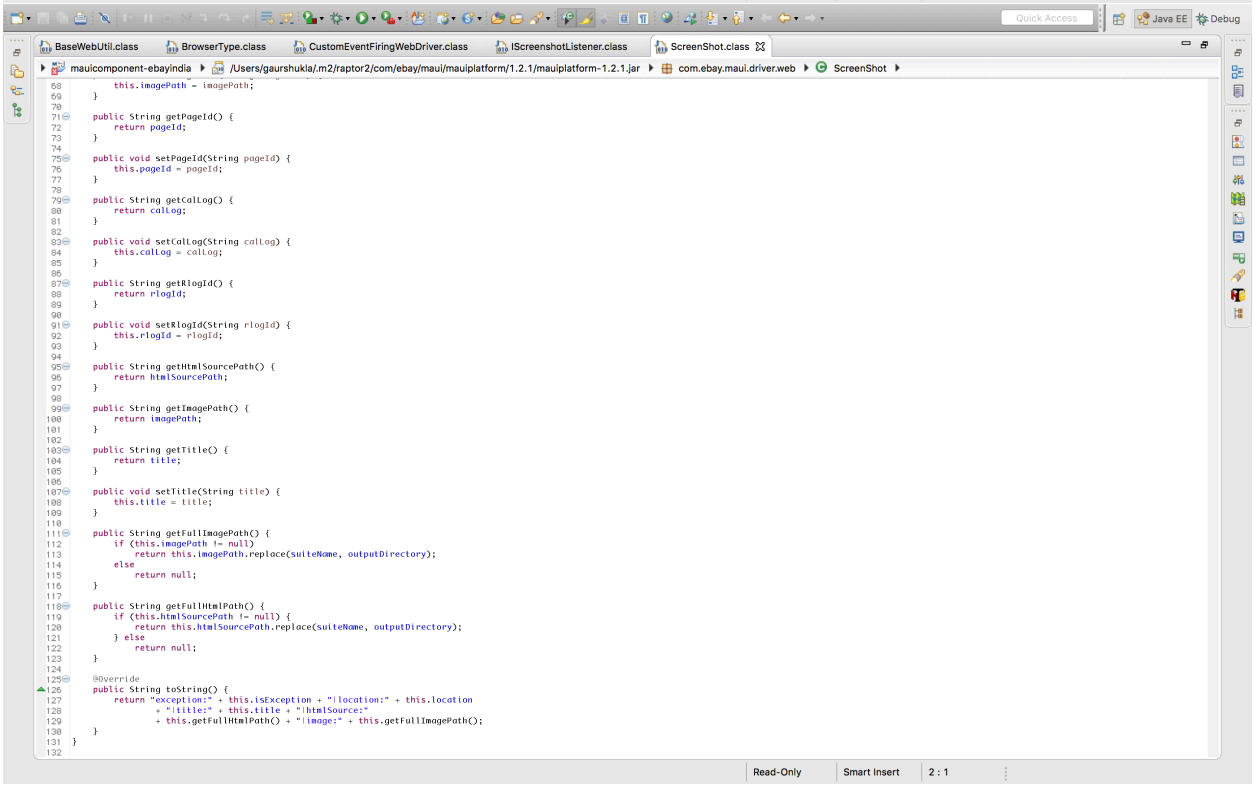

BaseWebUtil.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
BaseWebUtil
Read-Only
Smart Insert
2 : 1

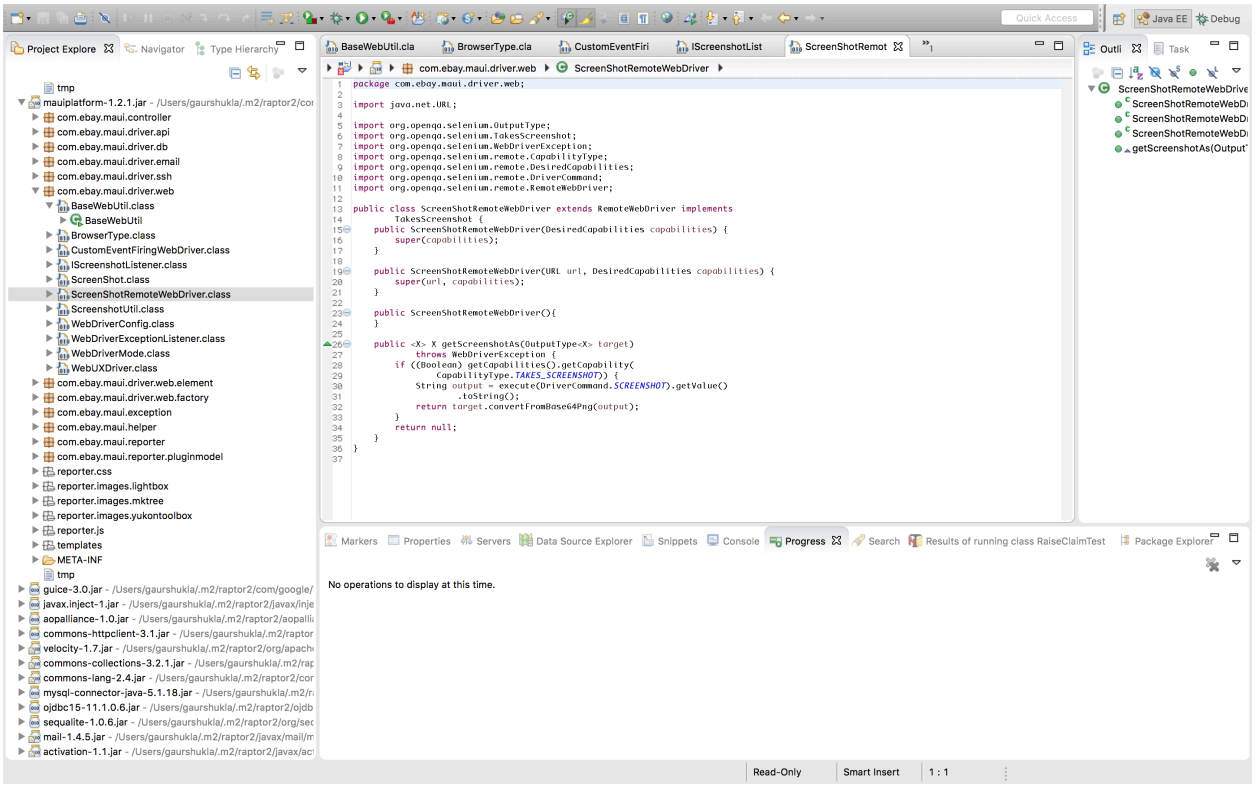

Project Explore
BaseWebUtil.class
BrowserType.class
com.ebay.maui.driver.web
BrowserType
Markers
Properties
Servers
Data Source Explorer
Snippets
Console
Progress
Search
Package Explorer
No operations to display at this time.
Read-Only
Smart Insert
1 : 1


BaseWebUtil.class
BrowserType.class
CustomEventFiringWebDriver.class
com.ebay.maui.driver.web
CustomEventFiringWebDriver
package com.ebay.maui.driver.web;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebDriverException;
import org.openqa.selenium.remote.FileDetector;
import org.openqa.selenium.remote.UselessFileDetector;
import org.openqa.selenium.support.events.EventFiringWebDriver;
/**
* This file is used to support upload file in remote webdriver
* @author junjshi
*
*/
public class CustomEventFiringWebDriver extends EventFiringWebDriver {
private FileDetector fileDetector = new UselessFileDetector();
private WebDriver driver = null;
public CustomEventFiringWebDriver(WebDriver driver) {
super(driver);
this.driver = driver;
}
public void setFileDetector(FileDetector detector) {
if (detector == null) {
throw new WebDriverException(
"You may not set a file detector that is null");
}
fileDetector = detector;
}
public FileDetector getFileDetector() {
return fileDetector;
}
public WebDriver getWebDriver(){
return driver;
}
}
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

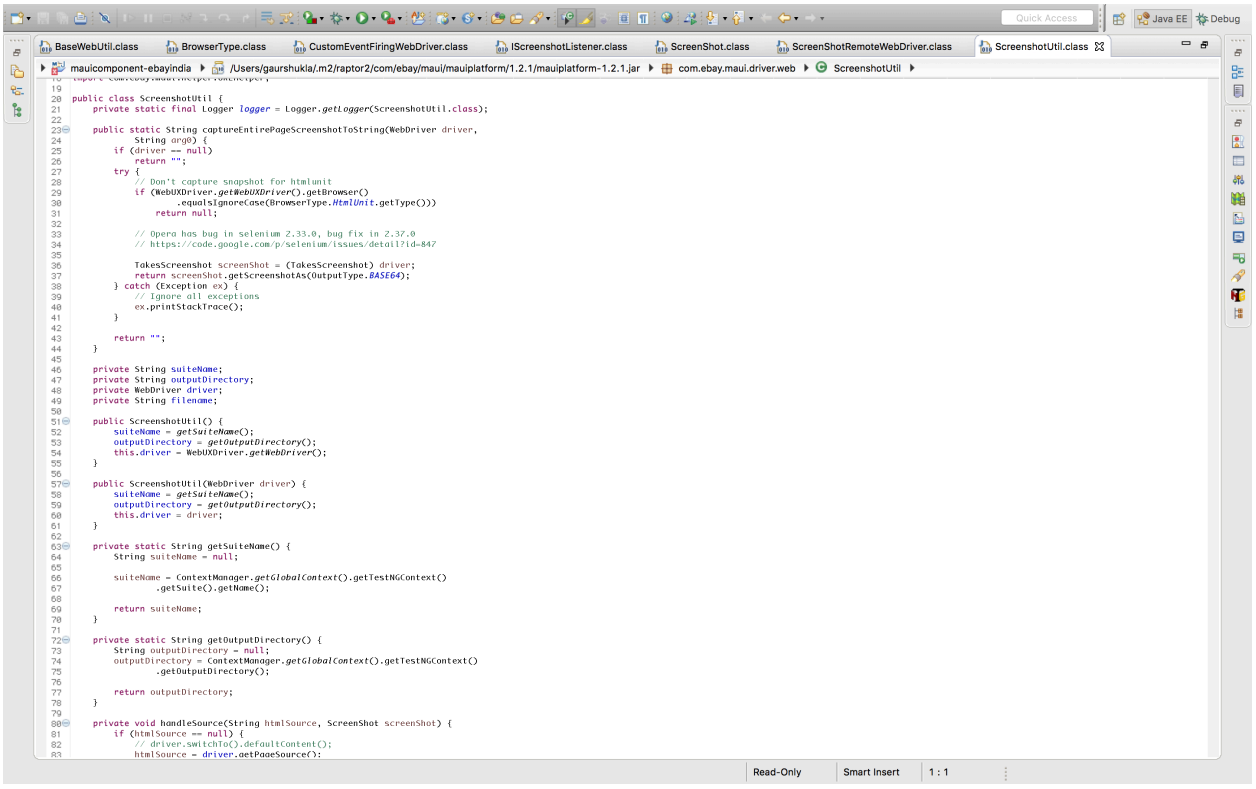

BaseWebUtil.class
BrowserType.class
CustomEventFiringWebDriver.class
IScreenshotListener.class
com.ebay.maui.driver.web
IScreenshotListener
package com.ebay.maui.driver.web;
/**
* Screenshot Listener to be implemented to tap screenshots and HTMLs captured in MAUI.
*
* @author knagamalla
*/
public interface IScreenshotListener {
//below method was commented and insted exposed onScreenshotCaptured method.
/*public void doScreenCapture(String pageId, String rlogId, String pageTitle,
String imgType, String imgPath);*/
public void onScreenshotCaptured(ScreenShot screenshot);
}
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


Quick Access
Java EE
Debug
BaseWebUtil.class
BrowserType.class
CustomEventFiringWebDriver.class
IScreenshotListener.class
ScreenShot.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
ScreenShot
import com.ebay.maui.controller.ContextManager;
public class ScreenShot {
private String location;
private String htmlSourcePath;
private String imagePath;
private String callog;
private String pageId;
private String rlogId;
private String title;
private String suiteName;
private boolean isException;
private String outputDirectory;
public ScreenShot() {
if (ContextManager.getGlobalContext().getTestNGContext() != null) {
suiteName = ContextManager.getGlobalContext().getTestNGContext()
.getSuite().getName();
outputDirectory = ContextManager.getGlobalContext()
.getTestNGContext().getOutputDirectory();
}
}
public boolean isException() {
return isException;
}
public void setException(boolean isException) {
this.isException = isException;
}
public String getSuiteName() {
return suiteName;
}
public String getOutputDirectory() {
return outputDirectory;
}
public void setOutputDirectory(String outputDirectory) {
this.outputDirectory = outputDirectory;
}
public void setSuiteName(String suiteName) {
this.suiteName = suiteName;
}
public String getLocation() {
return location;
}
public void setLocation(String location) {
this.location = location;
}
public void setHtmlSourcePath(String htmlSourcePath) {
this.htmlSourcePath = htmlSourcePath;
}
public void setImagePath(String imagePath) {
this.imagePath = imagePath;
}
Read-Only
Smart Insert
2 : 1

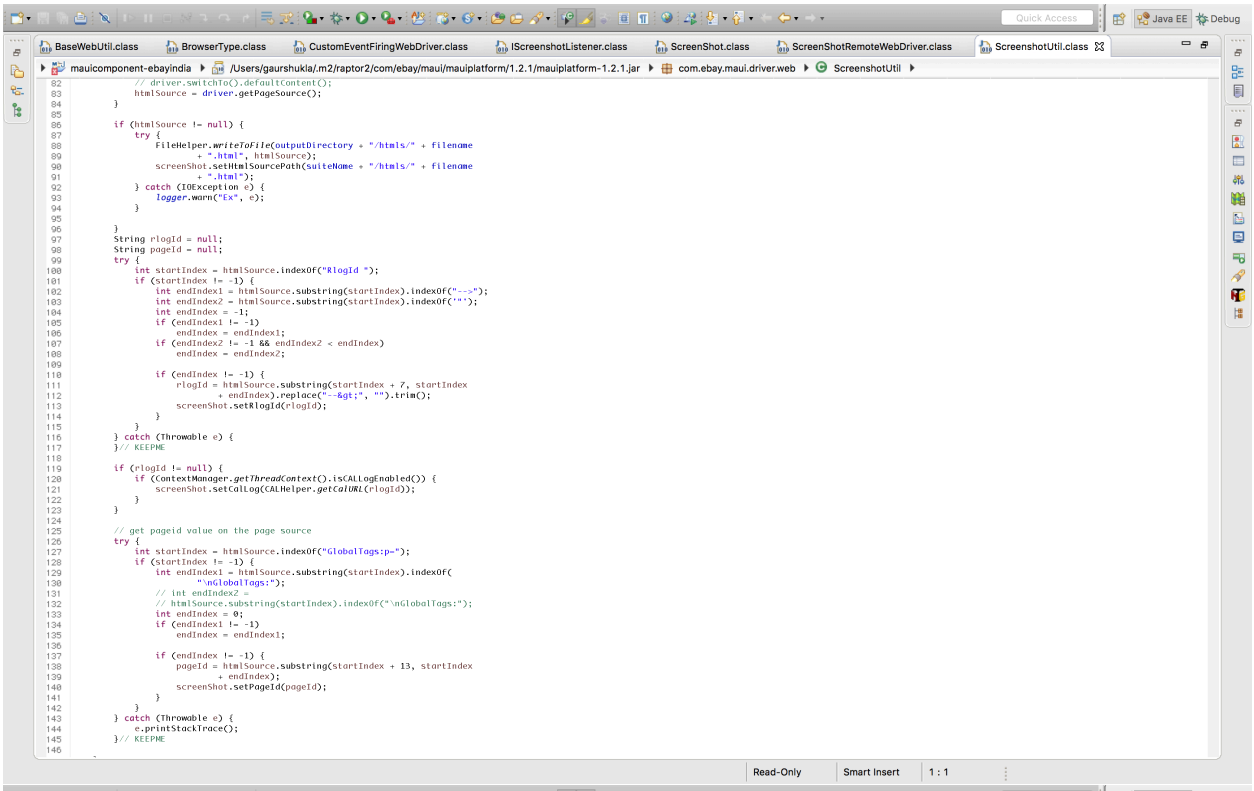

Quick Access
Java EE
Debug
BaseWebUtil.class
BrowserType.class
CustomEventFiringWebDriver.class
IScreenshotListener.class
ScreenShot.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
ScreenShot
this.imagePath = imagePath;
}
public String getPageId() {
return pageId;
}
public void setPageId(String pageId) {
this.pageId = pageId;
}
public String getCalLog() {
return callog;
}
public void setCalLog(String callog) {
this.callog = callog;
}
public String getRlogId() {
return rlogId;
}
public void setRlogId(String rlogId) {
this.rlogId = rlogId;
}
public String getHtmlSourcePath() {
return htmlSourcePath;
}
public String getImagePath() {
return imagePath;
}
public String getTitle() {
return title;
}
public void setTitle(String title) {
this.title = title;
}
public String getFullImagePath() {
if (this.imagePath != null)
return this.imagePath.replace(suiteName, outputDirectory);
else
return null;
}
public String getFullHtmlPath() {
if (this.htmlSourcePath != null) {
return this.htmlSourcePath.replace(suiteName, outputDirectory);
} else
return null;
}
@Override
public String toString() {
return "exception:" + this.isException + "|location:" + this.location
+ "|title:" + this.title + "|htmlSource:"
+ this.getFullHtmlPath() + "|image:" + this.getFullImagePath();
}
}
Read-Only
Smart Insert
2 : 1

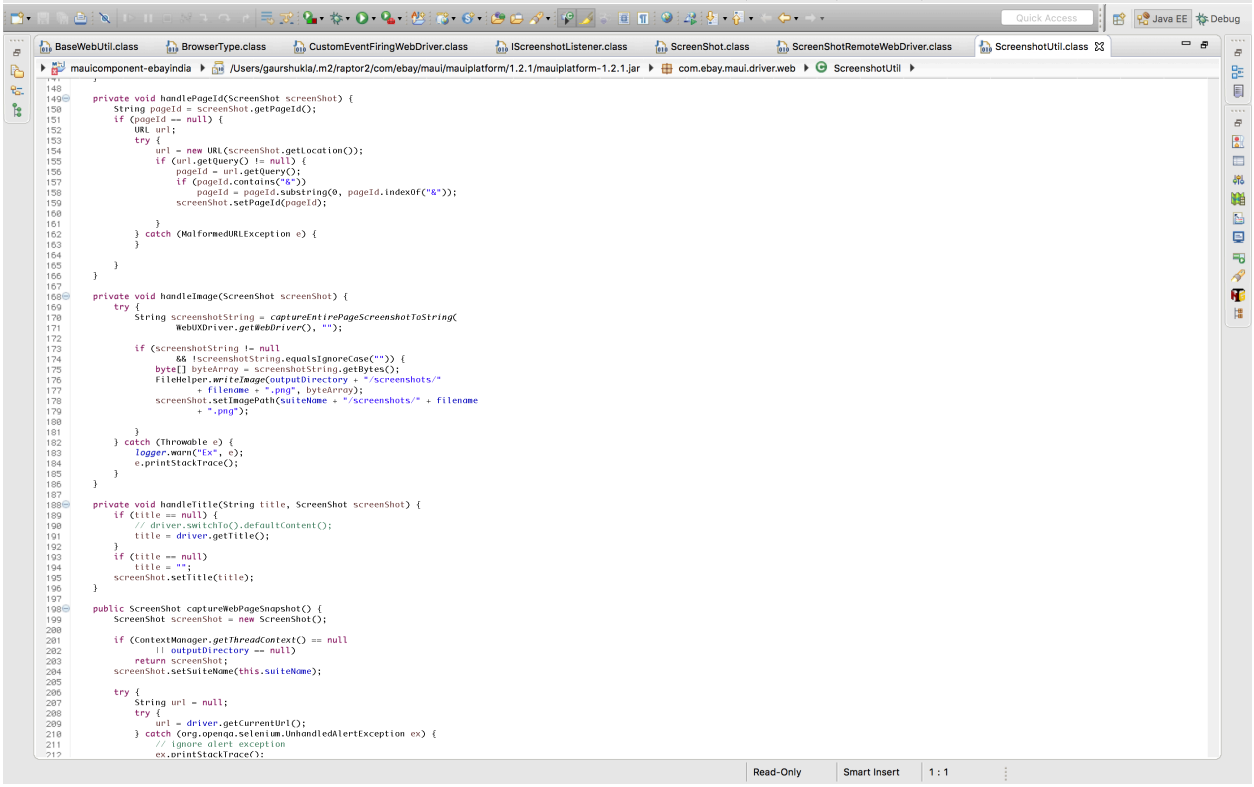


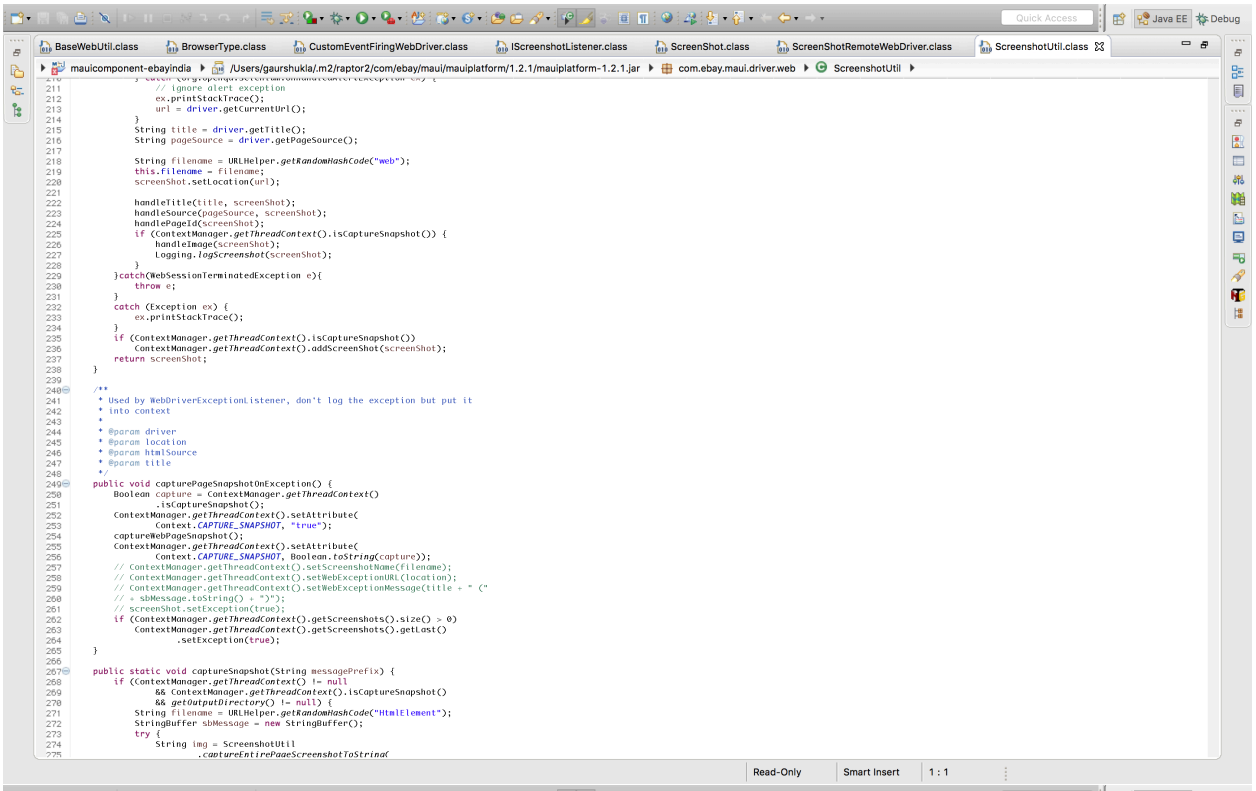
```
package com.ebay.maui.driver.web;

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.DriverCommand;
import org.openqa.selenium.remote.RemoteWebDriver;

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 = execute(DriverCommand.SCREENSHOT).getValue()
                    .toString();
            return target.convertFromBase64Png(output);
        }
        return null;
    }
}
```

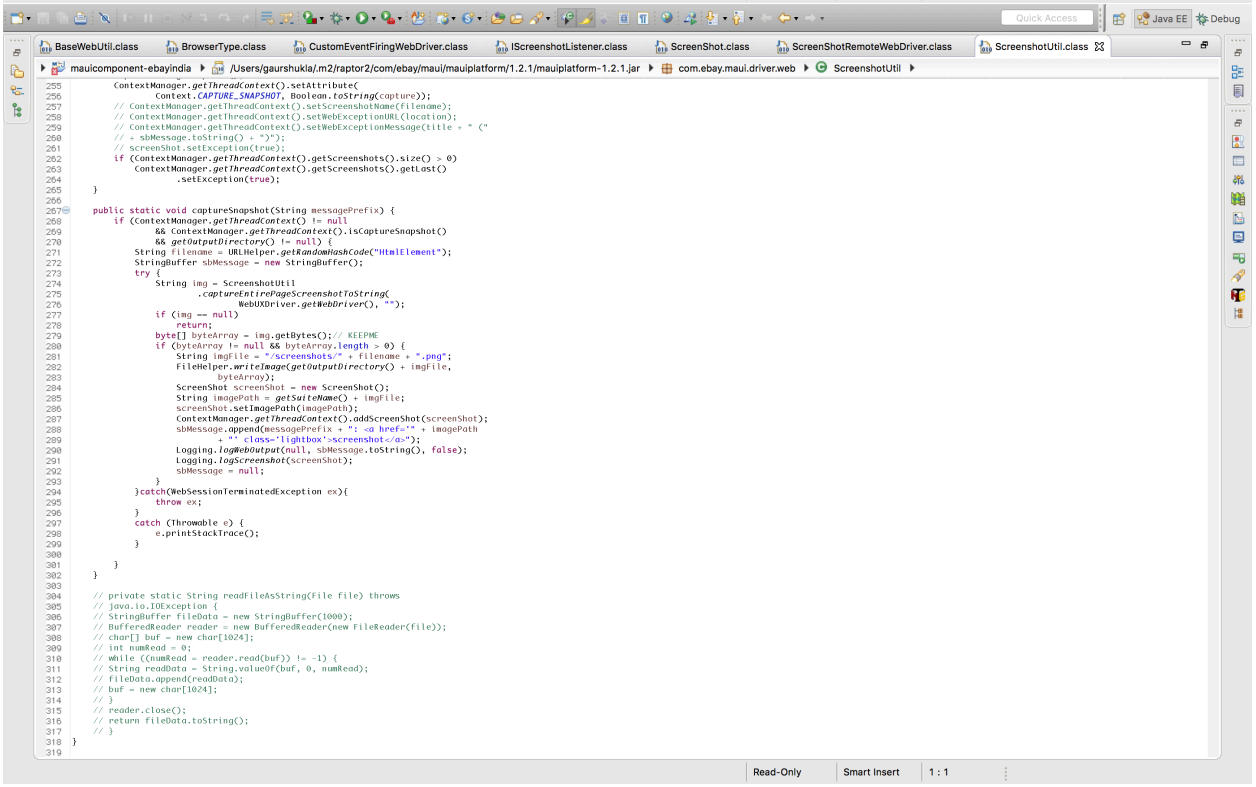


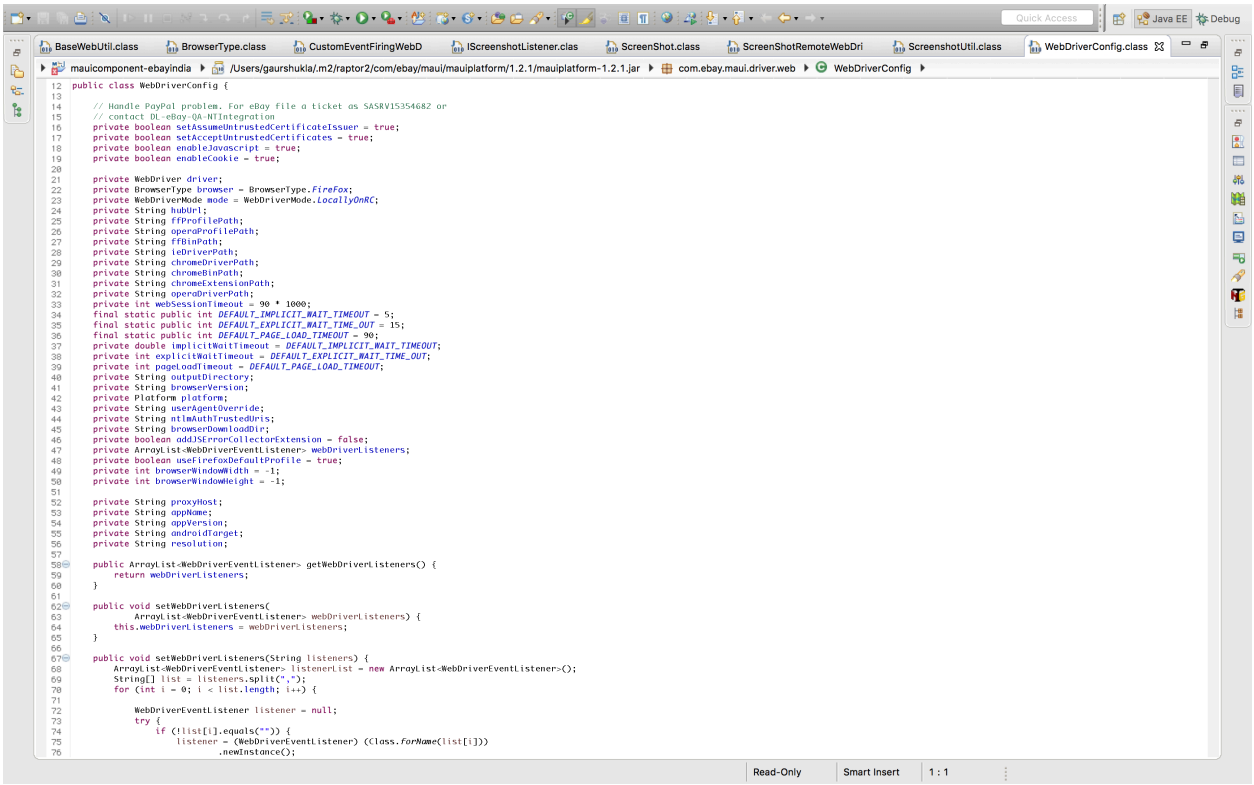
```
public class ScreenshotUtil {
    private static final Logger logger = Logger.getLogger(ScreenshotUtil.class);

    public static String captureEntirePageScreenshotToString(WebDriver driver,
            String arg0) {
        if (driver == null)
            return "";
        try {
            // Don't capture snapshot for htmlunit
            if (WebUXDriver.getWebUXDriver().getBrowser()
                    .equalsIgnoreCase(BrowserType.HtmlUnit.getType()))
                return null;

            // Opera has bug in selenium 2.33.0, bug fix in 2.37.0
            // https://code.google.com/p/selenium/issues/detail?id=847

            TakesScreenshot screenShot = (TakesScreenshot) driver;
            return screenShot.getScreenshotAs(OutputType.BASE64);
        } catch (Exception ex) {
            // Ignore all exceptions
            ex.printStackTrace();
        }

        return "";
    }

    private String suiteName;
    private String outputDirectory;
    private WebDriver driver;
    private String filename;

    public ScreenshotUtil() {
        suiteName = getSuiteName();
        outputDirectory = getOutputDirectory();
        this.driver = WebUXDriver.getWebDriver();
    }

    public ScreenshotUtil(WebDriver driver) {
        suiteName = getSuiteName();
        outputDirectory = getOutputDirectory();
        this.driver = driver;
    }

    private static String getSuiteName() {
        String suiteName = null;

        suiteName = ContextManager.getGlobalContext().getTestNGContext()
                .getSuite().getName();

        return suiteName;
    }

    private static String getOutputDirectory() {
        String outputDirectory = null;
        outputDirectory = ContextManager.getGlobalContext().getTestNGContext()
                .getOutputDirectory();

        return outputDirectory;
    }

    private void handleSource(String htmlSource, ScreenShot screenShot) {
        if (htmlSource == null) {
            // driver.switchTo().defaultContent();
            htmlSource = driver.getPageSource();
```


BaseWebUtil.class
BrowserType.class
CustomEventFiringWebDriver.class
IScreenshotListener.class
ScreenShot.class
ScreenShotRemoteWebDriver.class
ScreenshotUtil.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
ScreenshotUtil
Read-Only
Smart Insert
1 : 1


BaseWebUtil.class
BrowserType.class
CustomEventFiringWebDriver.class
IScreenshotListener.class
ScreenShot.class
ScreenShotRemoteWebDriver.class
ScreenshotUtil.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
ScreenshotUtil
Read-Only
Smart Insert
1 : 1

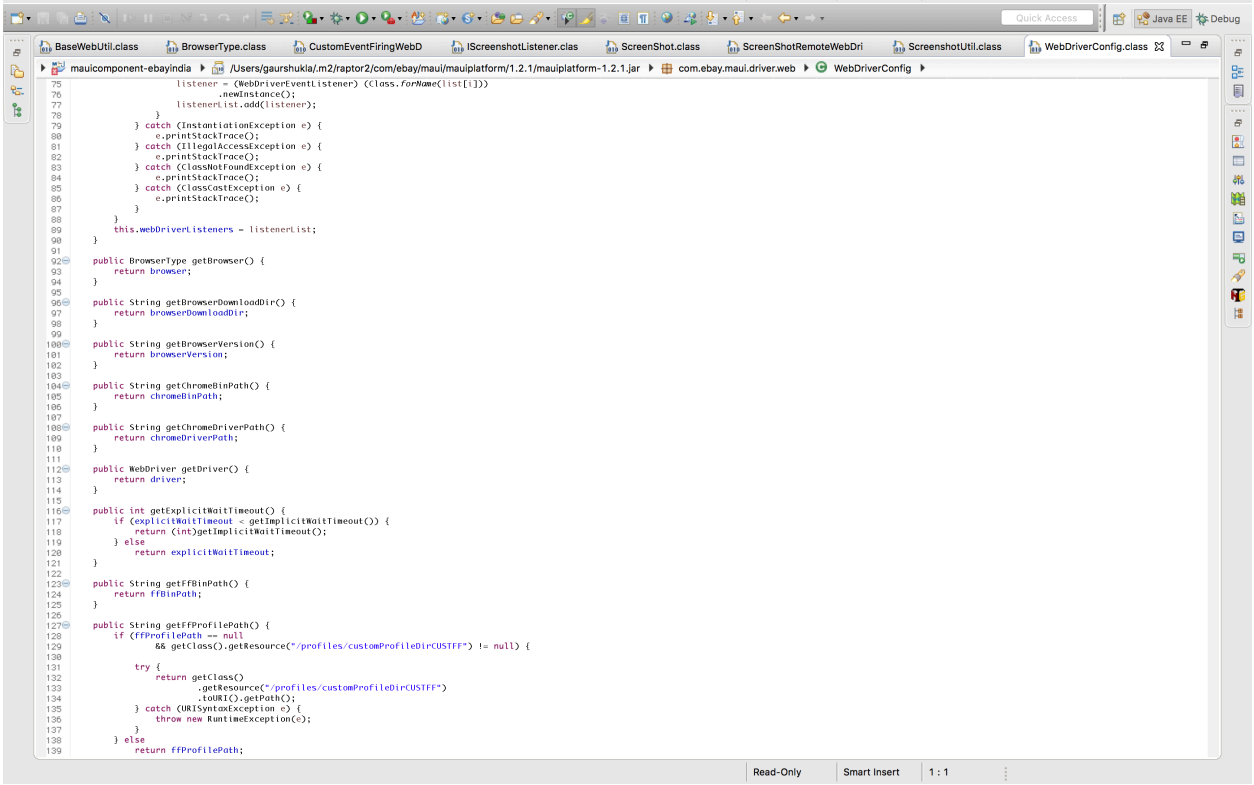

BaseWebUtil.class
BrowserType.class
CustomEventFiringWebDriver.class
IScreenshotListener.class
ScreenShot.class
ScreenShotRemoteWebDriver.class
ScreenshotUtil.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
ScreenshotUtil
Read-Only
Smart Insert
1 : 1


BaseWebUtil.class
BrowserType.class
CustomEventFiringWebDriver.class
IScreenshotListener.class
ScreenShot.class
ScreenShotRemoteWebDriver.class
ScreenshotUtil.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
ScreenshotUtil
Read-Only
Smart Insert
1 : 1



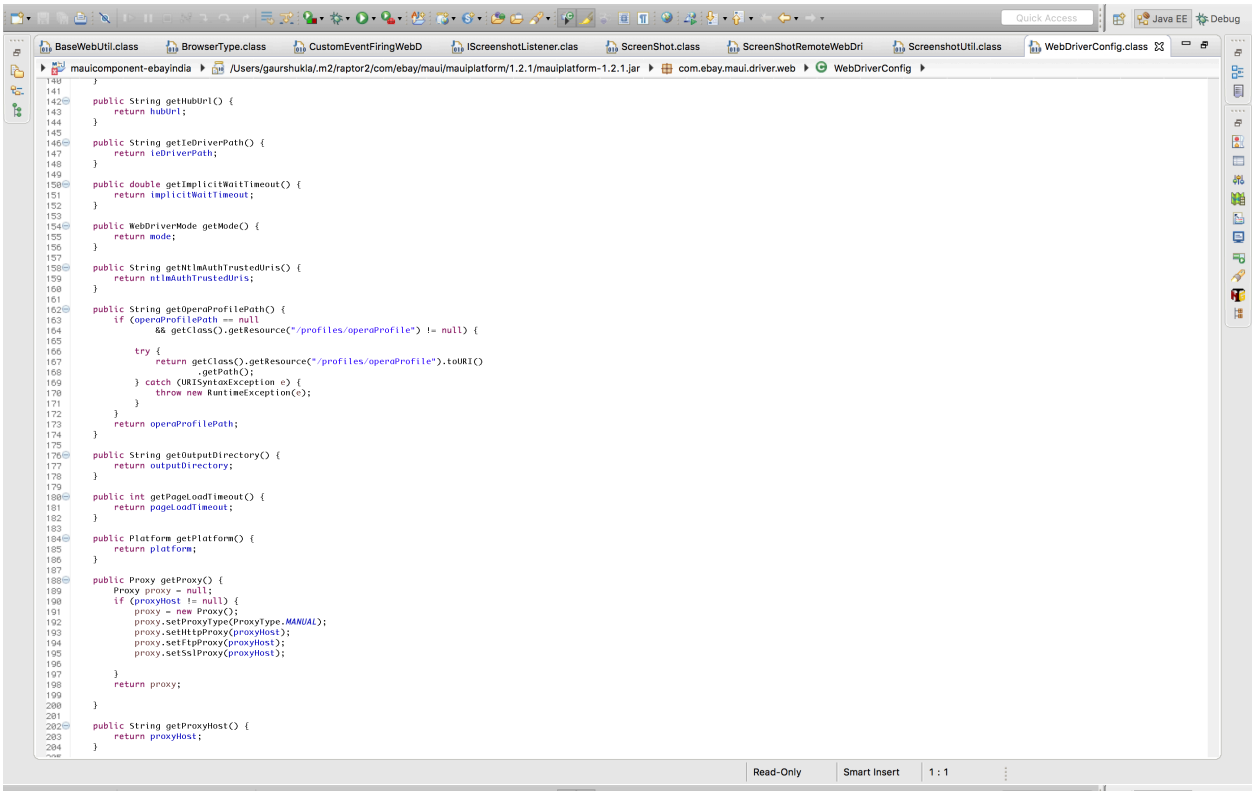
```
12  public class WebDriverConfig {
13
14      // Handle PayPal problem. For eBay file a ticket as SASRV15354682 or
15      // contact DL-eBay-QA-NTIntegration
16      private boolean setAssumeUntrustedCertificateIssuer = true;
17      private boolean setAcceptUntrustedCertificates = true;
18      private boolean enableJavascript = true;
19      private boolean enableCookie = 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 ffProfilePath;
26      private String operaProfilePath;
27      private String ffBinPath;
28      private String ieDriverPath;
29      private String chromeDriverPath;
30      private String chromeBinPath;
31      private String chromeExtensionPath;
32      private String operaDriverPath;
33      private int webSessionTimeout = 90 * 1000;
34      final static public int DEFAULT_IMPLICITT_WAIT_TIMEOUT = 5;
35      final static public int DEFAULT_EXPLICITT_WAIT_TIME_OUT = 15;
36      final static public int DEFAULT_PAGE_LOAD_TIMEOUT = 90;
37      private double implicittWaitTimeout = DEFAULT_IMPLICITT_WAIT_TIMEOUT;
38      private int explicittWaitTimeout = DEFAULT_EXPLICITT_WAIT_TIME_OUT;
39      private int pageLoadTimeout = DEFAULT_PAGE_LOAD_TIMEOUT;
40      private String outputDirectory;
41      private String browserVersion;
42      private Platform platform;
43      private String userAgentOverride;
44      private String ntlmAuthTrustedUris;
45      private String browserDownloadDir;
46      private boolean addJSErrorCollectorExtension = false;
47      private ArrayList<WebDriverEventListener> webDriverListeners;
48      private boolean useFirefoxDefaultProfile = true;
49      private int browserWindowWidth = -1;
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
72              WebDriverEventListener listener = null;
73              try {
74                  if (!list[i].equals("")) {
75                      listener = (WebDriverEventListener) (Class.forName(list[i]))
76                              .newInstance();
```

Read-Only | Smart Insert | 1 : 1

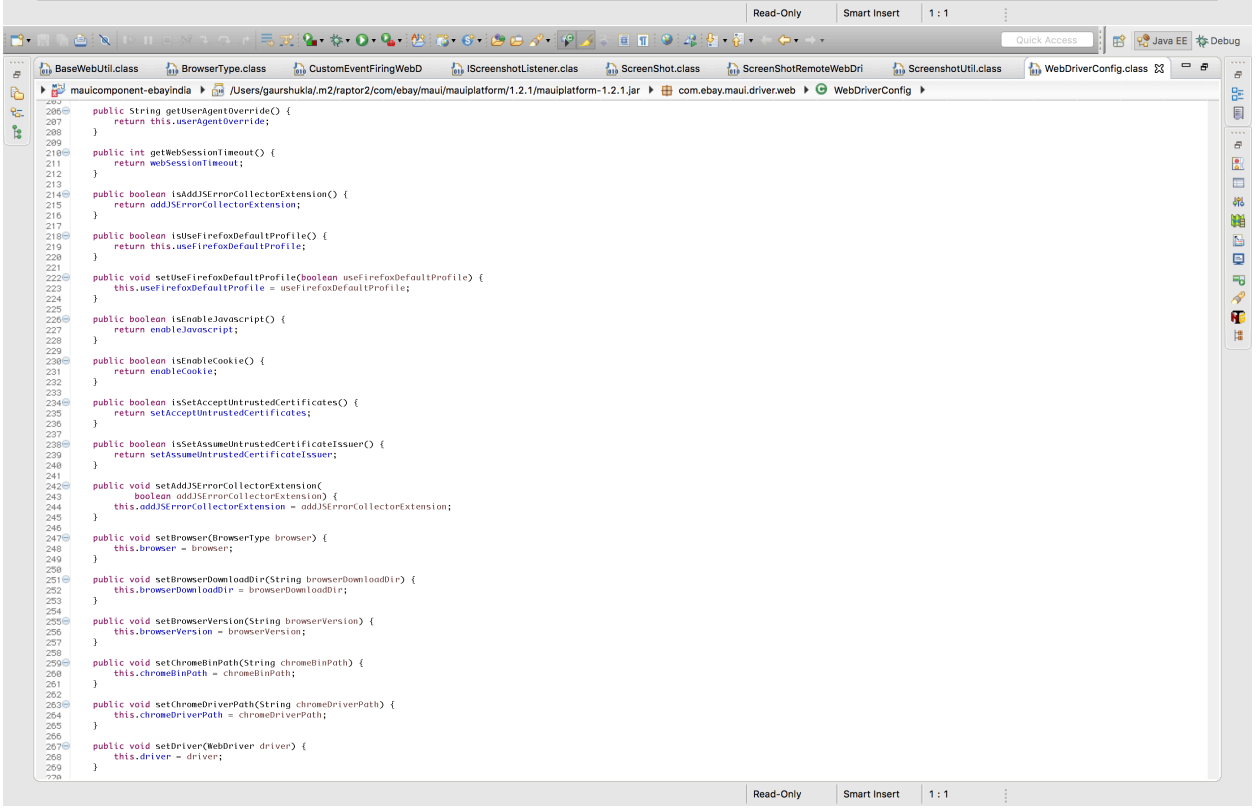


```
75                      listener = (WebDriverEventListener) (Class.forName(list[i]))
76                              .newInstance();
77                      listenerList.add(listener);
78                  }
79              } catch (InstantiationException e) {
80                  e.printStackTrace();
81              } catch (IllegalAccessException e) {
82                  e.printStackTrace();
83              } catch (ClassNotFoundException e) {
84                  e.printStackTrace();
85              } catch (ClassCastException 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 getChromeBinPath() {
105         return chromeBinPath;
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     public String getFfBinPath() {
124         return ffBinPath;
125     }
126
127     public String getFfProfilePath() {
128         if (ffProfilePath == null
129                 && getClass().getResource("/profiles/customProfileDirCUSTFF") != null) {
130
131             try {
132                 return getClass()
133                         .getResource("/profiles/customProfileDirCUSTFF")
134                         .toURI().getPath();
135             } catch (URISyntaxException e) {
136                 throw new RuntimeException(e);
137             }
138         } else
139             return ffProfilePath;
```

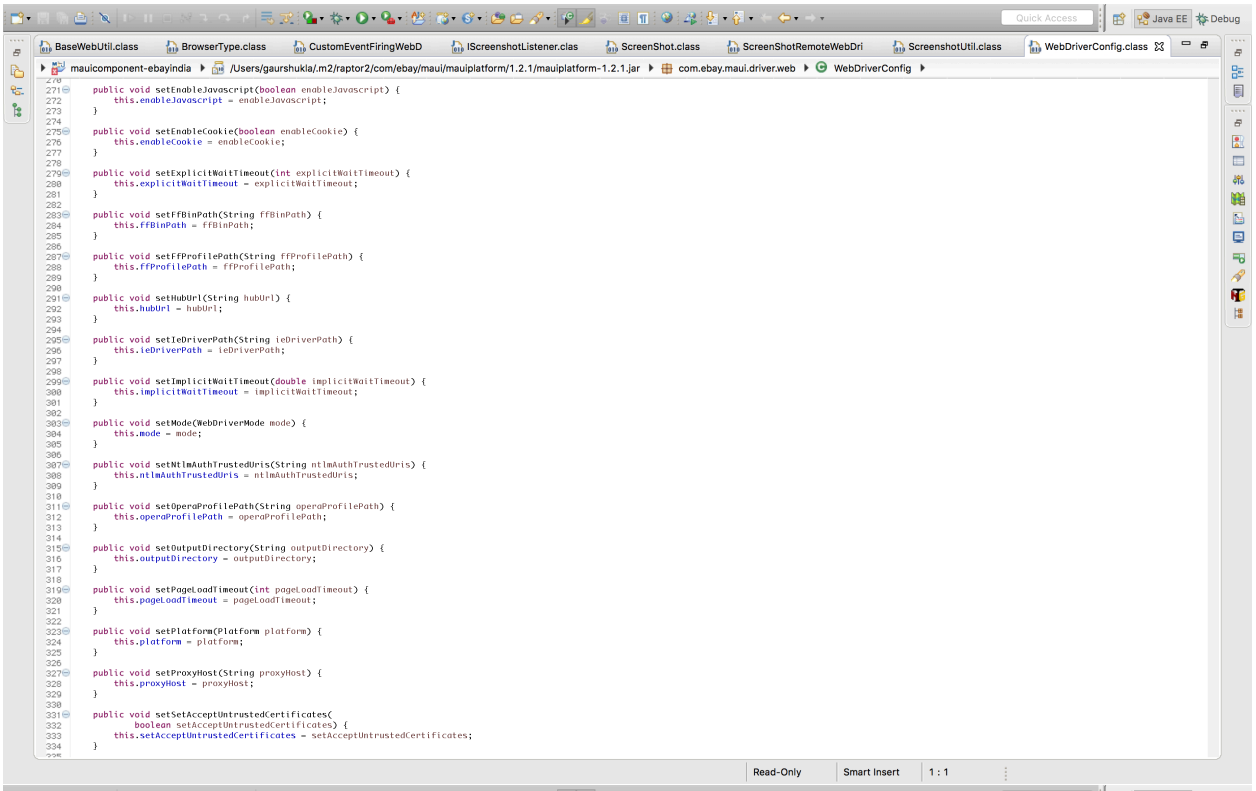
Read-Only | Smart Insert | 1 : 1

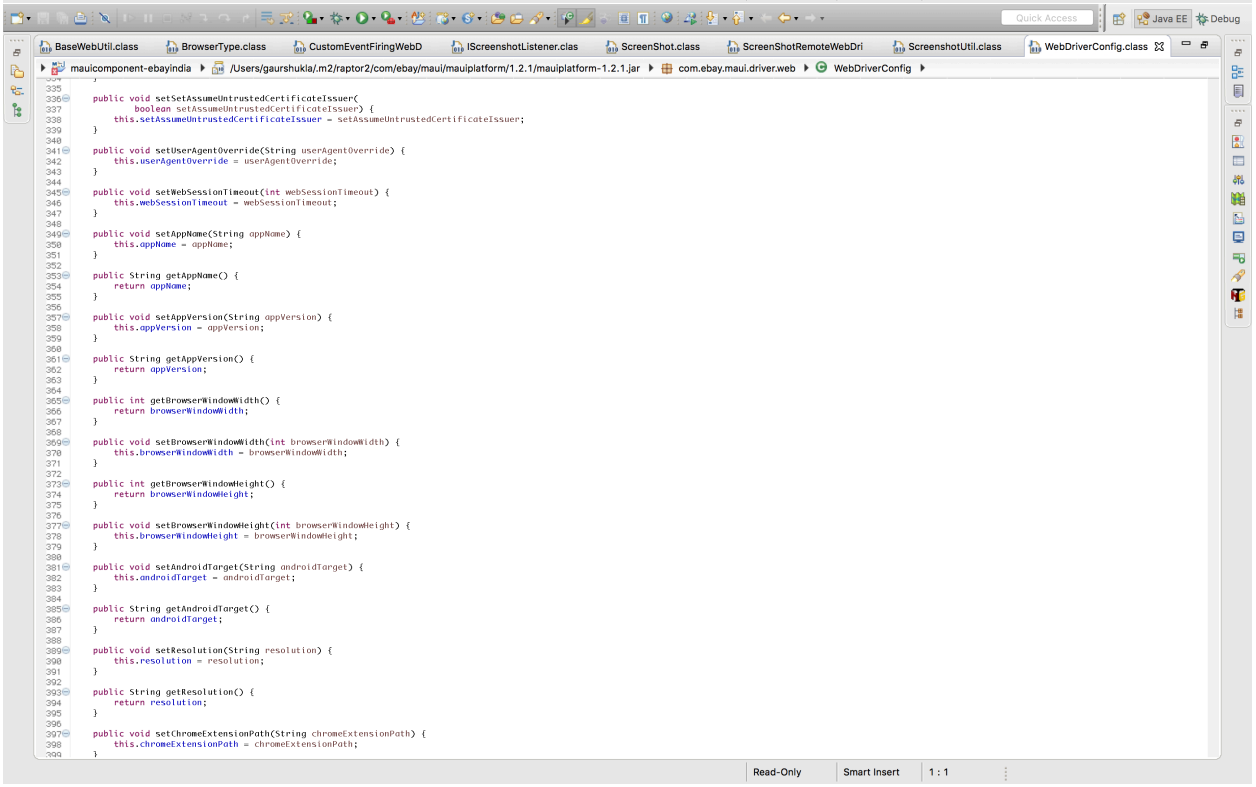


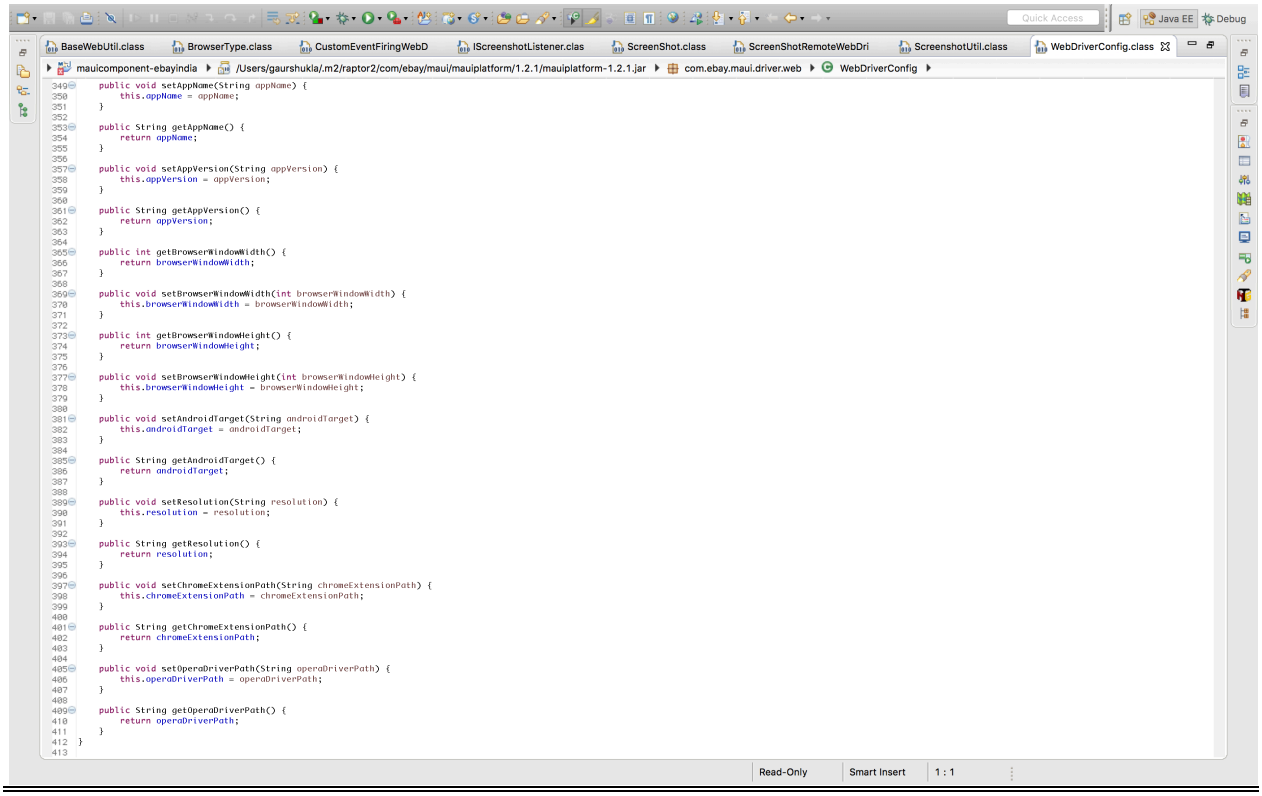
```
    }

    public String getHubUrl() {
        return hubUrl;
    }

    public String getIeDriverPath() {
        return ieDriverPath;
    }

    public double getImplicitWaitTimeout() {
        return implicitWaitTimeout;
    }

    public WebDriverMode getMode() {
        return mode;
    }

    public String getNtlmAuthTrustedUris() {
        return ntlmAuthTrustedUris;
    }

    public String getOperaProfilePath() {
        if (operaProfilePath == null
                && getClass().getResource("/profiles/operaProfile") != null) {

            try {
                return getClass().getResource("/profiles/operaProfile").toURI()
                        .getPath();
            } catch (URISyntaxException e) {
                throw new RuntimeException(e);
            }
        }
        return operaProfilePath;
    }

    public String getOutputDirectory() {
        return outputDirectory;
    }

    public int getPageLoadTimeout() {
        return pageLoadTimeout;
    }

    public Platform getPlatform() {
        return platform;
    }

    public Proxy getProxy() {
        Proxy proxy = null;
        if (proxyHost != null) {
            proxy = new Proxy();
            proxy.setProxyType(ProxyType.MANUAL);
            proxy.setHttpProxy(proxyHost);
            proxy.setFtpProxy(proxyHost);
            proxy.setSslProxy(proxyHost);

        }
        return proxy;

    }

    public String getProxyHost() {
        return proxyHost;
    }
```



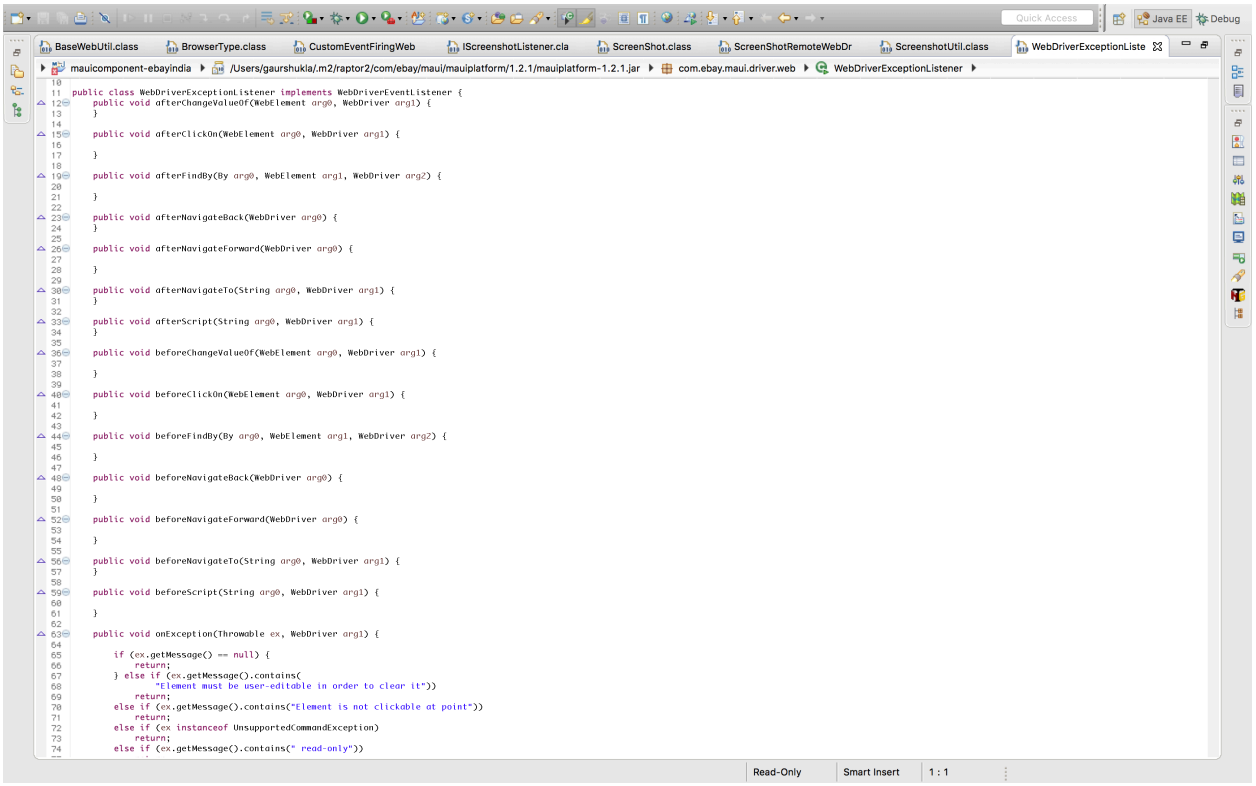
```
    public String getUserAgentOverride() {
        return this.userAgentOverride;
    }

    public int getWebSessionTimeout() {
        return webSessionTimeout;
    }

    public boolean isAddJSErrorCollectorExtension() {
        return addJSErrorCollectorExtension;
    }

    public boolean isUseFirefoxDefaultProfile() {
        return this.useFirefoxDefaultProfile;
    }

    public void setUseFirefoxDefaultProfile(boolean useFirefoxDefaultProfile) {
        this.useFirefoxDefaultProfile = useFirefoxDefaultProfile;
    }

    public boolean isEnableJavascript() {
        return enableJavascript;
    }

    public boolean isEnableCookie() {
        return enableCookie;
    }

    public boolean isSetAcceptUntrustedCertificates() {
        return setAcceptUntrustedCertificates;
    }

    public boolean isSetAssumeUntrustedCertificateIssuer() {
        return setAssumeUntrustedCertificateIssuer;
    }

    public void setAddJSErrorCollectorExtension(
            boolean addJSErrorCollectorExtension) {
        this.addJSErrorCollectorExtension = addJSErrorCollectorExtension;
    }

    public void setBrowser(BrowserType browser) {
        this.browser = browser;
    }

    public void setBrowserDownloadDir(String browserDownloadDir) {
        this.browserDownloadDir = browserDownloadDir;
    }

    public void setBrowserVersion(String browserVersion) {
        this.browserVersion = browserVersion;
    }

    public void setChromeBinPath(String chromeBinPath) {
        this.chromeBinPath = chromeBinPath;
    }

    public void setChromeDriverPath(String chromeDriverPath) {
        this.chromeDriverPath = chromeDriverPath;
    }

    public void setDriver(WebDriver driver) {
        this.driver = driver;
    }
```


Quick Access
Java EE
Debug
BaseWebUtil.class
BrowserType.class
CustomEventFiringWebD
IScreenshotListener.clas
ScreenShot.class
ScreenShotRemoteWebDri
ScreenshotUtil.class
WebDriverConfig.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
WebDriverConfig
public void setEnableJavascript(boolean enableJavascript) {
this.enableJavascript = enableJavascript;
}
public void setEnableCookie(boolean enableCookie) {
this.enableCookie = enableCookie;
}
public void setExplicitWaitTimeout(int explicitWaitTimeout) {
this.explicitWaitTimeout = explicitWaitTimeout;
}
public void setFfBinPath(String ffBinPath) {
this.ffBinPath = ffBinPath;
}
public void setFfProfilePath(String ffProfilePath) {
this.ffProfilePath = ffProfilePath;
}
public void setHubUrl(String hubUrl) {
this.hubUrl = hubUrl;
}
public void setIeDriverPath(String ieDriverPath) {
this.ieDriverPath = ieDriverPath;
}
public void setImplicitWaitTimeout(double implicitWaitTimeout) {
this.implicitWaitTimeout = implicitWaitTimeout;
}
public void setMode(WebDriverMode mode) {
this.mode = mode;
}
public void setNtlmAuthTrustedUris(String ntlmAuthTrustedUris) {
this.ntlmAuthTrustedUris = ntlmAuthTrustedUris;
}
public void setOperaProfilePath(String operaProfilePath) {
this.operaProfilePath = operaProfilePath;
}
public void setOutputDirectory(String outputDirectory) {
this.outputDirectory = outputDirectory;
}
public void setPageLoadTimeout(int pageLoadTimeout) {
this.pageLoadTimeout = pageLoadTimeout;
}
public void setPlatform(Platform platform) {
this.platform = platform;
}
public void setProxyHost(String proxyHost) {
this.proxyHost = proxyHost;
}
public void setSetAcceptUntrustedCertificates(
boolean setAcceptUntrustedCertificates) {
this.setAcceptUntrustedCertificates = setAcceptUntrustedCertificates;
}
Read-Only
Smart Insert
1 : 1

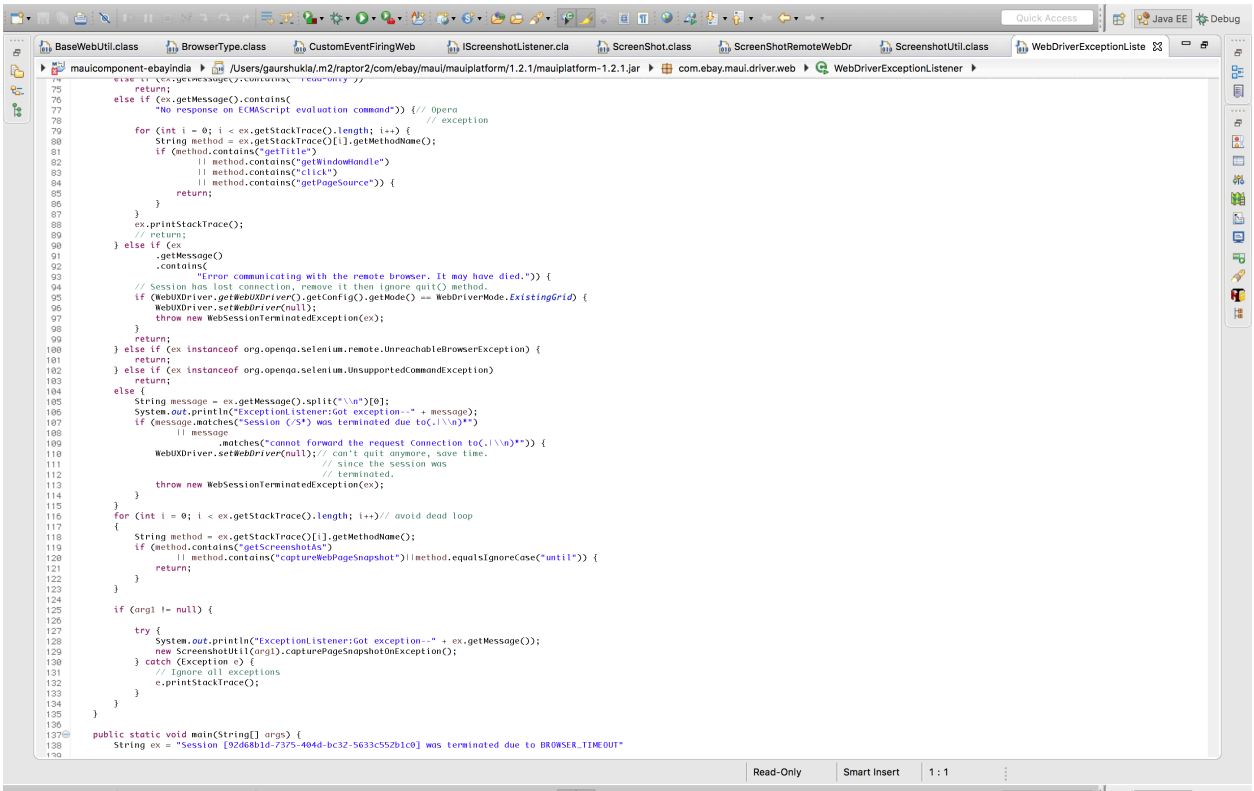

Quick Access
Java EE
Debug
BaseWebUtil.class
BrowserType.class
CustomEventFiringWebD
IScreenshotListener.clas
ScreenShot.class
ScreenShotRemoteWebDri
ScreenshotUtil.class
WebDriverConfig.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
WebDriverConfig
public void setSetAssumeUntrustedCertificateIssuer(
boolean setAssumeUntrustedCertificateIssuer) {
this.setAssumeUntrustedCertificateIssuer = setAssumeUntrustedCertificateIssuer;
}
public void setUserAgentOverride(String userAgentOverride) {
this.userAgentOverride = userAgentOverride;
}
public void setWebSessionTimeout(int webSessionTimeout) {
this.webSessionTimeout = webSessionTimeout;
}
public void setAppName(String appName) {
this.appName = appName;
}
public String getAppName() {
return appName;
}
public void setAppVersion(String appVersion) {
this.appVersion = appVersion;
}
public String getAppVersion() {
return appVersion;
}
public int getBrowserWindowWidth() {
return browserWindowWidth;
}
public void setBrowserWindowWidth(int browserWindowWidth) {
this.browserWindowWidth = browserWindowWidth;
}
public int getBrowserWindowHeight() {
return browserWindowHeight;
}
public void setBrowserWindowHeight(int browserWindowHeight) {
this.browserWindowHeight = browserWindowHeight;
}
public void setAndroidTarget(String androidTarget) {
this.androidTarget = androidTarget;
}
public String getAndroidTarget() {
return androidTarget;
}
public void setResolution(String resolution) {
this.resolution = resolution;
}
public String getResolution() {
return resolution;
}
public void setChromeExtensionPath(String chromeExtensionPath) {
this.chromeExtensionPath = chromeExtensionPath;
}
Read-Only
Smart Insert
1 : 1

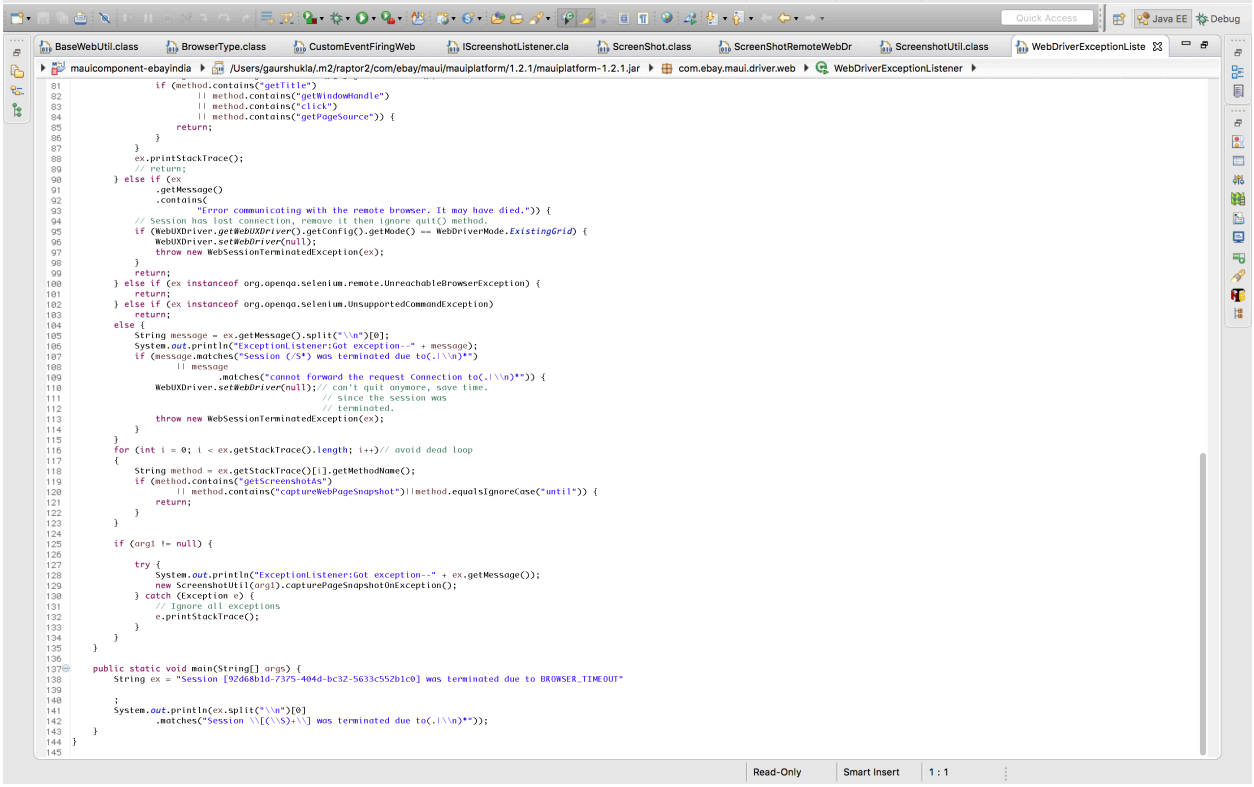


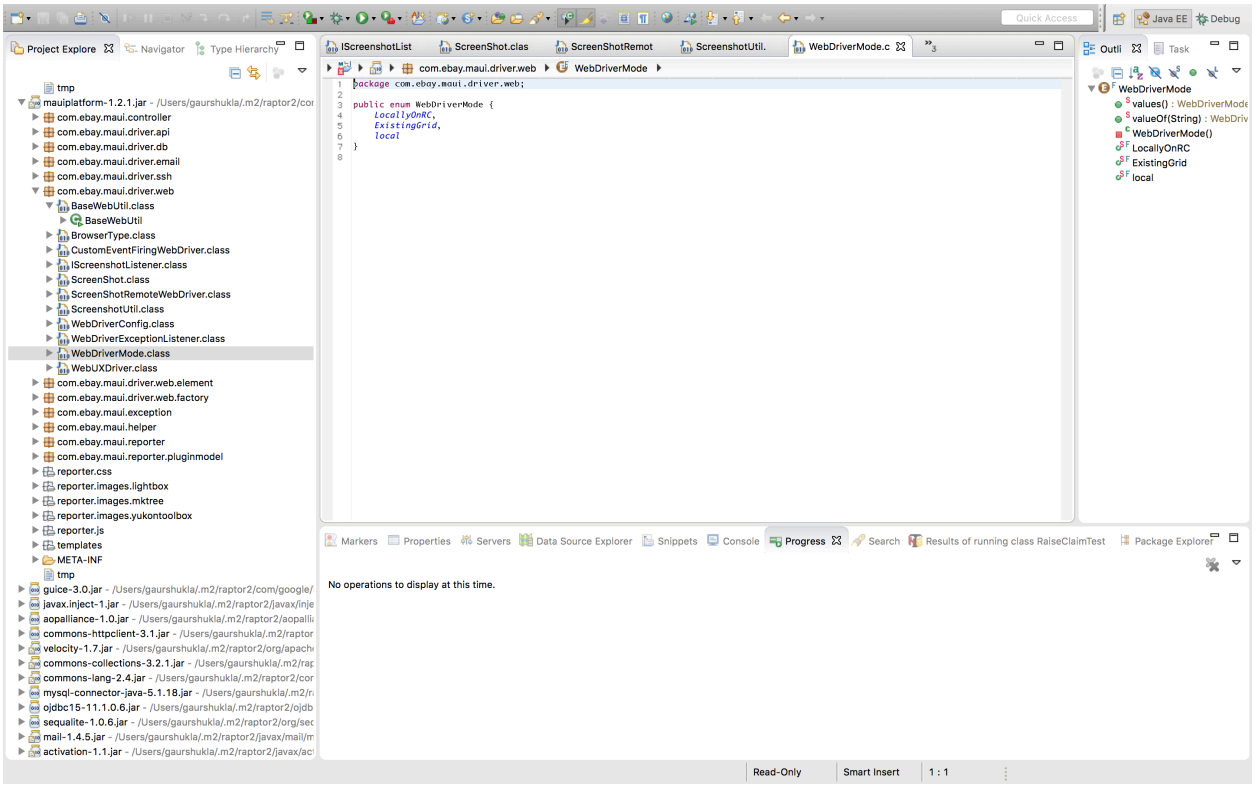
```
    public void setAppName(String appName) {
        this.appName = appName;
    }

    public String getAppName() {
        return appName;
    }

    public void setAppVersion(String appVersion) {
        this.appVersion = appVersion;
    }

    public String getAppVersion() {
        return appVersion;
    }

    public int getBrowserWindowWidth() {
        return browserWindowWidth;
    }

    public void setBrowserWindowWidth(int browserWindowWidth) {
        this.browserWindowWidth = browserWindowWidth;
    }

    public int getBrowserWindowHeight() {
        return browserWindowHeight;
    }

    public void setBrowserWindowHeight(int browserWindowHeight) {
        this.browserWindowHeight = browserWindowHeight;
    }

    public void setAndroidTarget(String androidTarget) {
        this.androidTarget = androidTarget;
    }

    public String getAndroidTarget() {
        return androidTarget;
    }

    public void setResolution(String resolution) {
        this.resolution = resolution;
    }

    public String getResolution() {
        return resolution;
    }

    public void setChromeExtensionPath(String chromeExtensionPath) {
        this.chromeExtensionPath = chromeExtensionPath;
    }

    public String getChromeExtensionPath() {
        return chromeExtensionPath;
    }

    public void setOperaDriverPath(String operaDriverPath) {
        this.operaDriverPath = operaDriverPath;
    }

    public String getOperaDriverPath() {
        return operaDriverPath;
    }
}
```



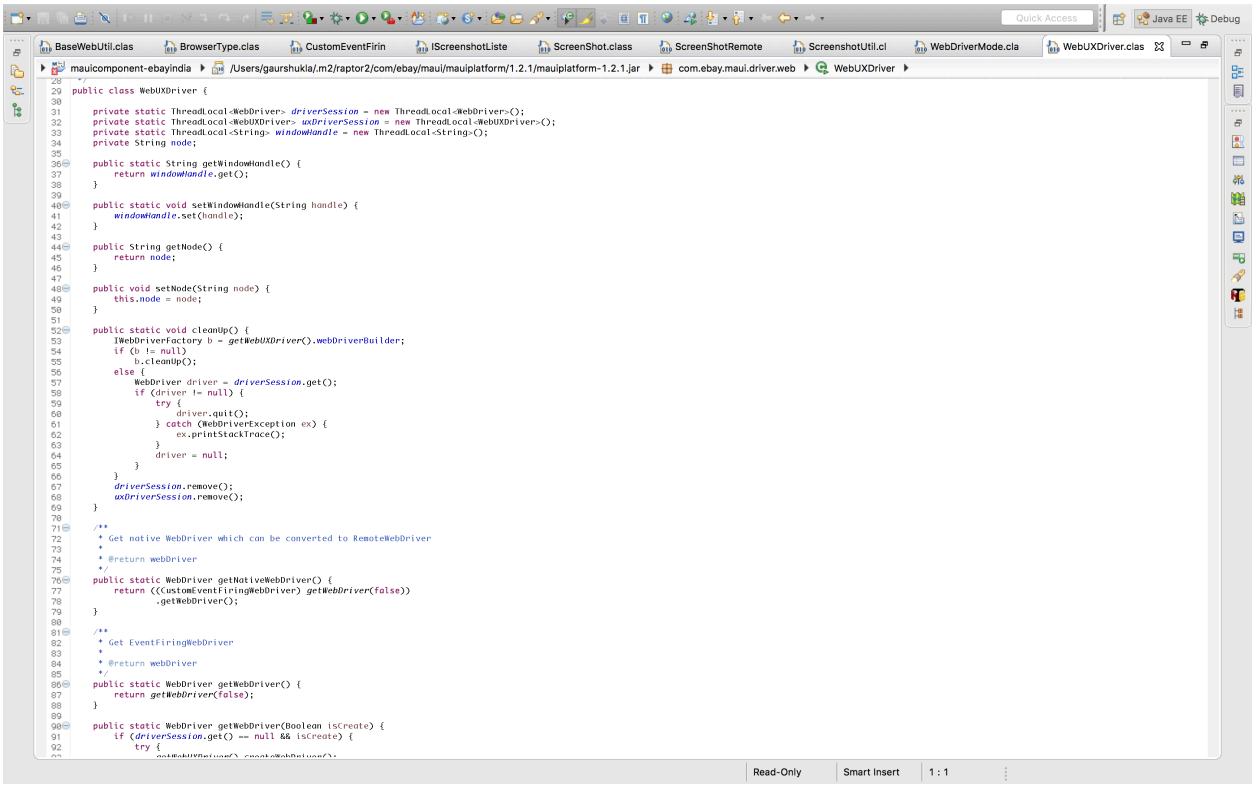
```
public class WebDriverExceptionListener implements WebDriverEventListener {
    public void afterChangeValueOf(WebElement arg0, WebDriver arg1) {
    }

    public void afterClickOn(WebElement arg0, WebDriver arg1) {

    }

    public void afterFindBy(By arg0, WebElement arg1, WebDriver arg2) {

    }

    public void afterNavigateBack(WebDriver arg0) {
    }

    public void afterNavigateForward(WebDriver arg0) {

    }

    public void afterNavigateTo(String arg0, WebDriver arg1) {
    }

    public void afterScript(String arg0, WebDriver arg1) {
    }

    public void beforeChangeValueOf(WebElement arg0, WebDriver arg1) {

    }

    public void beforeClickOn(WebElement arg0, WebDriver arg1) {

    }

    public void beforeFindBy(By arg0, WebElement arg1, WebDriver arg2) {

    }

    public void beforeNavigateBack(WebDriver arg0) {

    }

    public void beforeNavigateForward(WebDriver arg0) {

    }

    public void beforeNavigateTo(String arg0, WebDriver arg1) {
    }

    public void beforeScript(String arg0, WebDriver arg1) {

    }

    public void onException(Throwable ex, WebDriver arg1) {

        if (ex.getMessage() == null) {
            return;
        } else if (ex.getMessage().contains(
                "Element must be user-editable in order to clear it"))
            return;
        else if (ex.getMessage().contains("Element is not clickable at point"))
            return;
        else if (ex instanceof UnsupportedCommandException)
            return;
        else if (ex.getMessage().contains(" read-only"))
```


BaseWebUtil.class
BrowserType.class
CustomEventFiringWeb
IScreenshotListener.cla
ScreenShot.class
ScreenShotRemoteWebDr
ScreenshotUtil.class
WebDriverExceptionListe
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
WebDriverExceptionListener
Read-Only
Smart Insert
1 : 1

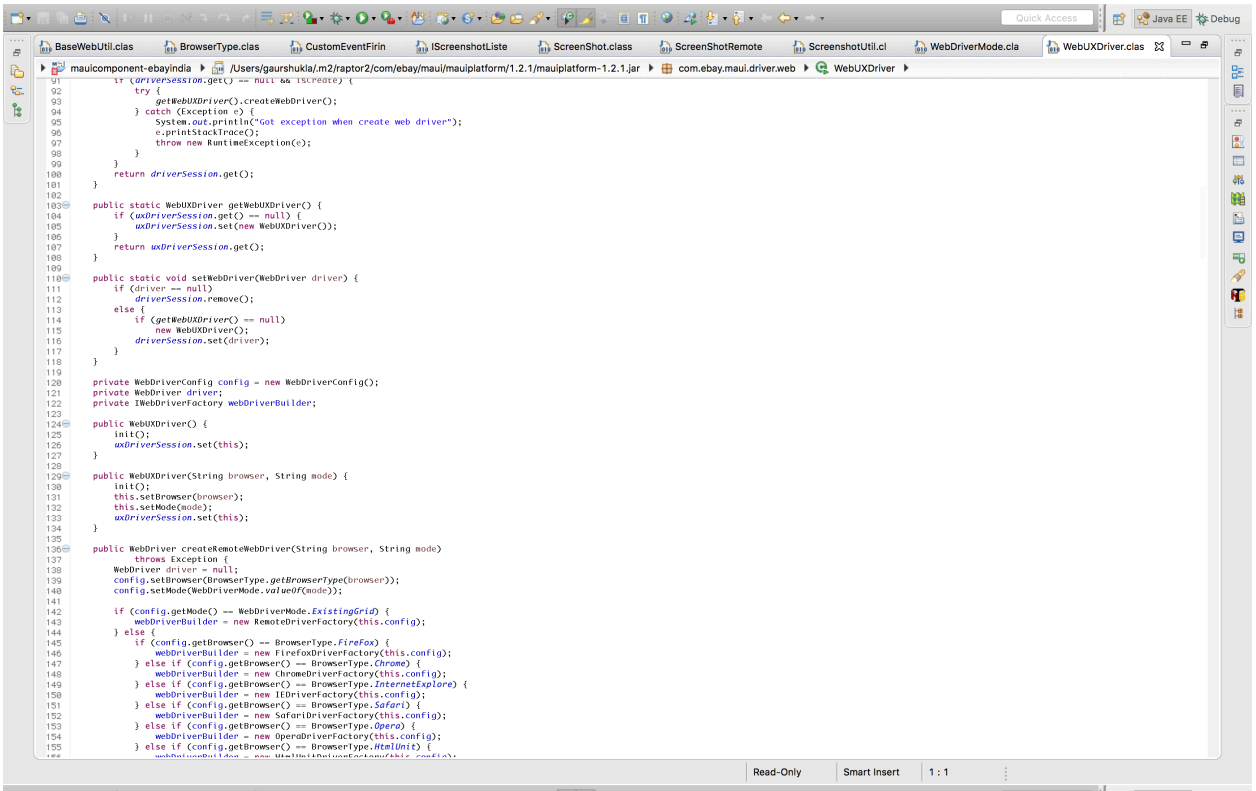

BaseWebUtil.class
BrowserType.class
CustomEventFiringWeb
IScreenshotListener.cla
ScreenShot.class
ScreenShotRemoteWebDr
ScreenshotUtil.class
WebDriverExceptionListe
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
WebDriverExceptionListener
Read-Only
Smart Insert
1 : 1

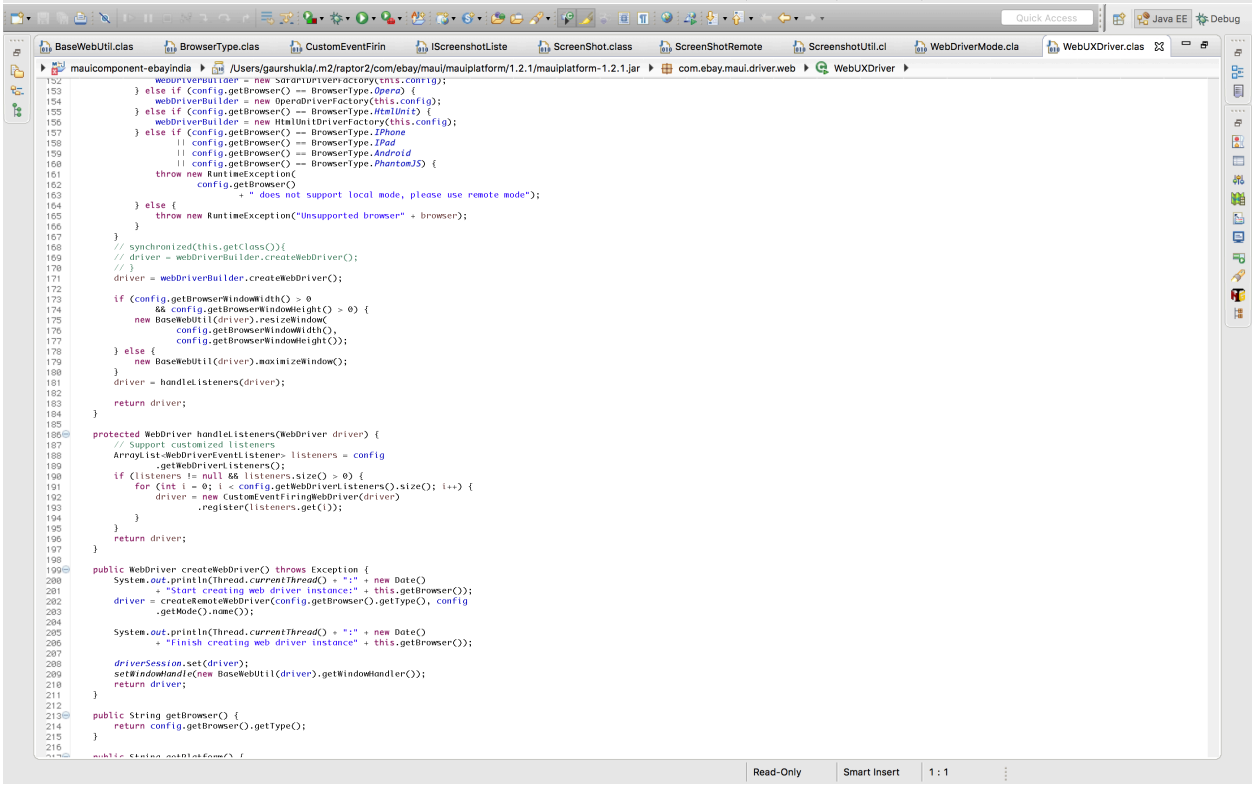


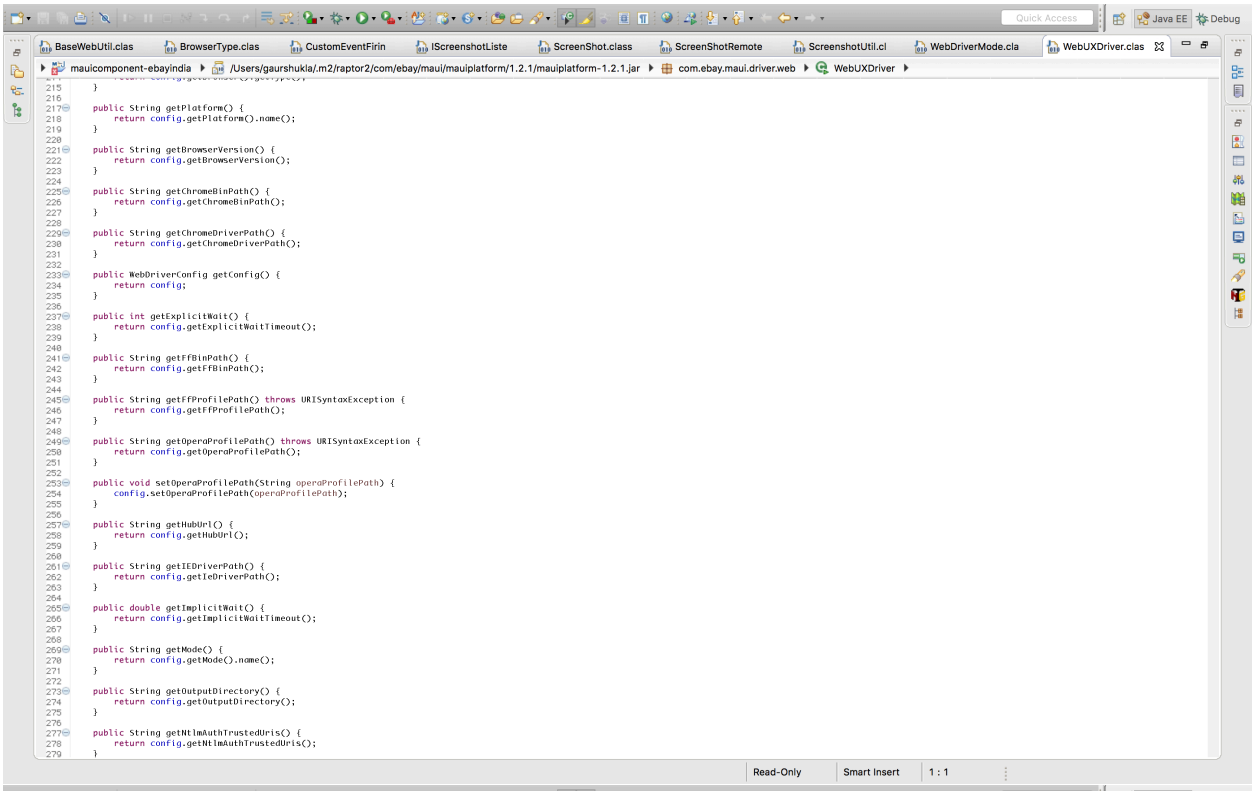
```
package com.ebay.maui.driver.web;

public enum WebDriverMode {
    LocallyOnRC,
    ExistingGrid,
    local
}
```

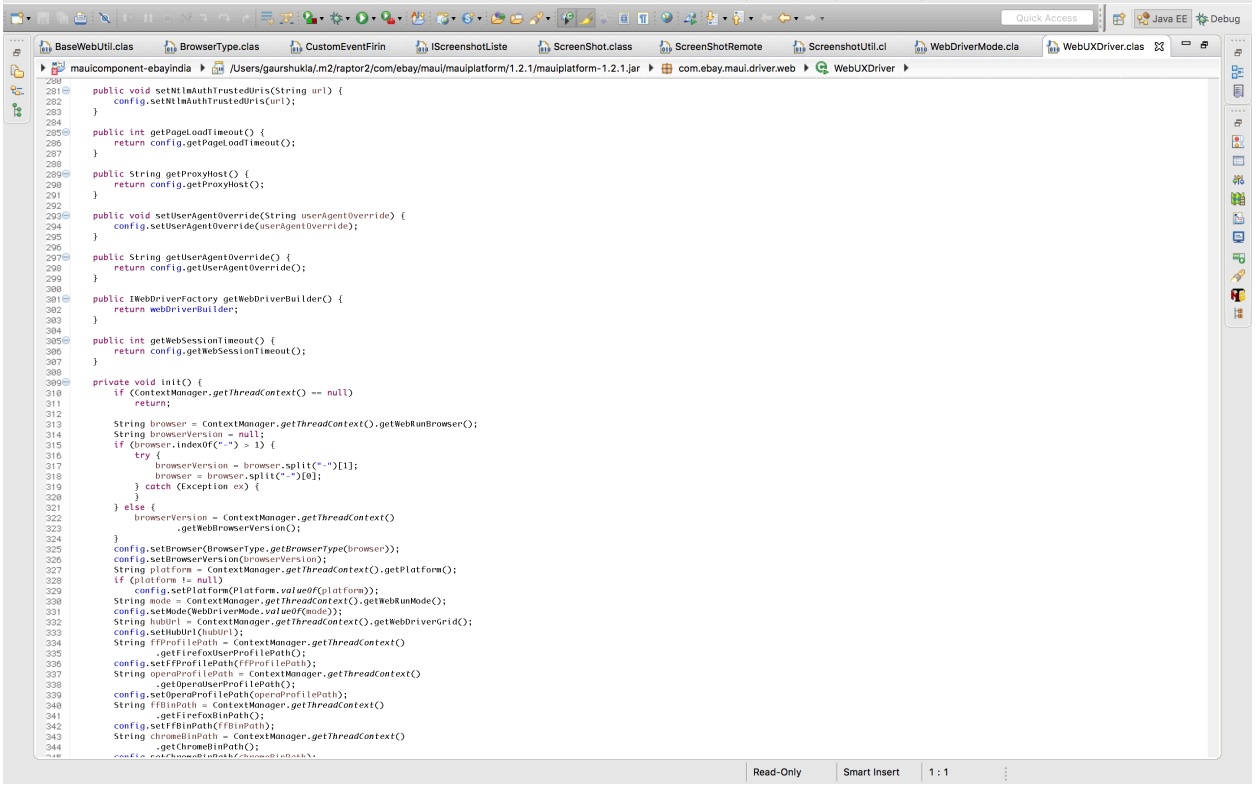


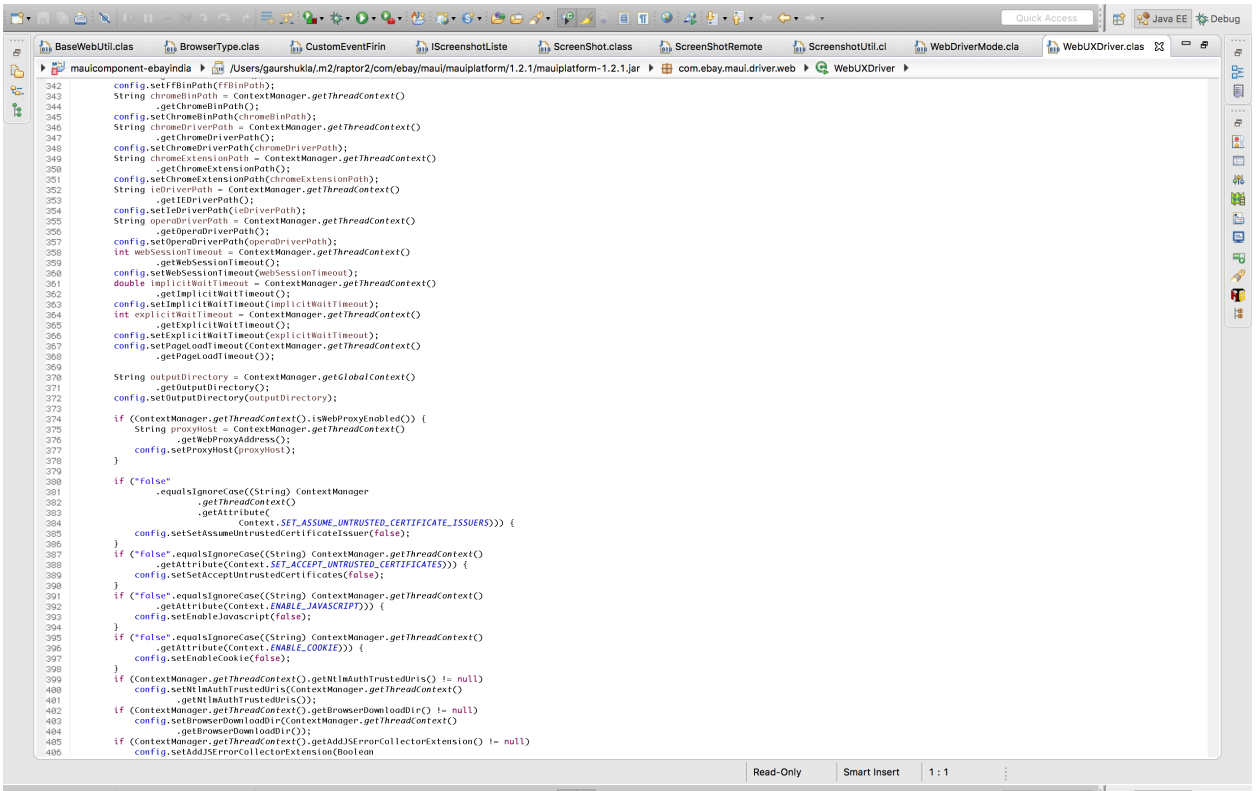
```
public class WebUXDriver {

    private static ThreadLocal<WebDriver> driverSession = new ThreadLocal<WebDriver>();
    private static ThreadLocal<WebUXDriver> uxDriverSession = new ThreadLocal<WebUXDriver>();
    private static ThreadLocal<String> windowHandle = new ThreadLocal<String>();
    private String node;

    public static String getWindowHandle() {
        return windowHandle.get();
    }

    public static void setWindowHandle(String handle) {
        windowHandle.set(handle);
    }

    public String getNode() {
        return node;
    }

    public void setNode(String node) {
        this.node = node;
    }

    public static void cleanUp() {
        IWebDriverFactory b = getWebUXDriver().webDriverBuilder;
        if (b != null)
            b.cleanUp();
        else {
            WebDriver driver = driverSession.get();
            if (driver != null) {
                try {
                    driver.quit();
                } catch (WebDriverException ex) {
                    ex.printStackTrace();
                }
                driver = null;
            }
        }
        driverSession.remove();
        uxDriverSession.remove();
    }

    /**
     * Get native WebDriver which can be converted to RemoteWebDriver
     *
     * @return webDriver
     */
    public static WebDriver getNativeWebDriver() {
        return ((CustomEventFiringWebDriver) getWebDriver(false))
                .getWebDriver();
    }

    /**
     * Get EventFiringWebDriver
     *
     * @return webDriver
     */
    public static WebDriver getWebDriver() {
        return getWebDriver(false);
    }

    public static WebDriver getWebDriver(Boolean isCreate) {
        if (driverSession.get() == null && isCreate) {
            try {
```

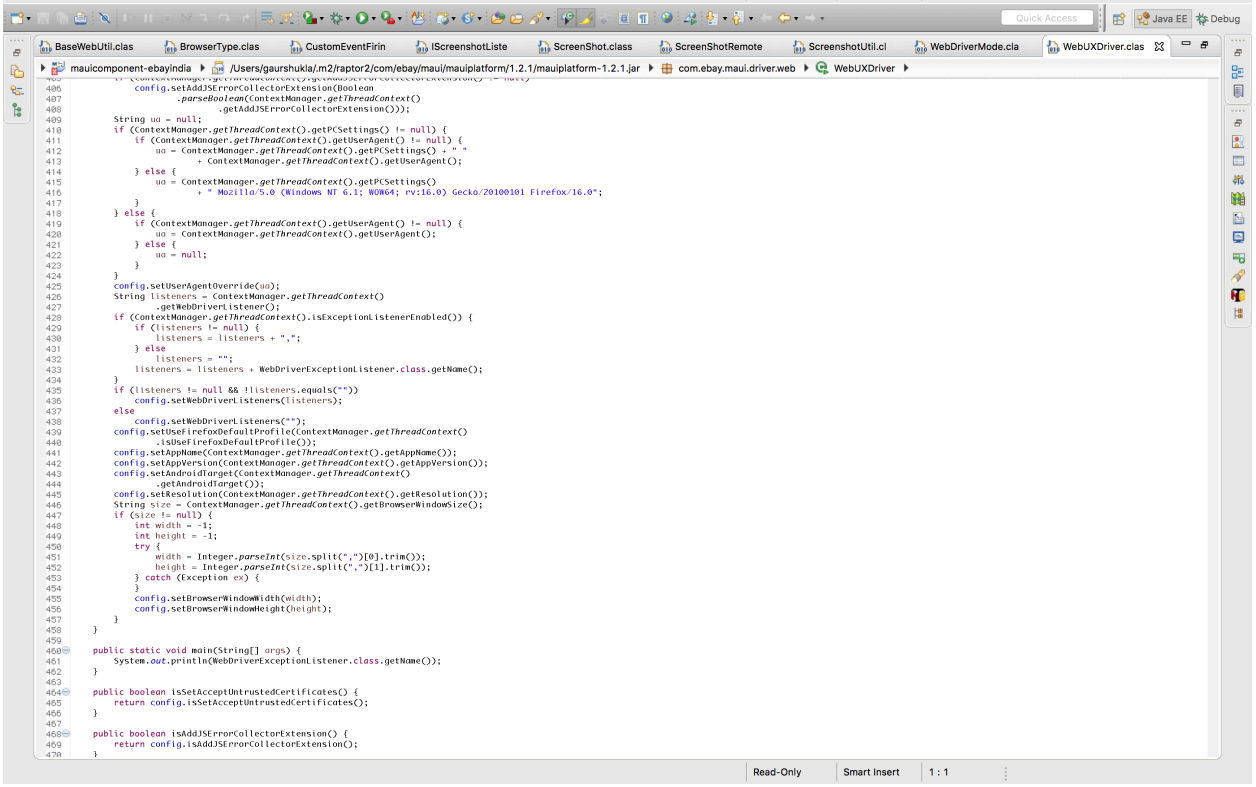

BaseWebUtil.clas
BrowserType.clas
CustomEventFirin
IScreenshotListe
ScreenShot.class
ScreenShotRemote
ScreenshotUtil.cl
WebDriverMode.cla
WebUXDriver.clas
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
WebUXDriver
Read-Only
Smart Insert
1 : 1

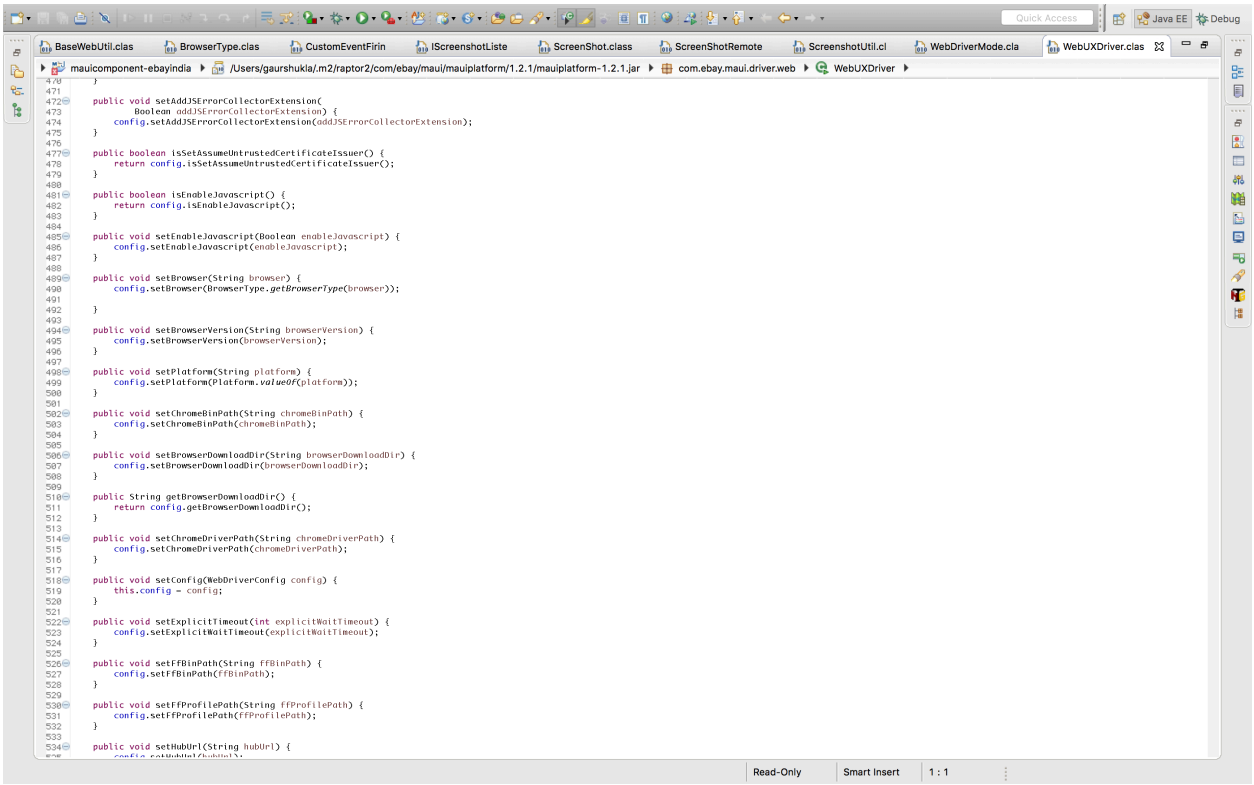

BaseWebUtil.clas
BrowserType.clas
CustomEventFirin
IScreenshotListe
ScreenShot.class
ScreenShotRemote
ScreenshotUtil.cl
WebDriverMode.cla
WebUXDriver.clas
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
WebUXDriver
Read-Only
Smart Insert
1 : 1


Quick Access
Java EE
Debug
BaseWebUtil.clas
BrowserType.clas
CustomEventFirin
IScreenshotListe
ScreenShot.class
ScreenShotRemote
ScreenshotUtil.cl
WebDriverMode.cla
WebUXDriver.clas
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
WebUXDriver
}
public String getPlatform() {
return config.getPlatform().name();
}
public String getBrowserVersion() {
return config.getBrowserVersion();
}
public String getChromeBinPath() {
return config.getChromeBinPath();
}
public String getChromeDriverPath() {
return config.getChromeDriverPath();
}
public WebDriverConfig getConfig() {
return config;
}
public int getExplicitWait() {
return config.getExplicitWaitTimeout();
}
public String getFfBinPath() {
return config.getFFBinPath();
}
public String getFfProfilePath() throws URISyntaxException {
return config.getFfProfilePath();
}
public String getOperaProfilePath() throws URISyntaxException {
return config.getOperaProfilePath();
}
public void setOperaProfilePath(String operaProfilePath) {
config.setOperaProfilePath(operaProfilePath);
}
public String getHubUrl() {
return config.getHubUrl();
}
public String getIEDriverPath() {
return config.getIeDriverPath();
}
public double getImplicitWait() {
return config.getImplicitWaitTimeout();
}
public String getMode() {
return config.getMode().name();
}
public String getOutputDirectory() {
return config.getOutputDirectory();
}
public String getNtlmAuthTrustedUris() {
return config.getNtlmAuthTrustedUris();
}
Read-Only
Smart Insert
1 : 1

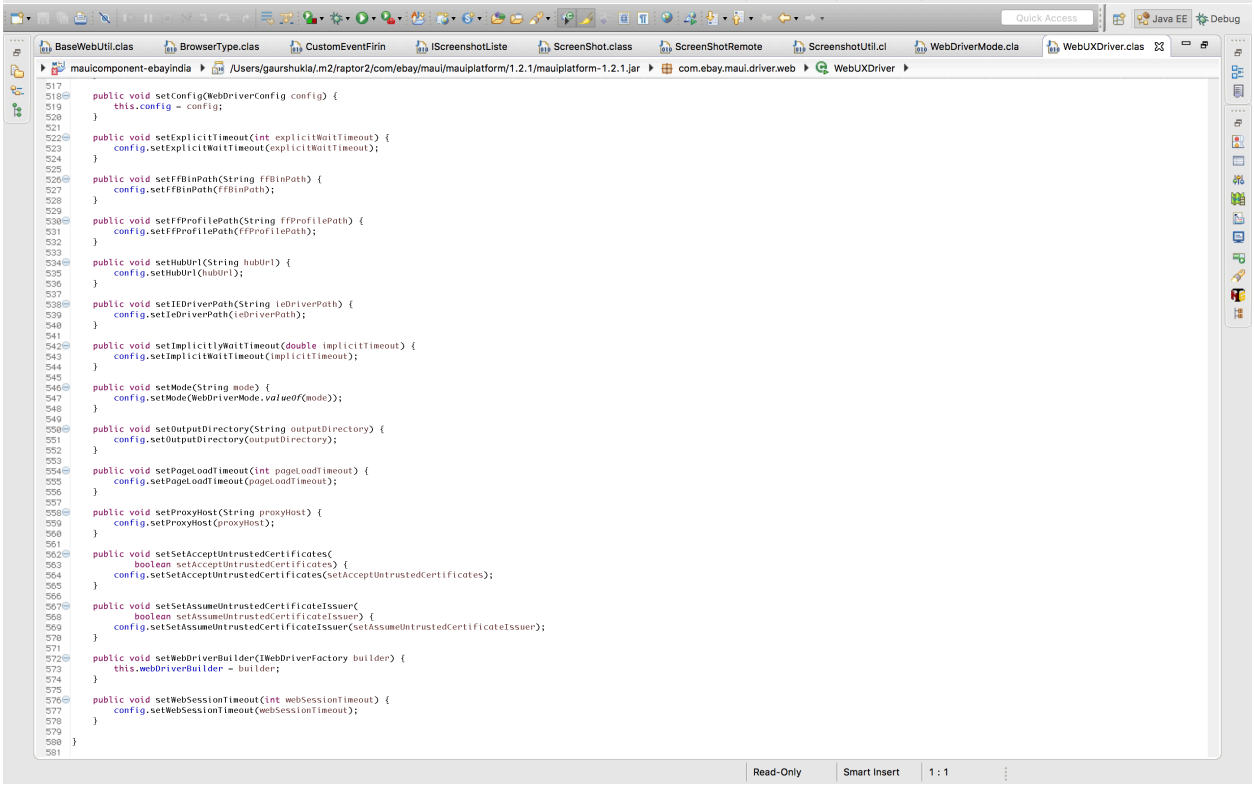

Quick Access
Java EE
Debug
BaseWebUtil.clas
BrowserType.clas
CustomEventFirin
IScreenshotListe
ScreenShot.class
ScreenShotRemote
ScreenshotUtil.cl
WebDriverMode.cla
WebUXDriver.clas
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
WebUXDriver
public void setNtlmAuthTrustedUris(String url) {
config.setNtlmAuthTrustedUris(url);
}
public int getPageLoadTimeout() {
return config.getPageLoadTimeout();
}
public String getProxyHost() {
return config.getProxyHost();
}
public void setUserAgentOverride(String userAgentOverride) {
config.setUserAgentOverride(userAgentOverride);
}
public String getUserAgentOverride() {
return config.getUserAgentOverride();
}
public IWebDriverFactory getWebDriverBuilder() {
return webDriverBuilder;
}
public int getWebSessionTimeout() {
return config.getWebSessionTimeout();
}
private void init() {
if (ContextManager.getThreadContext() == null)
return;
String browser = ContextManager.getThreadContext().getWebRunBrowser();
String browserVersion = null;
if (browser.indexOf("-") > 1) {
try {
browserVersion = browser.split("-")[1];
browser = browser.split("-")[0];
} catch (Exception ex) {
}
} else {
browserVersion = ContextManager.getThreadContext()
.getWebBrowserVersion();
}
config.setBrowser(BrowserType.getBrowserType(browser));
config.setBrowserVersion(browserVersion);
String platform = ContextManager.getThreadContext().getPlatform();
if (platform != null)
config.setPlatform(Platform.valueOf(platform));
String mode = ContextManager.getThreadContext().getWebRunMode();
config.setMode(WebDriverMode.valueOf(mode));
String hubUrl = ContextManager.getThreadContext().getWebDriverGrid();
config.setHubUrl(hubUrl);
String ffProfilePath = ContextManager.getThreadContext()
.getFirefoxUserProfilePath();
config.setFfProfilePath(ffProfilePath);
String operaProfilePath = ContextManager.getThreadContext()
.getOperaUserProfilePath();
config.setOperaProfilePath(operaProfilePath);
String ffBinPath = ContextManager.getThreadContext()
.getFirefoxBinPath();
config.setFfBinPath(ffBinPath);
String chromeBinPath = ContextManager.getThreadContext()
.getChromeBinPath();
Read-Only
Smart Insert
1 : 1


BaseWebUtil.clas
BrowserType.clas
CustomEventFirin
IScreenshotListe
ScreenShot.class
ScreenShotRemote
ScreenshotUtil.cl
WebDriverMode.cla
WebUXDriver.clas
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
WebUXDriver
Quick Access
Java EE
Debug
Read-Only
Smart Insert
1 : 1


BaseWebUtil.clas
BrowserType.clas
CustomEventFirin
IScreenshotListe
ScreenShot.class
ScreenShotRemote
ScreenshotUtil.cl
WebDriverMode.cla
WebUXDriver.clas
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web
WebUXDriver
Quick Access
Java EE
Debug
Read-Only
Smart Insert
1 : 1

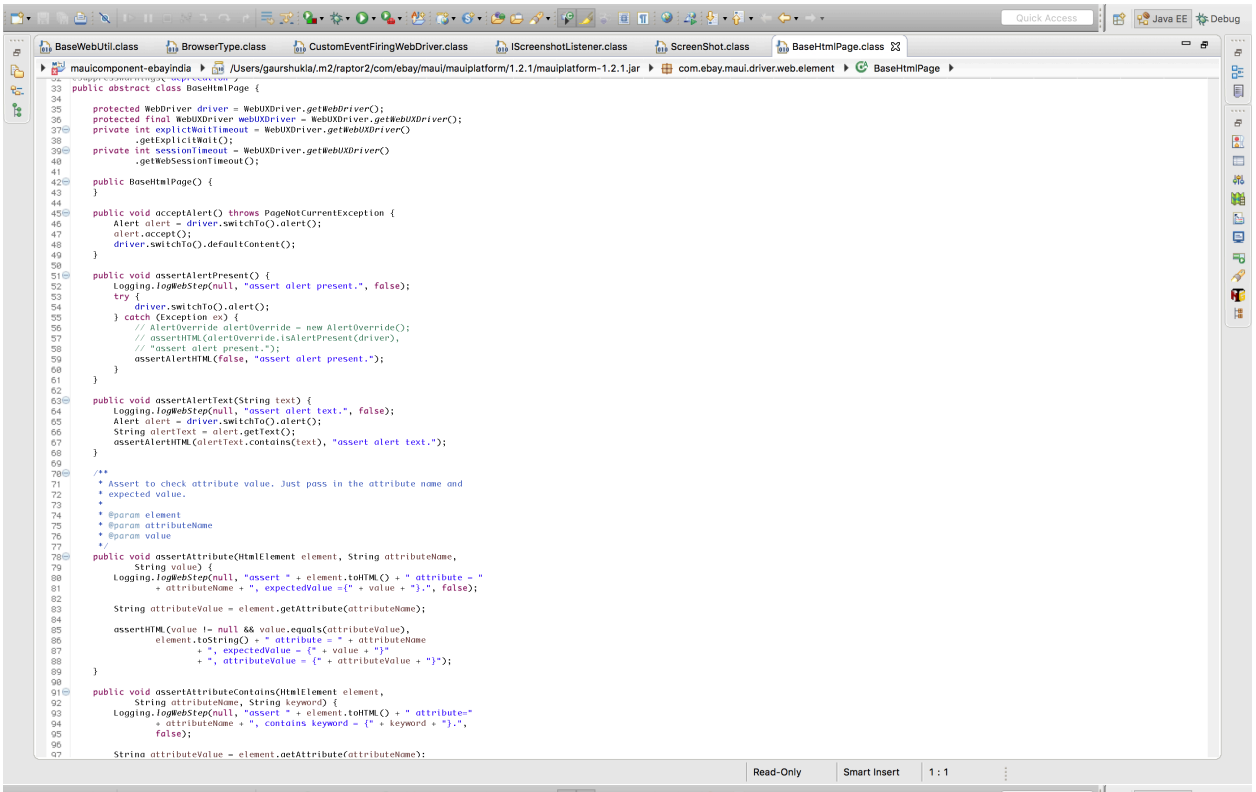


```
    }

    public void setAddJSErrorCollectorExtension(
            Boolean addJSErrorCollectorExtension) {
        config.setAddJSErrorCollectorExtension(addJSErrorCollectorExtension);
    }

    public boolean isSetAssumeUntrustedCertificateIssuer() {
        return config.isSetAssumeUntrustedCertificateIssuer();
    }

    public boolean isEnableJavascript() {
        return config.isEnableJavascript();
    }

    public void setEnableJavascript(Boolean enableJavascript) {
        config.setEnableJavascript(enableJavascript);
    }

    public void setBrowser(String browser) {
        config.setBrowser(BrowserType.getBrowserType(browser));

    }

    public void setBrowserVersion(String browserVersion) {
        config.setBrowserVersion(browserVersion);
    }

    public void setPlatform(String platform) {
        config.setPlatform(Platform.valueOf(platform));
    }

    public void setChromeBinPath(String chromeBinPath) {
        config.setChromeBinPath(chromeBinPath);
    }

    public void setBrowserDownloadDir(String browserDownloadDir) {
        config.setBrowserDownloadDir(browserDownloadDir);
    }

    public String getBrowserDownloadDir() {
        return config.getBrowserDownloadDir();
    }

    public void setChromeDriverPath(String chromeDriverPath) {
        config.setChromeDriverPath(chromeDriverPath);
    }

    public void setConfig(WebDriverConfig config) {
        this.config = config;
    }

    public void setExplicitTimeout(int explicitWaitTimeout) {
        config.setExplicitWaitTimeout(explicitWaitTimeout);
    }

    public void setFfBinPath(String ffBinPath) {
        config.setFfBinPath(ffBinPath);
    }

    public void setFfProfilePath(String ffProfilePath) {
        config.setFfProfilePath(ffProfilePath);
    }

    public void setHubUrl(String hubUrl) {
```

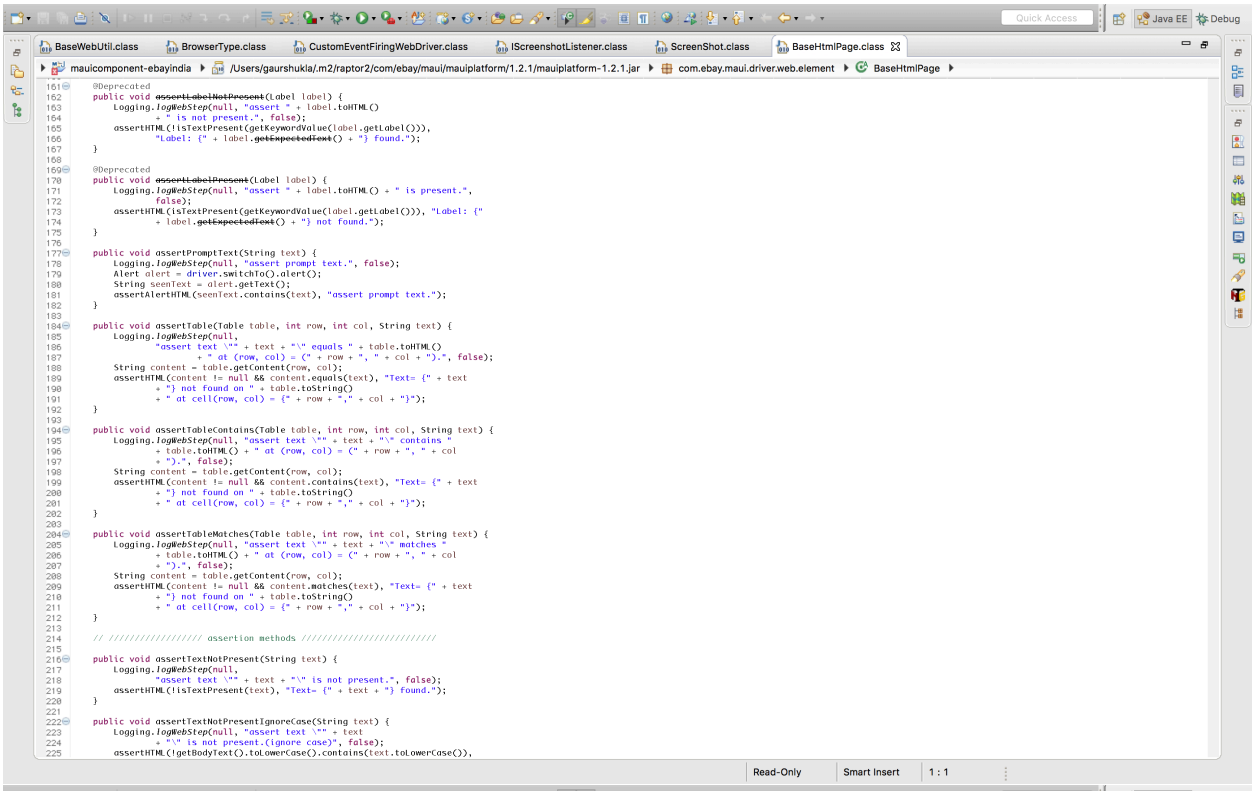


```
    public void setConfig(WebDriverConfig config) {
        this.config = config;
    }

    public void setExplicitTimeout(int explicitWaitTimeout) {
        config.setExplicitWaitTimeout(explicitWaitTimeout);
    }

    public void setFfBinPath(String ffBinPath) {
        config.setFfBinPath(ffBinPath);
    }

    public void setFfProfilePath(String ffProfilePath) {
        config.setFfProfilePath(ffProfilePath);
    }

    public void setHubUrl(String hubUrl) {
        config.setHubUrl(hubUrl);
    }

    public void setIEDriverPath(String ieDriverPath) {
        config.setIeDriverPath(ieDriverPath);
    }

    public void setImplicitlyWaitTimeout(double implicitTimeout) {
        config.setImplicitWaitTimeout(implicitTimeout);
    }

    public void setMode(String mode) {
        config.setMode(WebDriverMode.valueOf(mode));
    }

    public void setOutputDirectory(String outputDirectory) {
        config.setOutputDirectory(outputDirectory);
    }

    public void setPageLoadTimeout(int pageLoadTimeout) {
        config.setPageLoadTimeout(pageLoadTimeout);
    }

    public void setProxyHost(String proxyHost) {
        config.setProxyHost(proxyHost);
    }

    public void setSetAcceptUntrustedCertificates(
            boolean setAcceptUntrustedCertificates) {
        config.setSetAcceptUntrustedCertificates(setAcceptUntrustedCertificates);
    }

    public void setSetAssumeUntrustedCertificateIssuer(
            boolean setAssumeUntrustedCertificateIssuer) {
        config.setSetAssumeUntrustedCertificateIssuer(setAssumeUntrustedCertificateIssuer);
    }

    public void setWebDriverBuilder(IWebDriverFactory builder) {
        this.webDriverBuilder = builder;
    }

    public void setWebSessionTimeout(int webSessionTimeout) {
        config.setWebSessionTimeout(webSessionTimeout);
    }
}
```

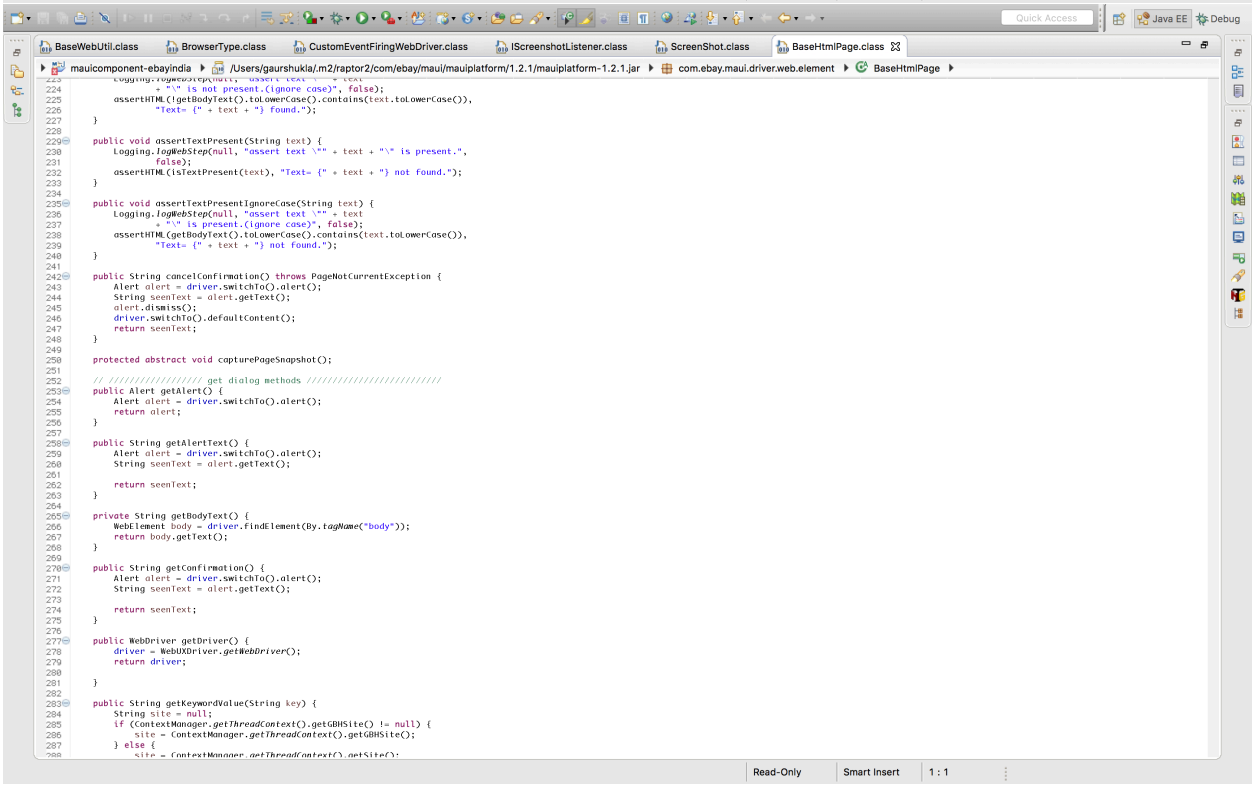
Web.element



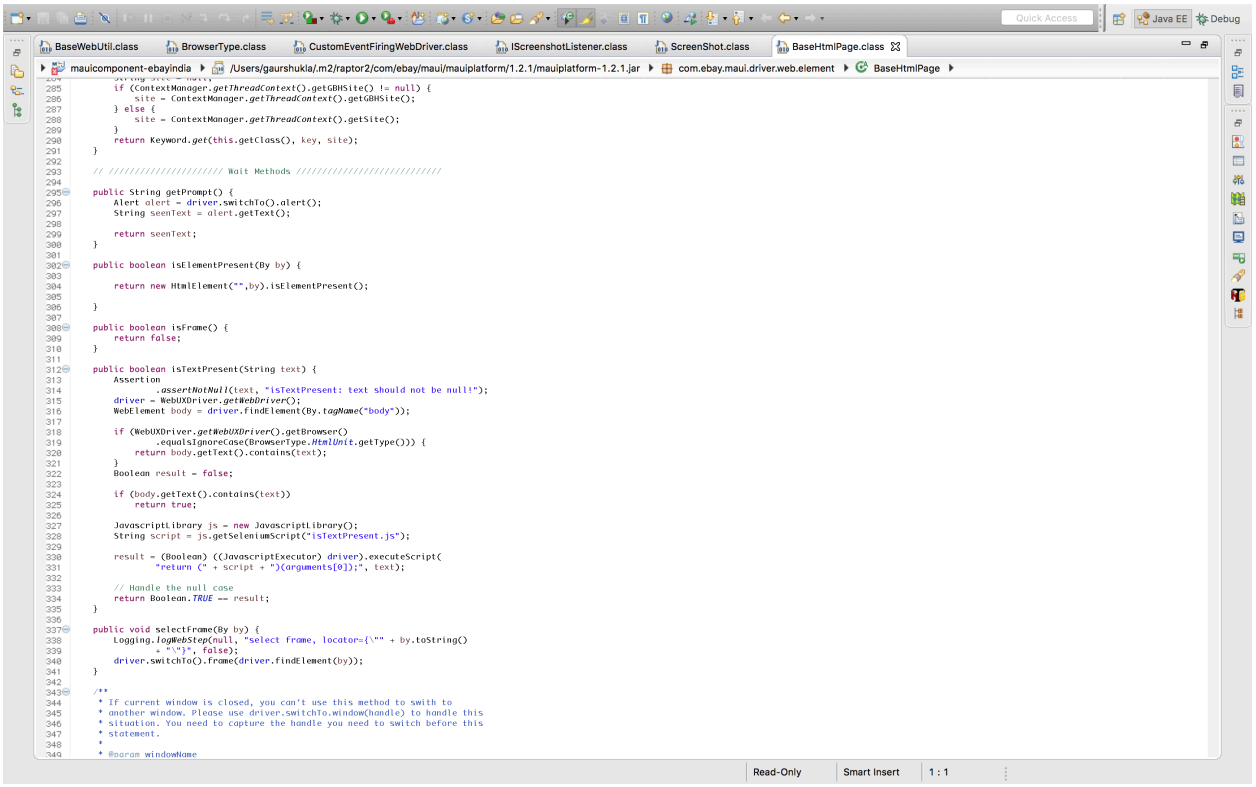

```
public abstract class BaseHtmlPage {

    protected WebDriver driver = WebUXDriver.getWebDriver();
    protected final WebUXDriver webUXDriver = WebUXDriver.getWebUXDriver();
    private int explicitWaitTimeout = WebUXDriver.getWebUXDriver()
            .getExplicitWait();
    private int sessionTimeout = WebUXDriver.getWebUXDriver()
            .getWebSessionTimeout();

    public BaseHtmlPage() {
    }

    public void acceptAlert() throws PageNotCurrentException {
        Alert alert = driver.switchTo().alert();
        alert.accept();
        driver.switchTo().defaultContent();
    }

    public void assertAlertPresent() {
        Logging.logWebStep(null, "assert alert present.", false);
        try {
            driver.switchTo().alert();
        } catch (Exception ex) {
            // AlertOverride alertOverride = new AlertOverride();
            // assertHTML(alertOverride.isAlertPresent(driver),
            // "assert alert present.");
            assertAlertHTML(false, "assert alert present.");
        }
    }

    public void assertAlertText(String text) {
        Logging.logWebStep(null, "assert alert text.", false);
        Alert alert = driver.switchTo().alert();
        String alertText = alert.getText();
        assertAlertHTML(alertText.contains(text), "assert alert text.");
    }

    /**
     * Assert to check attribute value. Just pass in the attribute name and
     * expected value.
     *
     * @param element
     * @param attributeName
     * @param value
     */
    public void assertAttribute(HtmlElement element, String attributeName,
            String value) {
        Logging.logWebStep(null, "assert " + element.toHTML() + " attribute - "
                + attributeName + ", expectedValue ={" + value + "}.", false);

        String attributeValue = element.getAttribute(attributeName);

        assertHTML(value != null && value.equals(attributeValue),
                element.toString() + " attribute = " + attributeName
                        + ", expectedValue = {" + value + "}"
                        + ", attributeValue = {" + attributeValue + "}");
    }

    public void assertAttributeContains(HtmlElement element,
            String attributeName, String keyword) {
        Logging.logWebStep(null, "assert " + element.toHTML() + " attribute="
                + attributeName + ", contains keyword = {" + keyword + "}.",
                false);

        String attributeValue = element.getAttribute(attributeName);

        assertHTML(
                attributeValue != null && keyword != null
                        && attributeValue.contains(keyword), element.toString()
                        + " attribute=" + attributeName
                        + ", expected to contains keyword {" + keyword + "}"
                        + ", attributeValue = {" + attributeValue + "}");
    }

    public void assertAttributeMatches(HtmlElement element,
            String attributeName, String regex) {
        Logging.logWebStep(null, "assert " + element.toHTML() + " attribute="
                + attributeName + ", matches regex = {" + regex + "}.", false);

        String attributeValue = element.getAttribute(attributeName);

        assertHTML(
                attributeValue != null && regex != null
                        && attributeValue.matches(regex), element.toString()
                        + " attribute=" + attributeName
                        + " expected to match regex {" + regex + "}"
                        + ", attributeValue = {" + attributeValue + "}");
    }

    public void assertConfirmationText(String text) {
        Logging.logWebStep(null, "assert confirmation text.", false);
        Alert alert = driver.switchTo().alert();
        String seenText = alert.getText();

        assertAlertHTML(seenText.contains(text), "assert confirmation text.");
    }

    protected void assertCurrentPage(boolean log)
            throws PageNotCurrentException {
        // do nothing
    }

    public void assertElementNotPresent(HtmlElement element) {
        Logging.logWebStep(null, "assert " + element.toHTML()
                + " is not present.", false);
        assertHTML(!element.isElementPresent(), element.toString() + " found.");
    }

    public void assertElementPresent(HtmlElement element) {
        Logging.logWebStep(null, "assert " + element.toHTML() + " is present.",
                false);
        assertHTML(element.isElementPresent(), element.toString()
                + " not found.");
    }

    void assertHTML(boolean condition, String message) {
        if (!condition) {
            capturePageSnapshot();
            Assertion.assertTrue(condition, message);
        }
    }

    void assertAlertHTML(boolean condition, String message) {
        if (!condition) {
            Assertion.assertTrue(condition, message);
        }
    }

    @Deprecated
```



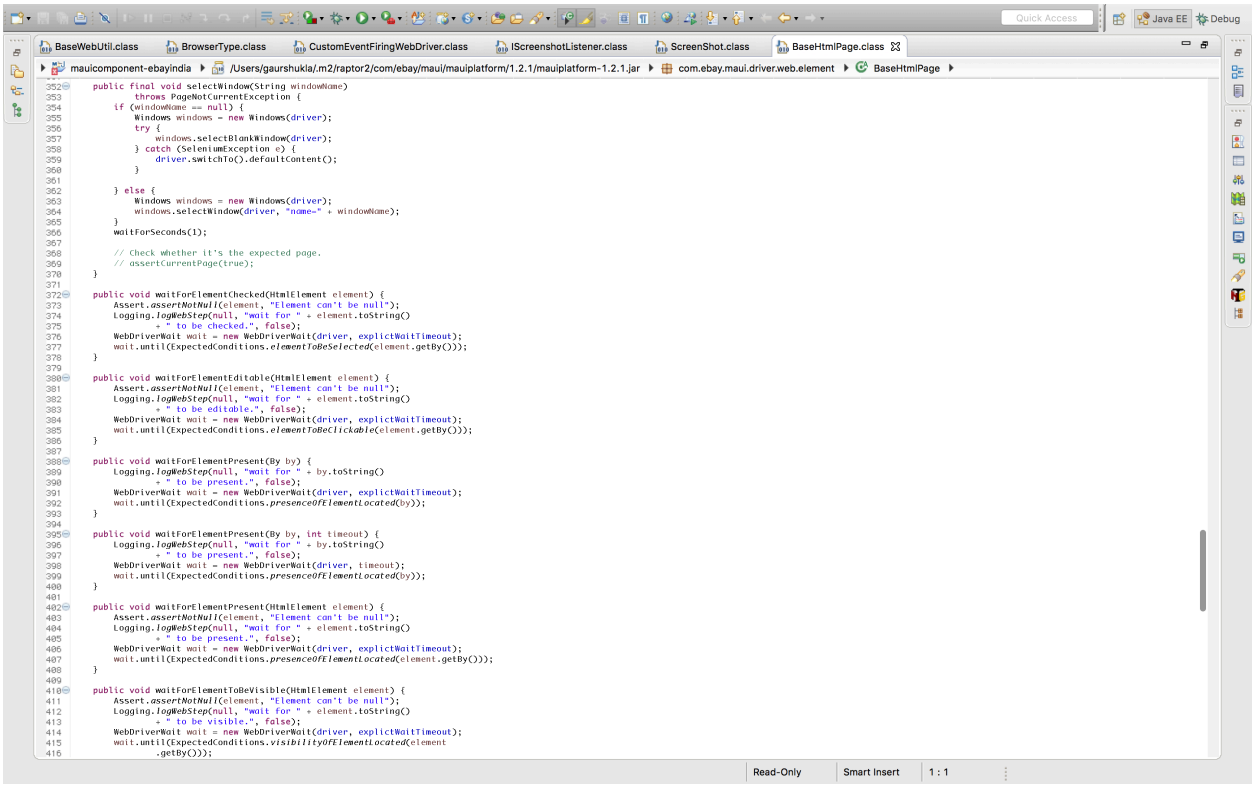

```
    @Deprecated
    public void assertLabelNotPresent(Label label) {
        Logging.logWebStep(null, "assert " + label.toHTML()
                + " is not present.", false);
        assertHTML(!isTextPresent(getKeywordValue(label.getLabel())),
                "Label: {" + label.getExpectedText() + "} found.");
    }

    @Deprecated
    public void assertLabelPresent(Label label) {
        Logging.logWebStep(null, "assert " + label.toHTML() + " is present.",
                false);
        assertHTML(isTextPresent(getKeywordValue(label.getLabel())), "Label: {"
                + label.getExpectedText() + "} not found.");
    }

    public void assertPromptText(String text) {
        Logging.logWebStep(null, "assert prompt text.", false);
        Alert alert = driver.switchTo().alert();
        String seenText = alert.getText();
        assertAlertHTML(seenText.contains(text), "assert prompt text.");
    }

    public void assertTable(Table table, int row, int col, String text) {
        Logging.logWebStep(null,
                "assert text \"" + text + "\" equals " + table.toHTML()
                        + " at (row, col) = (" + row + ", " + col + ").", false);
        String content = table.getContent(row, col);
        assertHTML(content != null && content.equals(text), "Text= {" + text
                + "} not found on " + table.toString()
                + " at cell(row, col) = {" + row + "," + col + "}");
    }

    public void assertTableContains(Table table, int row, int col, String text) {
        Logging.logWebStep(null, "assert text \"" + text + "\" contains "
                + table.toHTML() + " at (row, col) = (" + row + ", " + col
                + ").", false);
        String content = table.getContent(row, col);
        assertHTML(content != null && content.contains(text), "Text= {" + text
                + "} not found on " + table.toString()
                + " at cell(row, col) = {" + row + "," + col + "}");
    }

    public void assertTableMatches(Table table, int row, int col, String text) {
        Logging.logWebStep(null, "assert text \"" + text + "\" matches "
                + table.toHTML() + " at (row, col) = (" + row + ", " + col
                + ").", false);
        String content = table.getContent(row, col);
        assertHTML(content != null && content.matches(text), "Text= {" + text
                + "} not found on " + table.toString()
                + " at cell(row, col) = {" + row + "," + col + "}");
    }

    // ////////////////// assertion methods //////////////////////////

    public void assertTextNotPresent(String text) {
        Logging.logWebStep(null,
                "assert text \"" + text + "\" is not present.", false);
        assertHTML(!isTextPresent(text), "Text= {" + text + "} found.");
    }

    public void assertTextNotPresentIgnoreCase(String text) {
        Logging.logWebStep(null, "assert text \"" + text
                + "\" is not present.(ignore case)", false);
        assertHTML(!getBodyText().toLowerCase().contains(text.toLowerCase()),
```



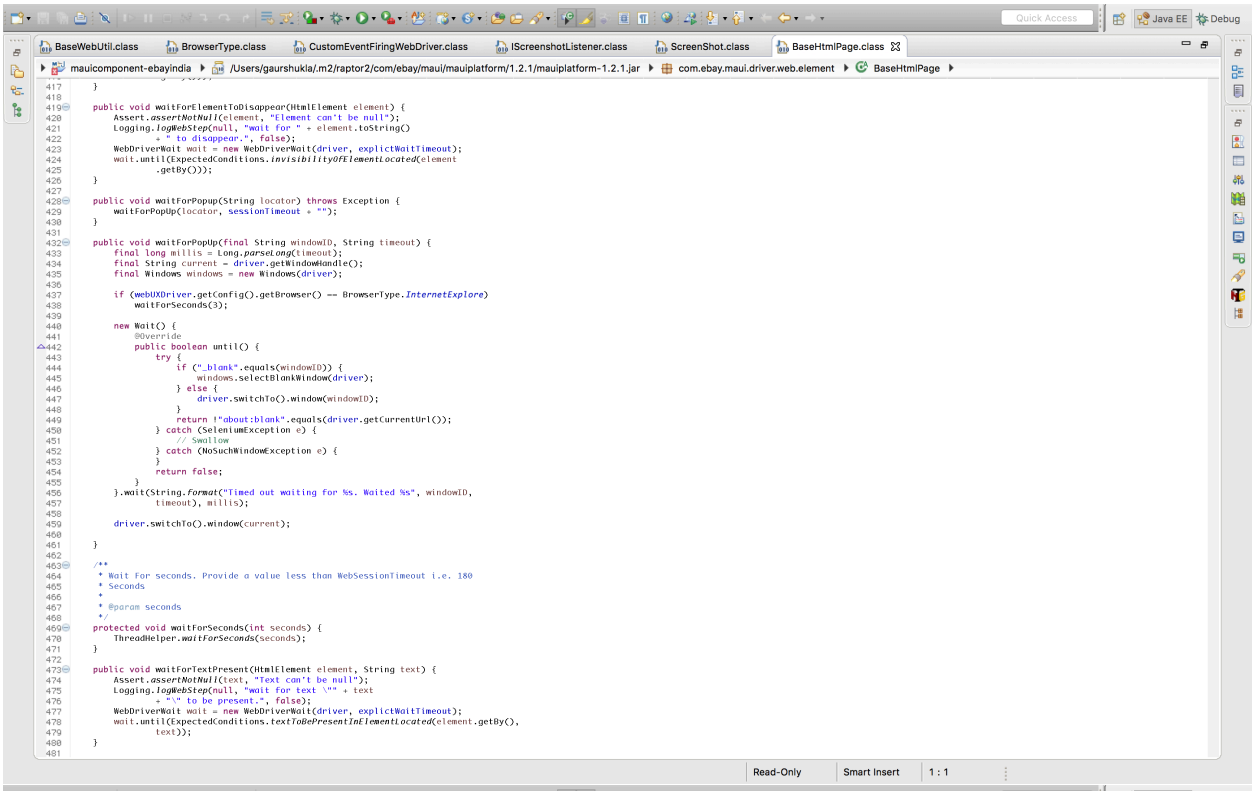

```
                + "\" is not present.(ignore case)", false);
        assertHTML(!getBodyText().toLowerCase().contains(text.toLowerCase()),
                "Text= {" + text + "} found.");
    }

    public void assertTextPresent(String text) {
        Logging.logWebStep(null, "assert text \"" + text + "\" is present.",
                false);
        assertHTML(isTextPresent(text), "Text= {" + text + "} not found.");
    }

    public void assertTextPresentIgnoreCase(String text) {
        Logging.logWebStep(null, "assert text \"" + text
                + "\" is present.(ignore case)", false);
        assertHTML(getBodyText().toLowerCase().contains(text.toLowerCase()),
                "Text= {" + text + "} not found.");
    }

    public String cancelConfirmation() throws PageNotCurrentException {
        Alert alert = driver.switchTo().alert();
        String seenText = alert.getText();
        alert.dismiss();
        driver.switchTo().defaultContent();
        return seenText;
    }

    protected abstract void capturePageSnapshot();

    // ////////////////// get dialog methods //////////////////////////
    public Alert getAlert() {
        Alert alert = driver.switchTo().alert();
        return alert;
    }

    public String getAlertText() {
        Alert alert = driver.switchTo().alert();
        String seenText = alert.getText();

        return seenText;
    }

    private String getBodyText() {
        WebElement body = driver.findElement(By.tagName("body"));
        return body.getText();
    }

    public String getConfirmation() {
        Alert alert = driver.switchTo().alert();
        String seenText = alert.getText();

        return seenText;
    }

    public WebDriver getDriver() {
        driver = WebUIDriver.getWebDriver();
        return driver;

    }

    public String getKeywordValue(String key) {
        String site = null;
        if (ContextManager.getThreadContext().getGBHSite() != null) {
            site = ContextManager.getThreadContext().getGBHSite();
        } else {
```



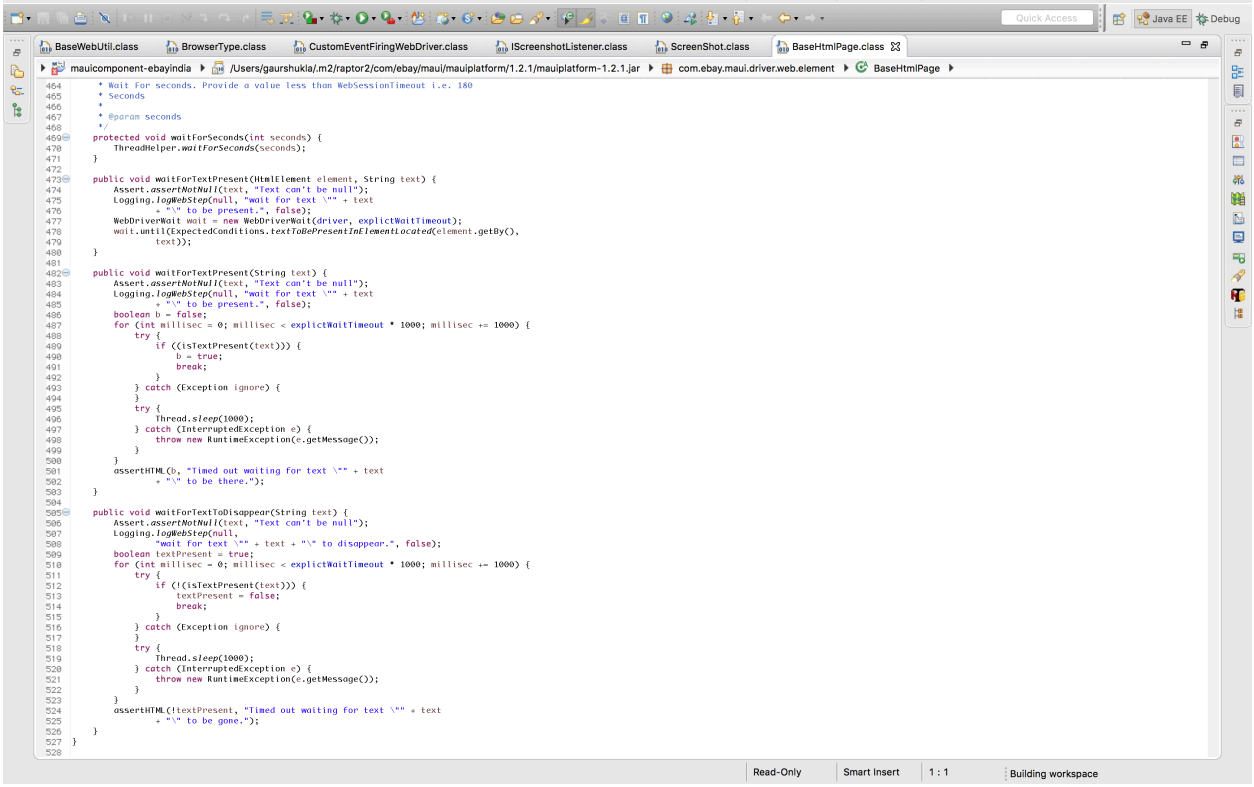
```
        if (ContextManager.getThreadContext().getGBHSite() != null) {
            site = ContextManager.getThreadContext().getGBHSite();
        } else {
            site = ContextManager.getThreadContext().getSite();
        }
        return Keyword.get(this.getClass(), key, site);
    }

    // ////////////////////// Wait Methods //////////////////////////

    public String getPrompt() {
        Alert alert = driver.switchTo().alert();
        String seenText = alert.getText();

        return seenText;
    }

    public boolean isElementPresent(By by) {

        return new HtmlElement("",by).isElementPresent();

    }

    public boolean isFrame() {
        return false;
    }

    public boolean isTextPresent(String text) {
        Assertion
                .assertNotNull(text, "isTextPresent: text should not be null!");
        driver = WebUXDriver.getWebDriver();
        WebElement body = driver.findElement(By.tagName("body"));

        if (WebUXDriver.getWebUXDriver().getBrowser()
                .equalsIgnoreCase(BrowserType.HtmlUnit.getType())) {
            return body.getText().contains(text);
        }
        Boolean result = false;

        if (body.getText().contains(text))
            return true;

        JavascriptLibrary js = new JavascriptLibrary();
        String script = js.getSeleniumScript("isTextPresent.js");

        result = (Boolean) ((JavascriptExecutor) driver).executeScript(
                "return (" + script + ")(arguments[0]);", text);

        // Handle the null case
        return Boolean.TRUE == result;
    }

    public void selectFrame(By by) {
        Logging.logWebStep(null, "select frame, locator={\"" + by.toString()
                + "\"}", false);
        driver.switchTo().frame(driver.findElement(by));
    }

    /**
     * If current window is closed, you can't use this method to swith to
     * another window. Please use driver.switchTo.window(handle) to handle this
     * situation. You need to capture the handle you need to switch before this
     * statement.
     *
     * @param windowName
```

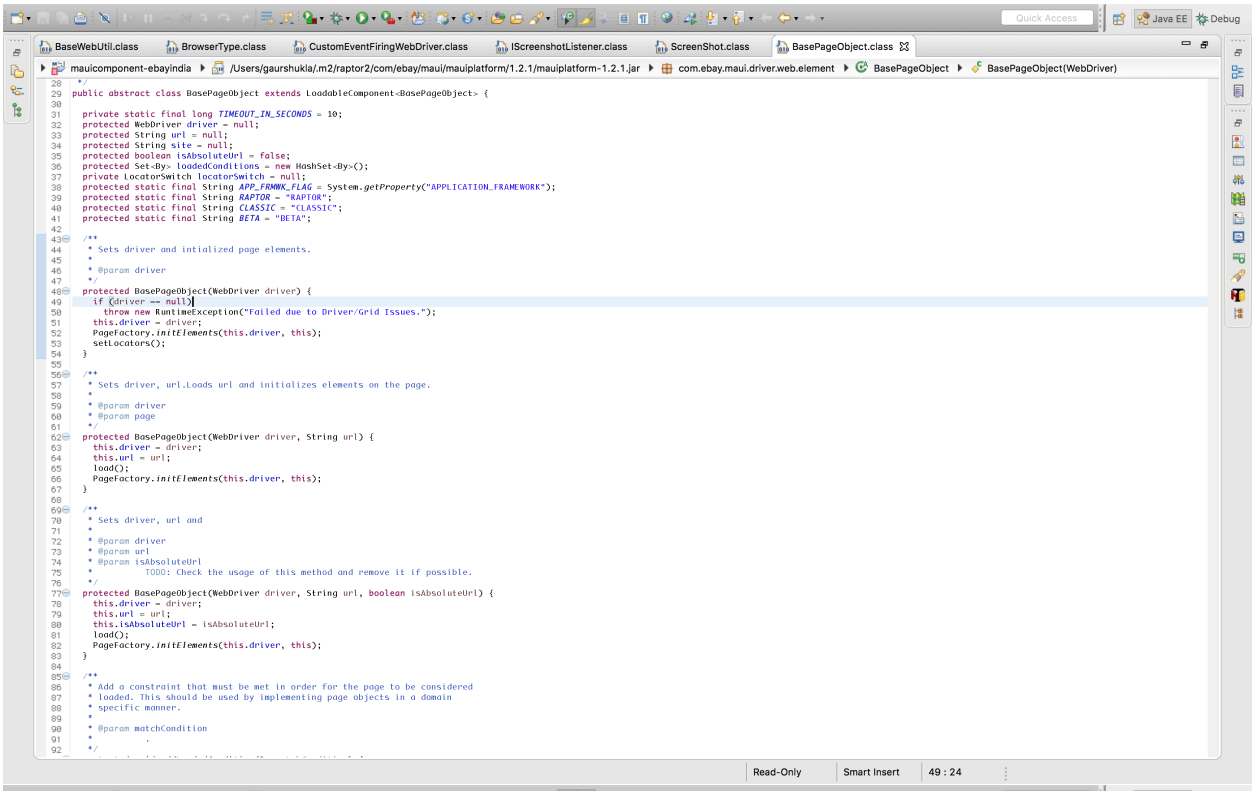
```
    public final void selectWindow(String windowName)
            throws PageNotCurrentException {
        if (windowName == null) {
            Windows windows = new Windows(driver);
            try {
                windows.selectBlankWindow(driver);
            } catch (SeleniumException e) {
                driver.switchTo().defaultContent();
            }

        } else {
            Windows windows = new Windows(driver);
            windows.selectWindow(driver, "name=" + windowName);
        }
        waitForSeconds(1);

        // Check whether it's the expected page.
        // assertCurrentPage(true);
    }

    public void waitForElementChecked(HtmlElement element) {
        Assert.assertNotNull(element, "Element can't be null");
        Logging.logWebStep(null, "wait for " + element.toString()
                + " to be checked.", false);
        WebDriverWait wait = new WebDriverWait(driver, explictWaitTimeout);
        wait.until(ExpectedConditions.elementToBeSelected(element.getBy()));
    }

    public void waitForElementEditable(HtmlElement element) {
        Assert.assertNotNull(element, "Element can't be null");
        Logging.logWebStep(null, "wait for " + element.toString()
                + " to be editable.", false);
        WebDriverWait wait = new WebDriverWait(driver, explictWaitTimeout);
        wait.until(ExpectedConditions.elementToBeClickable(element.getBy()));
    }

    public void waitForElementPresent(By by) {
        Logging.logWebStep(null, "wait for " + by.toString()
                + " to be present.", false);
        WebDriverWait wait = new WebDriverWait(driver, explictWaitTimeout);
        wait.until(ExpectedConditions.presenceOfElementLocated(by));
    }

    public void waitForElementPresent(By by, int timeout) {
        Logging.logWebStep(null, "wait for " + by.toString()
                + " to be present.", false);
        WebDriverWait wait = new WebDriverWait(driver, timeout);
        wait.until(ExpectedConditions.presenceOfElementLocated(by));
    }

    public void waitForElementPresent(HtmlElement element) {
        Assert.assertNotNull(element, "Element can't be null");
        Logging.logWebStep(null, "wait for " + element.toString()
                + " to be present.", false);
        WebDriverWait wait = new WebDriverWait(driver, explictWaitTimeout);
        wait.until(ExpectedConditions.presenceOfElementLocated(element.getBy()));
    }

    public void waitForElementToBeVisible(HtmlElement element) {
        Assert.assertNotNull(element, "Element can't be null");
        Logging.logWebStep(null, "wait for " + element.toString()
                + " to be visible.", false);
        WebDriverWait wait = new WebDriverWait(driver, explictWaitTimeout);
        wait.until(ExpectedConditions.visibilityOfElementLocated(element
                .getBy()));
```



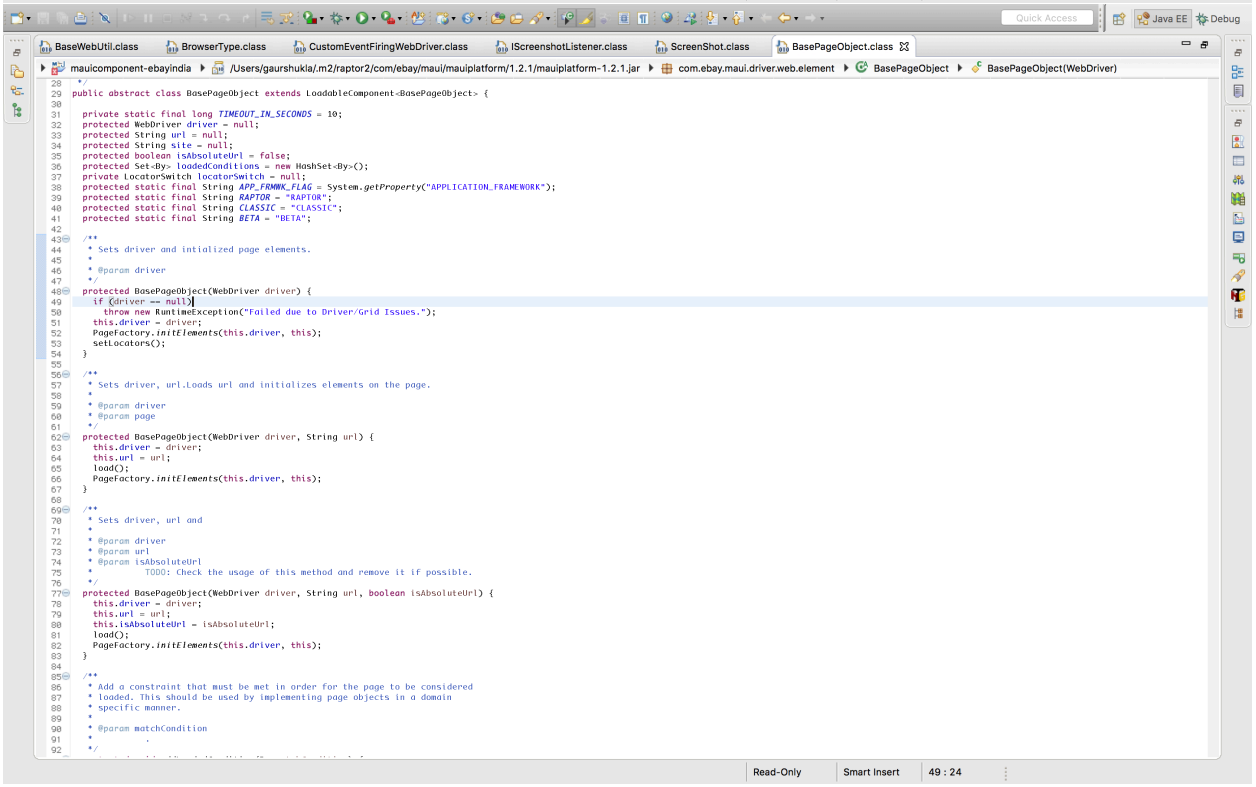
```
    }

    public void waitForElementToDisappear(HtmlElement element) {
        Assert.assertNotNull(element, "Element can't be null");
        Logging.logWebStep(null, "wait for " + element.toString()
                + " to disappear.", false);
        WebDriverWait wait = new WebDriverWait(driver, explictWaitTimeout);
        wait.until(ExpectedConditions.invisibilityOfElementLocated(element
                .getBy()));
    }

    public void waitForPopup(String locator) throws Exception {
        waitForPopUp(locator, sessionTimeout + "");
    }

    public void waitForPopUp(final String windowID, String timeout) {
        final long millis = Long.parseLong(timeout);
        final String current = driver.getWindowHandle();
        final Windows windows = new Windows(driver);

        if (webUXDriver.getConfig().getBrowser() == BrowserType.InternetExplore)
            waitForSeconds(3);

        new Wait() {
            @Override
            public boolean until() {
                try {
                    if ("_blank".equals(windowID)) {
                        windows.selectBlankWindow(driver);
                    } else {
                        driver.switchTo().window(windowID);
                    }
                    return !"about:blank".equals(driver.getCurrentUrl());
                } catch (SeleniumException e) {
                    // Swallow
                } catch (NoSuchWindowException e) {
                }
                return false;
            }
        }.wait(String.format("Timed out waiting for %s. Waited %s", windowID,
                timeout), millis);

        driver.switchTo().window(current);

    }

    /**
     * Wait For seconds. Provide a value less than WebSessionTimeout i.e. 180
     * Seconds
     *
     * @param seconds
     */
    protected void waitForSeconds(int seconds) {
        ThreadHelper.waitForSeconds(seconds);
    }

    public void waitForTextPresent(HtmlElement element, String text) {
        Assert.assertNotNull(text, "Text can't be null");
        Logging.logWebStep(null, "wait for text \"" + text
                + "\" to be present.", false);
        WebDriverWait wait = new WebDriverWait(driver, explictWaitTimeout);
        wait.until(ExpectedConditions.textToBePresentInElementLocated(element.getBy(),
                text));
    }
```



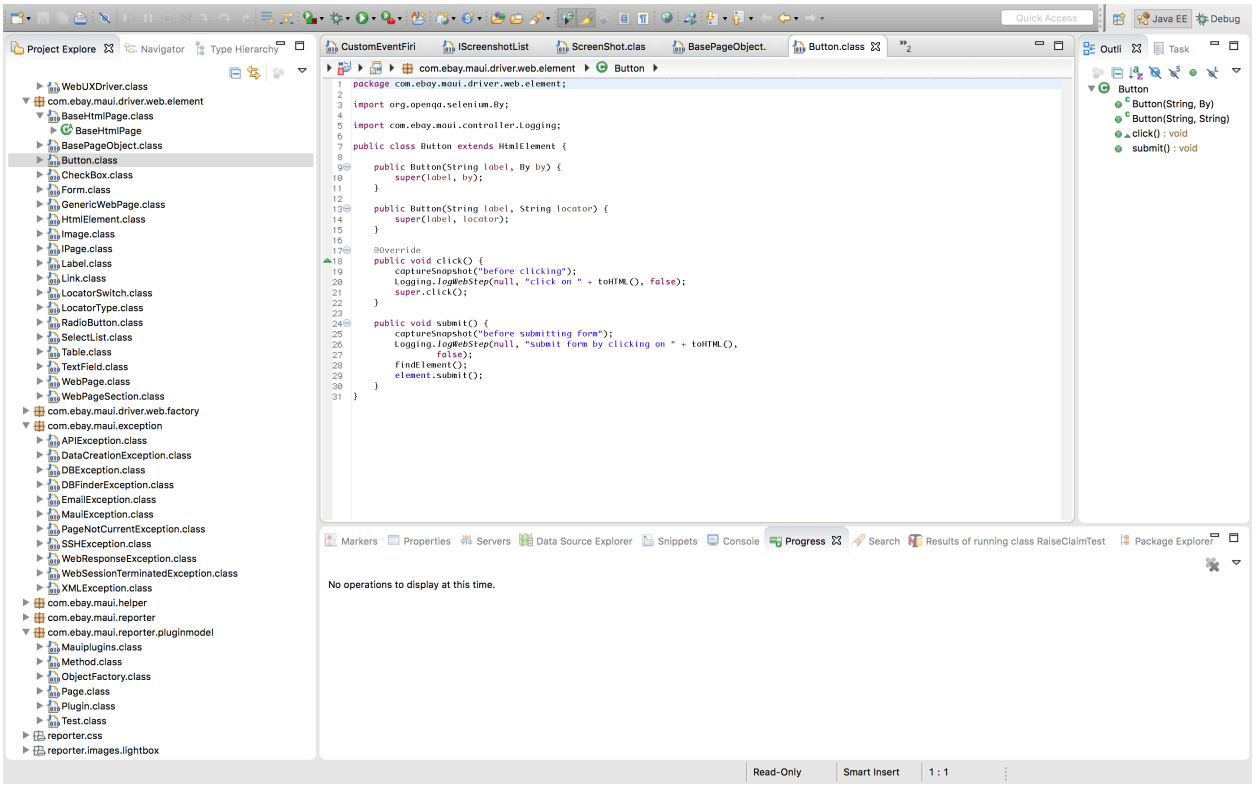
```
     * Wait For seconds. Provide a value less than WebSessionTimeout i.e. 180
     * Seconds
     *
     * @param seconds
     */
    protected void waitForSeconds(int seconds) {
        ThreadHelper.waitForSeconds(seconds);
    }

    public void waitForTextPresent(HtmlElement element, String text) {
        Assert.assertNotNull(text, "Text can't be null");
        Logging.logWebStep(null, "wait for text \"" + text
                + "\" to be present.", false);
        WebDriverWait wait = new WebDriverWait(driver, explictWaitTimeout);
        wait.until(ExpectedConditions.textToBePresentInElementLocated(element.getBy(),
                text));
    }

    public void waitForTextPresent(String text) {
        Assert.assertNotNull(text, "Text can't be null");
        Logging.logWebStep(null, "wait for text \"" + text
                + "\" to be present.", false);
        boolean b = false;
        for (int millisec = 0; millisec < explictWaitTimeout * 1000; millisec += 1000) {
            try {
                if ((isTextPresent(text))) {
                    b = true;
                    break;
                }
            } catch (Exception ignore) {
            }
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e.getMessage());
            }
        }
        assertHTML(b, "Timed out waiting for text \"" + text
                + "\" to be there.");
    }

    public void waitForTextToDisappear(String text) {
        Assert.assertNotNull(text, "Text can't be null");
        Logging.logWebStep(null,
                "wait for text \"" + text + "\" to disappear.", false);
        boolean textPresent = true;
        for (int millisec = 0; millisec < explictWaitTimeout * 1000; millisec += 1000) {
            try {
                if (!(isTextPresent(text))) {
                    textPresent = false;
                    break;
                }
            } catch (Exception ignore) {
            }
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e.getMessage());
            }
        }
        assertHTML(!textPresent, "Timed out waiting for text \"" + text
                + "\" to be gone.");
    }
}
```

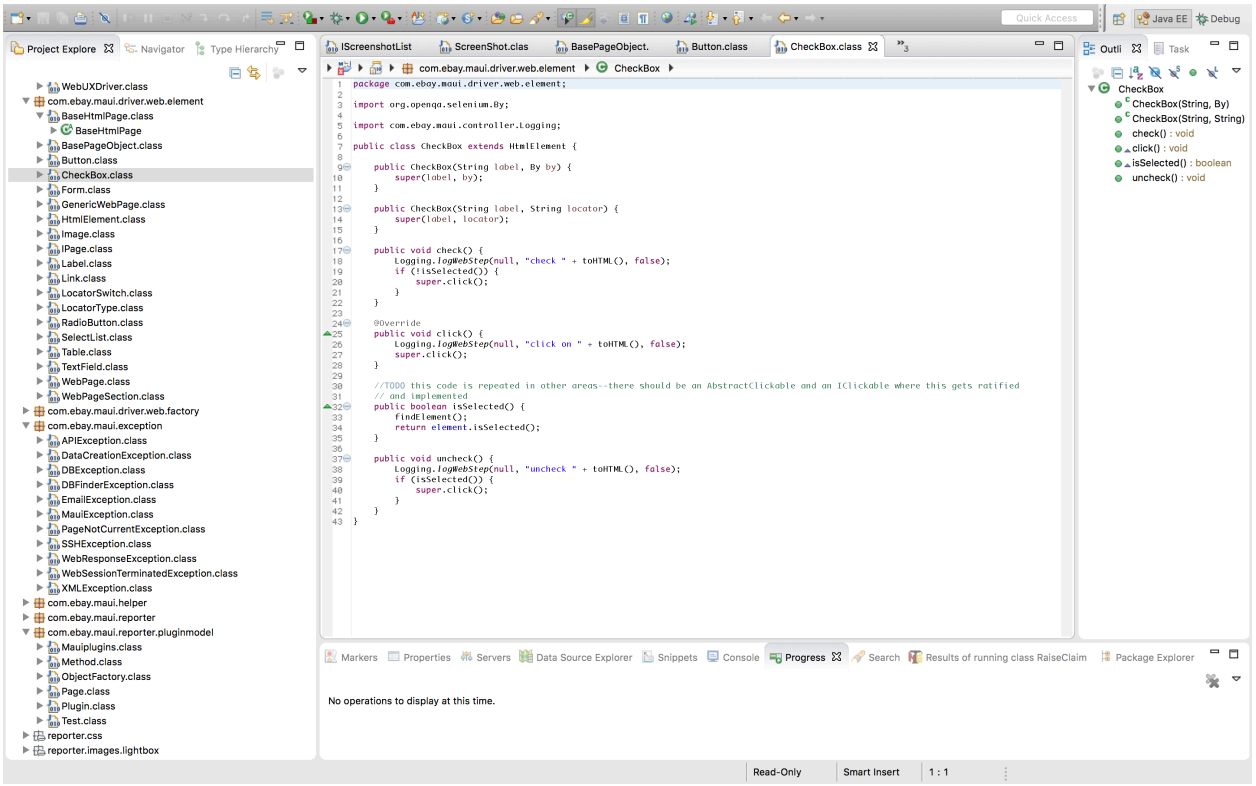


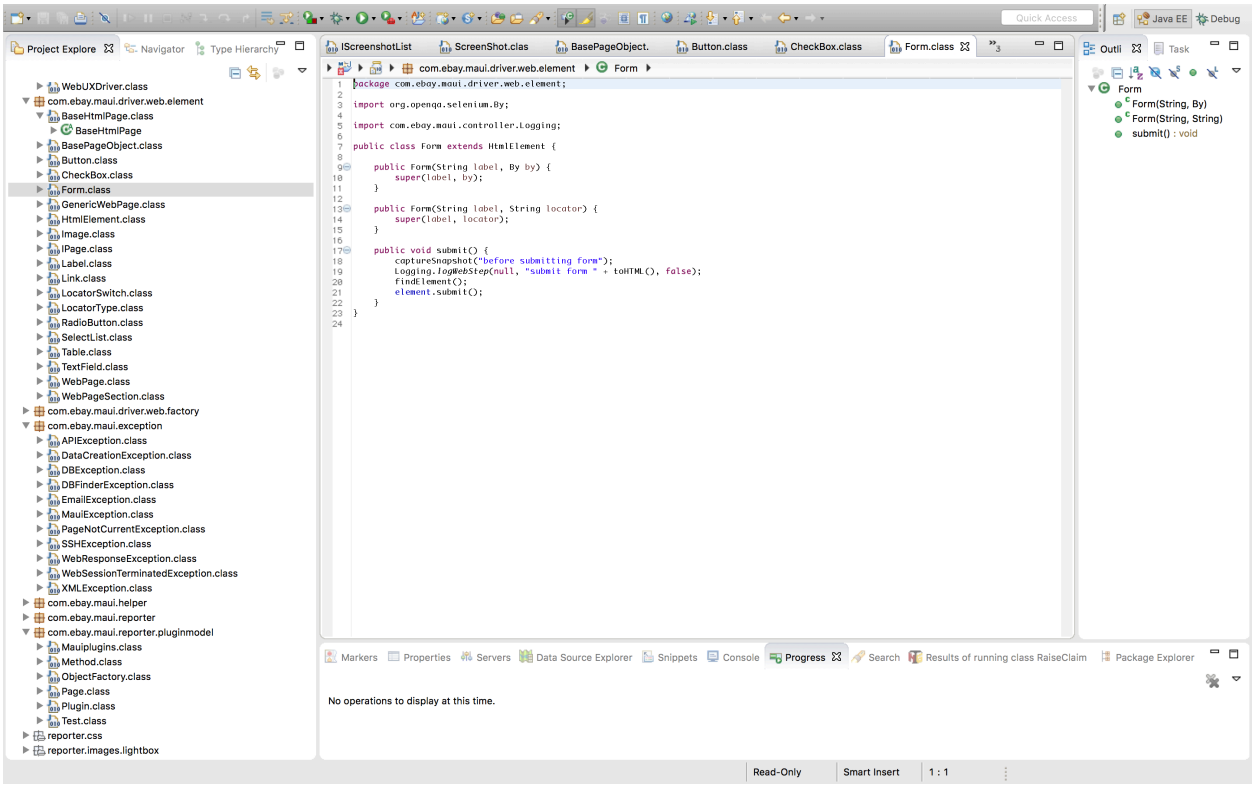

```
 */
public abstract class BasePageObject extends LoadableComponent<BasePageObject> {

  private static final long TIMEOUT_IN_SECONDS = 10;
  protected WebDriver driver = null;
  protected String url = null;
  protected String site = null;
  protected boolean isAbsoluteUrl = false;
  protected Set<By> loadedConditions = new HashSet<By>();
  private LocatorSwitch locatorSwitch = null;
  protected static final String APP_FRMWK_FLAG = System.getProperty("APPLICATION_FRAMEWORK");
  protected static final String RAPTOR = "RAPTOR";
  protected static final String CLASSIC = "CLASSIC";
  protected static final String BETA = "BETA";

  /**
   * Sets driver and intialized page elements.
   *
   * @param driver
   */
  protected BasePageObject(WebDriver driver) {
    if (driver == null)
      throw new RuntimeException("Failed due to Driver/Grid Issues.");
    this.driver = driver;
    PageFactory.initElements(this.driver, this);
    setLocators();
  }

  /**
   * Sets driver, url.Loads url and initializes elements on the page.
   *
   * @param driver
   * @param page
   */
  protected BasePageObject(WebDriver driver, String url) {
    this.driver = driver;
    this.url = url;
    load();
    PageFactory.initElements(this.driver, this);
  }

  /**
   * Sets driver, url and
   *
   * @param driver
   * @param url
   * @param isAbsoluteUrl
   *          TODO: Check the usage of this method and remove it if possible.
   */
  protected BasePageObject(WebDriver driver, String url, boolean isAbsoluteUrl) {
    this.driver = driver;
    this.url = url;
    this.isAbsoluteUrl = isAbsoluteUrl;
    load();
    PageFactory.initElements(this.driver, this);
  }

  /**
   * Add a constraint that must be met in order for the page to be considered
   * loaded. This should be used by implementing page objects in a domain
   * specific manner.
   *
   * @param matchCondition
   *          .
   */
```

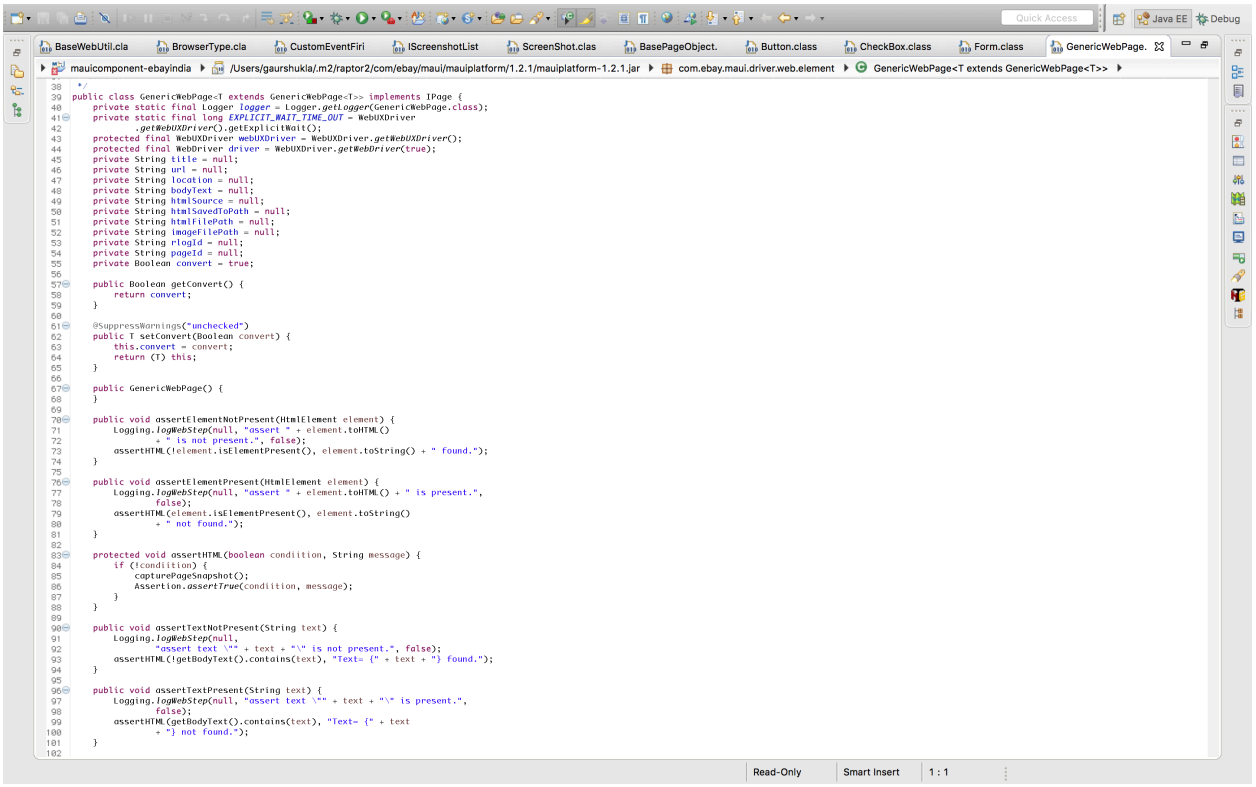


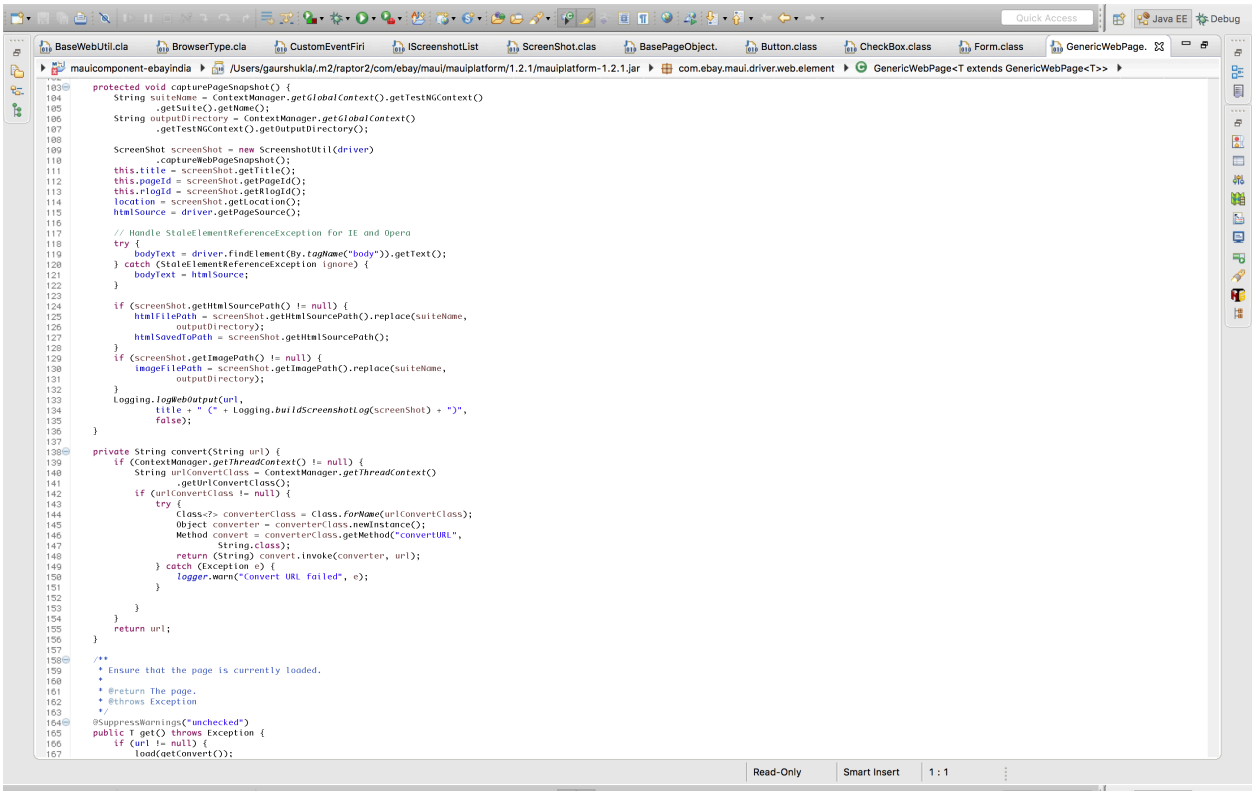

```
 */
public abstract class BasePageObject extends LoadableComponent<BasePageObject> {

  private static final long TIMEOUT_IN_SECONDS = 10;
  protected WebDriver driver = null;
  protected String url = null;
  protected String site = null;
  protected boolean isAbsoluteUrl = false;
  protected Set<By> loadedConditions = new HashSet<By>();
  private LocatorSwitch locatorSwitch = null;
  protected static final String APP_FRMWK_FLAG = System.getProperty("APPLICATION_FRAMEWORK");
  protected static final String RAPTOR = "RAPTOR";
  protected static final String CLASSIC = "CLASSIC";
  protected static final String BETA = "BETA";

  /**
   * Sets driver and intialized page elements.
   *
   * @param driver
   */
  protected BasePageObject(WebDriver driver) {
    if (driver == null)
      throw new RuntimeException("Failed due to Driver/Grid Issues.");
    this.driver = driver;
    PageFactory.initElements(this.driver, this);
    setLocators();
  }

  /**
   * Sets driver, url.Loads url and initializes elements on the page.
   *
   * @param driver
   * @param page
   */
  protected BasePageObject(WebDriver driver, String url) {
    this.driver = driver;
    this.url = url;
    load();
    PageFactory.initElements(this.driver, this);
  }

  /**
   * Sets driver, url and
   *
   * @param driver
   * @param url
   * @param isAbsoluteUrl
   *          TODO: Check the usage of this method and remove it if possible.
   */
  protected BasePageObject(WebDriver driver, String url, boolean isAbsoluteUrl) {
    this.driver = driver;
    this.url = url;
    this.isAbsoluteUrl = isAbsoluteUrl;
    load();
    PageFactory.initElements(this.driver, this);
  }

  /**
   * Add a constraint that must be met in order for the page to be considered
   * loaded. This should be used by implementing page objects in a domain
   * specific manner.
   *
   * @param matchCondition
   *          .
   */
```


Project Explore
Navigator
Type Hierarchy
WebUXDriver.class
com.ebay.maui.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
com.ebay.maui.driver.web.factory
com.ebay.maui.exception
APIException.class
DataCreationException.class
DBException.class
DBFinderException.class
EmailException.class
MauiException.class
PageNotCurrentException.class
SSHException.class
WebResponseException.class
WebSessionTerminatedException.class
XMLException.class
com.ebay.maui.helper
com.ebay.maui.reporter
com.ebay.maui.reporter.pluginmodel
Mauiplugins.class
Method.class
ObjectFactory.class
Page.class
Plugin.class
Test.class
reporter.css
reporter.images.lightbox
CustomEventFiri
IScreenshotList
ScreenShot.clas
BasePageObject.
Button.class
com.ebay.maui.driver.web.element
Button
package com.ebay.maui.driver.web.element;
import org.openqa.selenium.By;
import com.ebay.maui.controller.Logging;
public class Button extends HtmlElement {
public Button(String label, By by) {
super(label, by);
}
public Button(String label, String locator) {
super(label, locator);
}
@Override
public void click() {
captureSnapshot("before clicking");
Logging.logWebStep(null, "click on " + toHTML(), false);
super.click();
}
public void submit() {
captureSnapshot("before submitting form");
Logging.logWebStep(null, "submit form by clicking on " + toHTML(),
false);
findElement();
element.submit();
}
}
Outli
Task
Button
Button(String, By)
Button(String, String)
click() : void
submit() : void
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

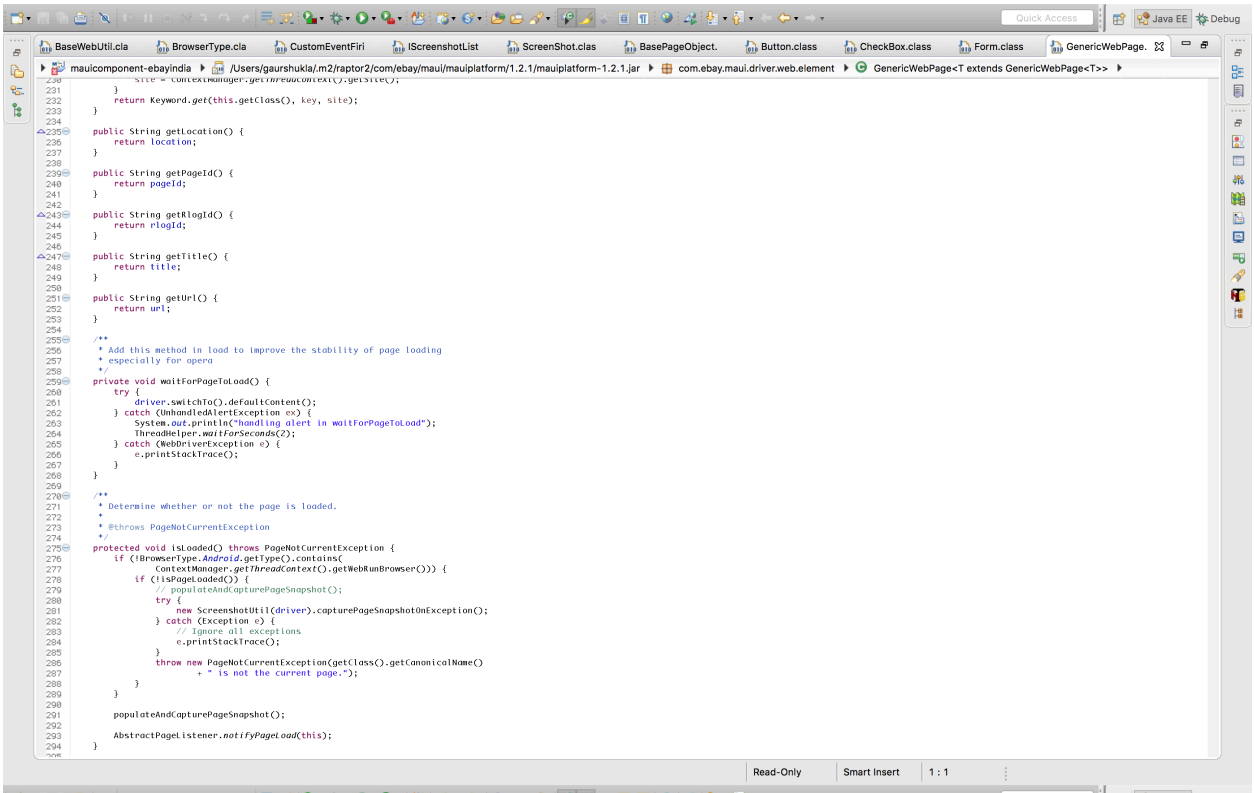

Project Explore
Navigator
Type Hierarchy
WebUXDriver.class
com.ebay.maui.driver.web.element
BaseHtmlPage.class
BaseHtmlPage
BasePageObject.class
Button.class
CheckBox.class
Form.class
GenericWebPage.class
HtmlElement.class
Image.class
IPage.class
Label.class
Link.class
LocatorSwitch.class
LocatorType.class
RadioButton.class
SelectList.class
Table.class
TextField.class
WebPage.class
WebPageSection.class
com.ebay.maui.driver.web.factory
com.ebay.maui.exception
APIException.class
DataCreationException.class
DBException.class
DBFinderException.class
EmailException.class
MauiException.class
PageNotCurrentException.class
SSHException.class
WebResponseException.class
WebSessionTerminatedException.class
XMLException.class
com.ebay.maui.helper
com.ebay.maui.reporter
com.ebay.maui.reporter.pluginmodel
Mauiplugins.class
Method.class
ObjectFactory.class
Page.class
Plugin.class
Test.class
reporter.css
reporter.images.lightbox
IScreenshotList
ScreenShot.clas
BasePageObject.
Button.class
CheckBox.class
com.ebay.maui.driver.web.element
CheckBox
package com.ebay.maui.driver.web.element;
import org.openqa.selenium.By;
import com.ebay.maui.controller.Logging;
public class CheckBox extends HtmlElement {
public CheckBox(String label, By by) {
super(label, by);
}
public CheckBox(String label, String locator) {
super(label, locator);
}
public void check() {
Logging.logWebStep(null, "check " + toHTML(), false);
if (!isSelected()) {
super.click();
}
}
@Override
public void click() {
Logging.logWebStep(null, "click on " + toHTML(), false);
super.click();
}
//TODO this code is repeated in other areas--there should be an AbstractClickable and an IClickable where this gets ratified
// and implemented
public boolean isSelected() {
findElement();
return element.isSelected();
}
public void uncheck() {
Logging.logWebStep(null, "uncheck " + toHTML(), false);
if (isSelected()) {
super.click();
}
}
}
Outli
Task
CheckBox
CheckBox(String, By)
CheckBox(String, String)
check() : void
click() : void
isSelected() : boolean
uncheck() : void
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


Project Explore
Navigator
Type Hierarchy
WebUXDriver.class
com.ebay.maui.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
com.ebay.maui.driver.web.factory
com.ebay.maui.exception
APIException.class
DataCreationException.class
DBException.class
DBFinderException.class
EmailException.class
MauiException.class
PageNotCurrentException.class
SSHException.class
WebResponseException.class
WebSessionTerminatedException.class
XMLException.class
com.ebay.maui.helper
com.ebay.maui.reporter
com.ebay.maui.reporter.pluginmodel
Mauiplugins.class
Method.class
ObjectFactory.class
Page.class
Plugin.class
Test.class
reporter.css
reporter.images.lightbox
IScreenshotList
ScreenShot.clas
BasePageObject.
Button.class
CheckBox.class
Form.class
com.ebay.maui.driver.web.element
Form
package com.ebay.maui.driver.web.element;
import org.openqa.selenium.By;
import com.ebay.maui.controller.Logging;
public class Form extends HtmlElement {
public Form(String label, By by) {
super(label, by);
}
public Form(String label, String locator) {
super(label, locator);
}
public void submit() {
captureSnapshot("before submitting form");
Logging.logWebStep(null, "submit form " + toHTML(), false);
findElement();
element.submit();
}
}
Outli
Task
Form
Form(String, By)
Form(String, String)
submit() : void
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

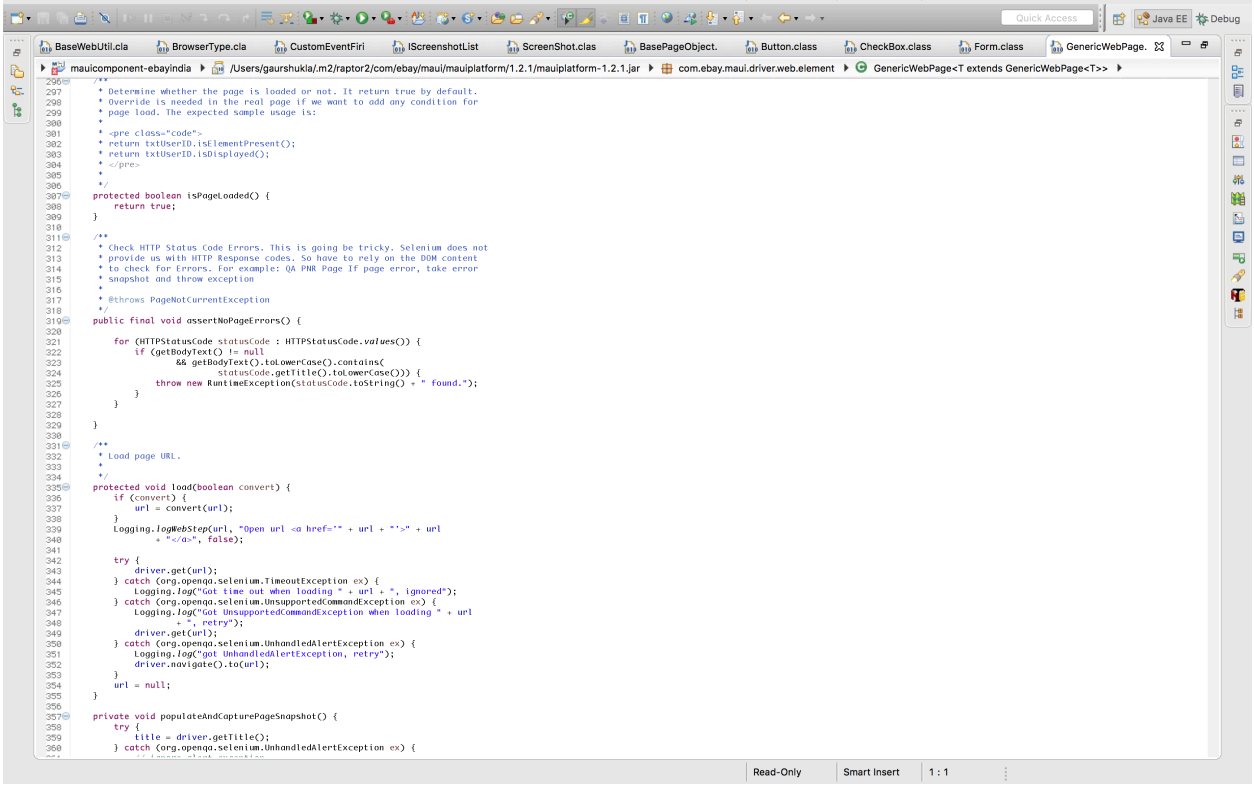

BaseWebUtil.cla
BrowserType.cla
CustomEventFiri
IScreenshotList
ScreenShot.clas
BasePageObject.
Button.class
CheckBox.class
Form.class
GenericWebPage.
maucomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web.element
GenericWebPage<T extends GenericWebPage<T>>
*/
public class GenericWebPage<T extends GenericWebPage<T>> implements IPage {
private static final Logger logger = Logger.getLogger(GenericWebPage.class);
private static final long EXPLICIT_WAIT_TIME_OUT = WebUXDriver
.getWebUXDriver().getExplicitWait();
protected final WebUXDriver webUXDriver = WebUXDriver.getWebUXDriver();
protected final WebDriver driver = WebUXDriver.getWebDriver(true);
private String title = null;
private String url = null;
private String location = null;
private String bodyText = null;
private String htmlSource = null;
private String htmlSavedToPath = null;
private String htmlFilePath = null;
private String imageFilePath = null;
private String rlogId = null;
private String pageId = null;
private Boolean convert = true;
public Boolean getConvert() {
return convert;
}
@SuppressWarnings("unchecked")
public T setConvert(Boolean convert) {
this.convert = convert;
return (T) this;
}
public GenericWebPage() {
}
public void assertElementNotPresent(HtmlElement element) {
Logging.logWebStep(null, "assert " + element.toHTML()
+ " is not present.", false);
assertHTML(!element.isElementPresent(), element.toString() + " found.");
}
public void assertElementPresent(HtmlElement element) {
Logging.logWebStep(null, "assert " + element.toHTML() + " is present.",
false);
assertHTML(element.isElementPresent(), element.toString()
+ " not found.");
}
protected void assertHTML(boolean condition, String message) {
if (!condition) {
capturePageSnapshot();
Assertion.assertTrue(condition, message);
}
}
public void assertTextNotPresent(String text) {
Logging.logWebStep(null,
"assert text \"" + text + "\" is not present.", false);
assertHTML(!getBodyText().contains(text), "Text= {" + text + "} found.");
}
public void assertTextPresent(String text) {
Logging.logWebStep(null, "assert text \"" + text + "\" is present.",
false);
assertHTML(getBodyText().contains(text), "Text= {" + text
+ "} not found.");
}
Read-Only
Smart Insert
1 : 1



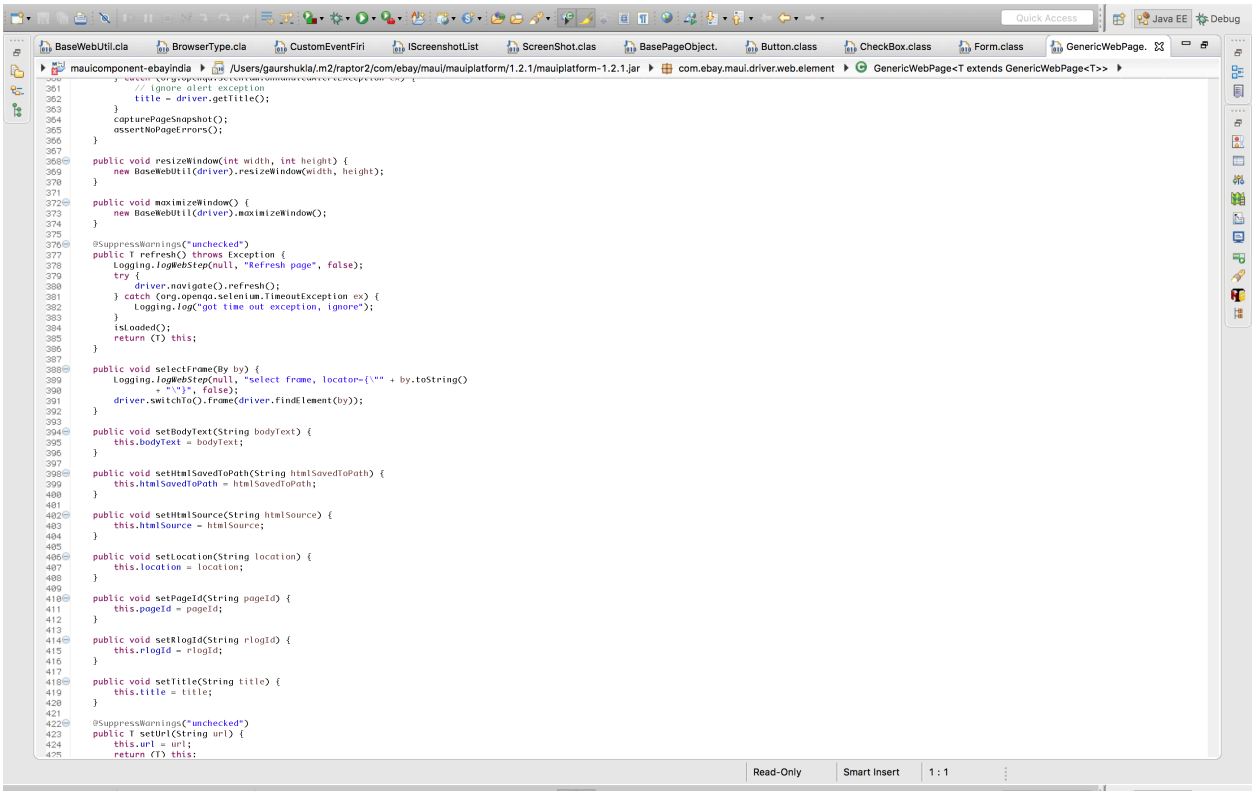
```
protected void capturePageSnapshot() {
    String suiteName = ContextManager.getGlobalContext().getTestNGContext()
            .getSuite().getName();
    String outputDirectory = ContextManager.getGlobalContext()
            .getTestNGContext().getOutputDirectory();

    ScreenShot screenShot = new ScreenshotUtil(driver)
            .capturePageSnapshot();
    this.title = screenShot.getTitle();
    this.pageId = screenShot.getPageId();
    this.rlogId = screenShot.getRlogId();
    location = screenShot.getLocation();
    htmlSource = driver.getPageSource();

    // Handle StaleElementReferenceException for IE and Opera
    try {
        bodyText = driver.findElement(By.tagName("body")).getText();
    } catch (StaleElementReferenceException ignore) {
        bodyText = htmlSource;
    }

    if (screenShot.getHtmlSourcePath() != null) {
        htmlFilePath = screenShot.getHtmlSourcePath().replace(suiteName,
                outputDirectory);
        htmlSavedToPath = screenShot.getHtmlSourcePath();
    }
    if (screenShot.getImagePath() != null) {
        imageFilePath = screenShot.getImagePath().replace(suiteName,
                outputDirectory);
    }
    Logging.logWebOutput(url,
            title + " (" + Logging.buildScreenshotLog(screenShot) + ")",
            false);
}

private String convert(String url) {
    if (ContextManager.getThreadContext() != null) {
        String urlConvertClass = ContextManager.getThreadContext()
                .getUrlConvertClass();
        if (urlConvertClass != null) {
            try {
                Class<?> converterClass = Class.forName(urlConvertClass);
                Object converter = converterClass.newInstance();
                Method convert = converterClass.getMethod("convertURL",
                        String.class);
                return (String) convert.invoke(converter, url);
            } catch (Exception e) {
                logger.warn("Convert URL failed", e);
            }

        }
    }
    return url;
}

/**
 * Ensure that the page is currently loaded.
 *
 * @return The page.
 * @throws Exception
 */
@SuppressWarnings("unchecked")
public T get() throws Exception {
    if (url != null) {
        load(getConvert());
    }
    waitForPageToLoad();

    isLoaded();
    return (T) this;
}

/**
 * Ensure that the page is currently loaded.
 *
 * @return The page.
 * @throws Exception
 */
@SuppressWarnings("unchecked")
public T get(boolean convert) throws Exception {
    if (url != null) {
        load(convert);
    }
    waitForPageToLoad();

    isLoaded();
    return (T) this;
}

public String getBodyText() {
    return bodyText;
}

public int getElementCount(HtmlElement element) {
    return element.getAllElements().size();
}

public String getHtmlSavedToPath() {
    return htmlSavedToPath;
}

public String getHtmlSource() {
    return htmlSource;
}

public String getJSErrors() {
    if (WebUXDriver.getWebUXDriver().isAddJSErrorCollectorExtension()) {
        List<JavaScriptError> jsErrorList = JavaScriptError
                .readErrors(driver);
        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;
        }
    }
    return null;
}

public String getKeywordValue(String key) {
    String site = null;
    if (ContextManager.getThreadContext().getGBHSite() != null) {
        site = ContextManager.getThreadContext().getGBHSite();
    } else {
        site = ContextManager.getThreadContext().getSite();
    }
```



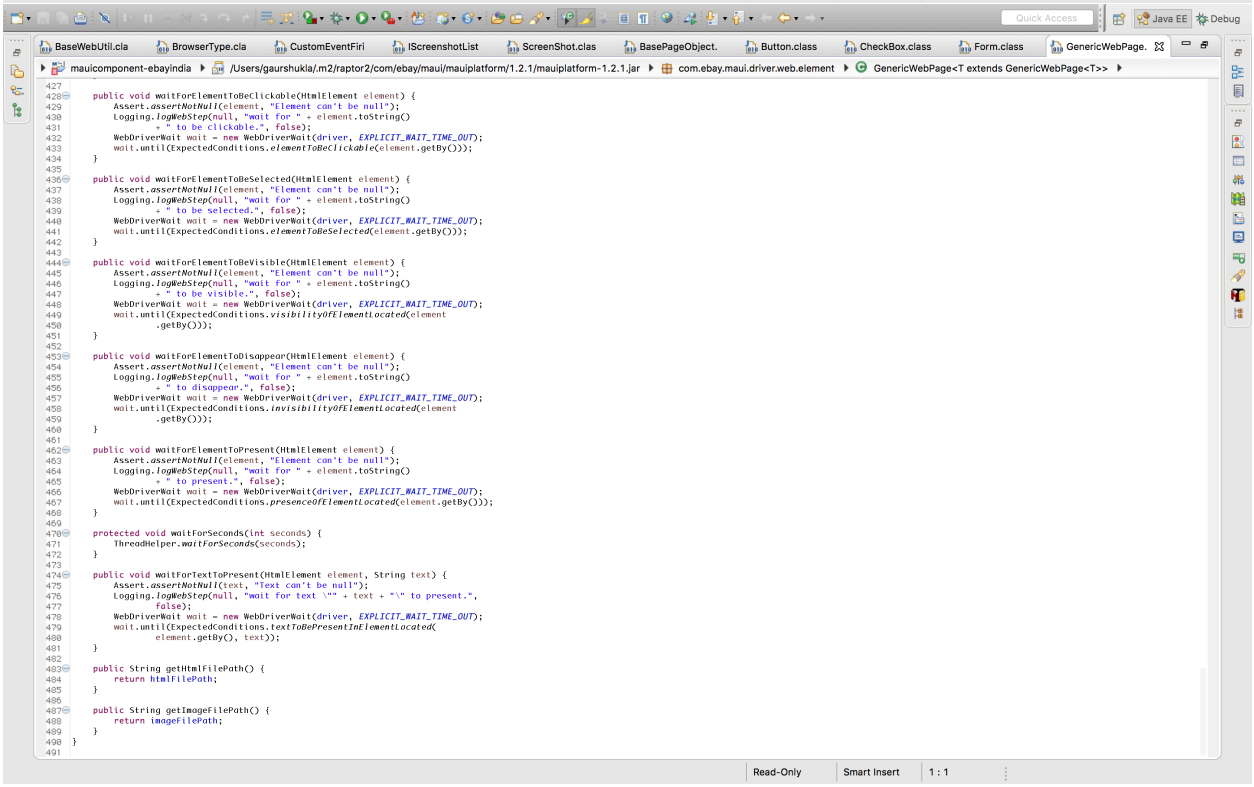
```
        site = contextManager.getThreadContext().getSite();
        }
        return Keyword.get(this.getClass(), key, site);
    }

    public String getLocation() {
        return location;
    }

    public String getPageId() {
        return pageId;
    }

    public String getRlogId() {
        return rlogId;
    }

    public String getTitle() {
        return title;
    }

    public String getUrl() {
        return url;
    }

    /**
     * Add this method in load to improve the stability of page loading
     * especially for opera
     */
    private void waitForPageToLoad() {
        try {
            driver.switchTo().defaultContent();
        } catch (UnhandledAlertException ex) {
            System.out.println("Handling alert in waitForPageToLoad");
            ThreadHelper.waitForSeconds(2);
        } catch (WebDriverException e) {
            e.printStackTrace();
        }
    }

    /**
     * Determine whether or not the page is loaded.
     *
     * @throws PageNotCurrentException
     */
    protected void isLoaded() throws PageNotCurrentException {
        if (!BrowserType.Android.getType().contains(
                ContextManager.getThreadContext().getWebRunBrowser())) {
            if (!isPageLoaded()) {
                // populateAndCapturePageSnapshot();
                try {
                    new ScreenshotUtil(driver).capturePageSnapshotOnException();
                } catch (Exception e) {
                    // Ignore all exceptions
                    e.printStackTrace();
                }
                throw new PageNotCurrentException(getClass().getCanonicalName()
                        + " is not the current page.");
            }
        }

        populateAndCapturePageSnapshot();

        AbstractPageListener.notifyPageLoad(this);
    }
```

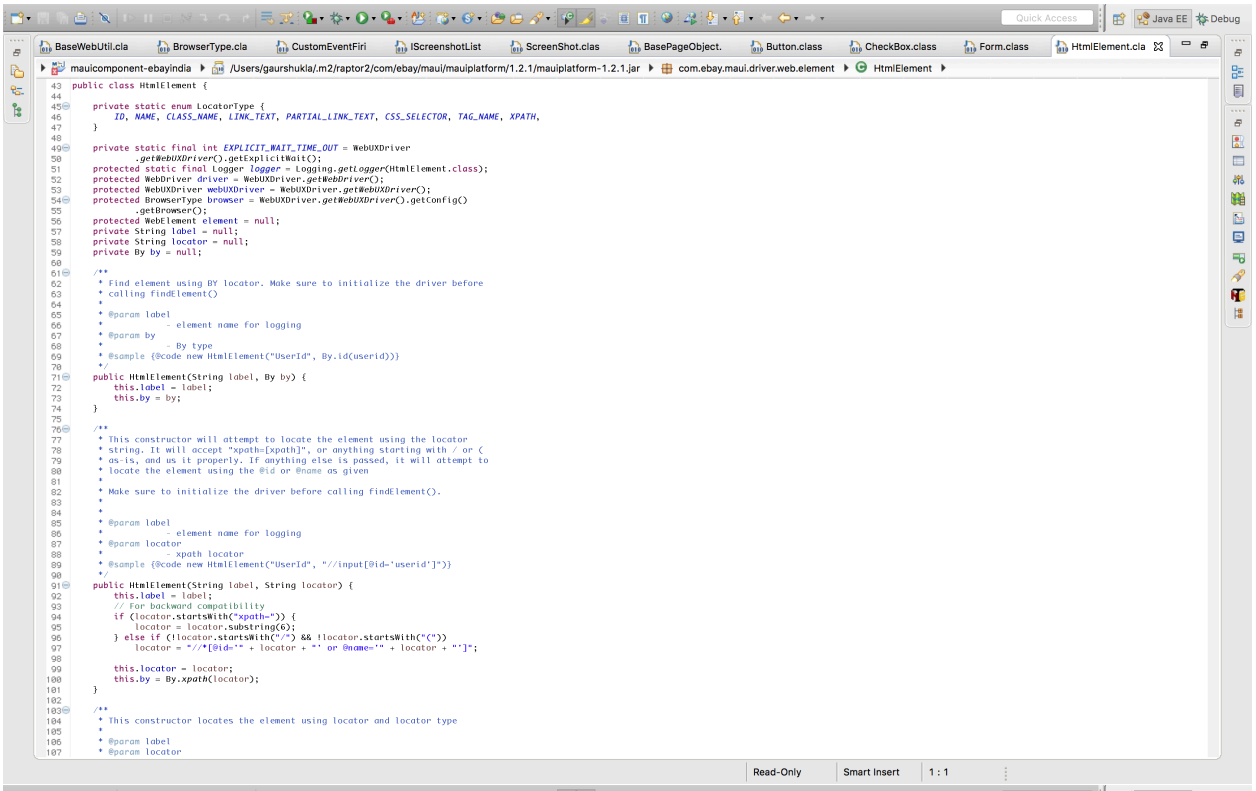
```
    /**
     * Determine whether the page is loaded or not. It return true by default.
     * Override is needed in the real page if we want to add any condition for
     * page load. The expected sample usage is:
     *
     * <pre class="code">
     * return txtUserID.isElementPresent();
     * return txtUserID.isDisplayed();
     * </pre>
     *
     */
    protected boolean isPageLoaded() {
        return true;
    }

    /**
     * Check HTTP Status Code Errors. This is going be tricky. Selenium does not
     * provide us with HTTP Response codes. So have to rely on the DOM content
     * to check for Errors. For example: QA PNR Page If page error, take error
     * snapshot and throw exception
     *
     * @throws PageNotCurrentException
     */
    public final void assertNoPageErrors() {

        for (HTTPStatusCode statusCode : HTTPStatusCode.values()) {
            if (getBodyText() != null
                    && getBodyText().toLowerCase().contains(
                            statusCode.getTitle().toLowerCase())) {
                throw new RuntimeException(statusCode.toString() + " found.");
            }
        }

    }

    /**
     * Load page URL.
     *
     */
    protected void load(boolean convert) {
        if (convert) {
            url = convert(url);
        }
        Logging.logWebStep(url, "Open url <a href='" + url + "'>" + url
                + "</a>", false);

        try {
            driver.get(url);
        } catch (org.openqa.selenium.TimeoutException ex) {
            Logging.log("Got time out when loading " + url + ", ignored");
        } catch (org.openqa.selenium.UnsupportedCommandException ex) {
            Logging.log("Got UnsupportedCommandException when loading " + url
                    + ", retry");
            driver.get(url);
        } catch (org.openqa.selenium.UnhandledAlertException ex) {
            Logging.log("got UnhandledAlertException, retry");
            driver.navigate().to(url);
        }
        url = null;
    }

    private void populateAndCapturePageSnapshot() {
        try {
            title = driver.getTitle();
        } catch (org.openqa.selenium.UnhandledAlertException ex) {
```



```
            // ignore alert exception
            title = driver.getTitle();
        }
        capturePageSnapshot();
        assertNoPageErrors();
    }

    public void resizeWindow(int width, int height) {
        new BaseWebUtil(driver).resizeWindow(width, height);
    }

    public void maximizeWindow() {
        new BaseWebUtil(driver).maximizeWindow();
    }

    @SuppressWarnings("unchecked")
    public T refresh() throws Exception {
        Logging.logWebStep(null, "Refresh page", false);
        try {
            driver.navigate().refresh();
        } catch (org.openqa.selenium.TimeoutException ex) {
            Logging.log("got time out exception, ignore");
        }
        isLoaded();
        return (T) this;
    }

    public void selectFrame(By by) {
        Logging.logWebStep(null, "select frame, locator={\"" + by.toString()
                + "\"}", false);
        driver.switchTo().frame(driver.findElement(by));
    }

    public void setBodyText(String bodyText) {
        this.bodyText = bodyText;
    }

    public void setHtmlSavedToPath(String htmlSavedToPath) {
        this.htmlSavedToPath = htmlSavedToPath;
    }

    public void setHtmlSource(String htmlSource) {
        this.htmlSource = htmlSource;
    }

    public void setLocation(String location) {
        this.location = location;
    }

    public void setPageId(String pageId) {
        this.pageId = pageId;
    }

    public void setRlogId(String rlogId) {
        this.rlogId = rlogId;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    @SuppressWarnings("unchecked")
    public T setUrl(String url) {
        this.url = url;
        return (T) this;
```

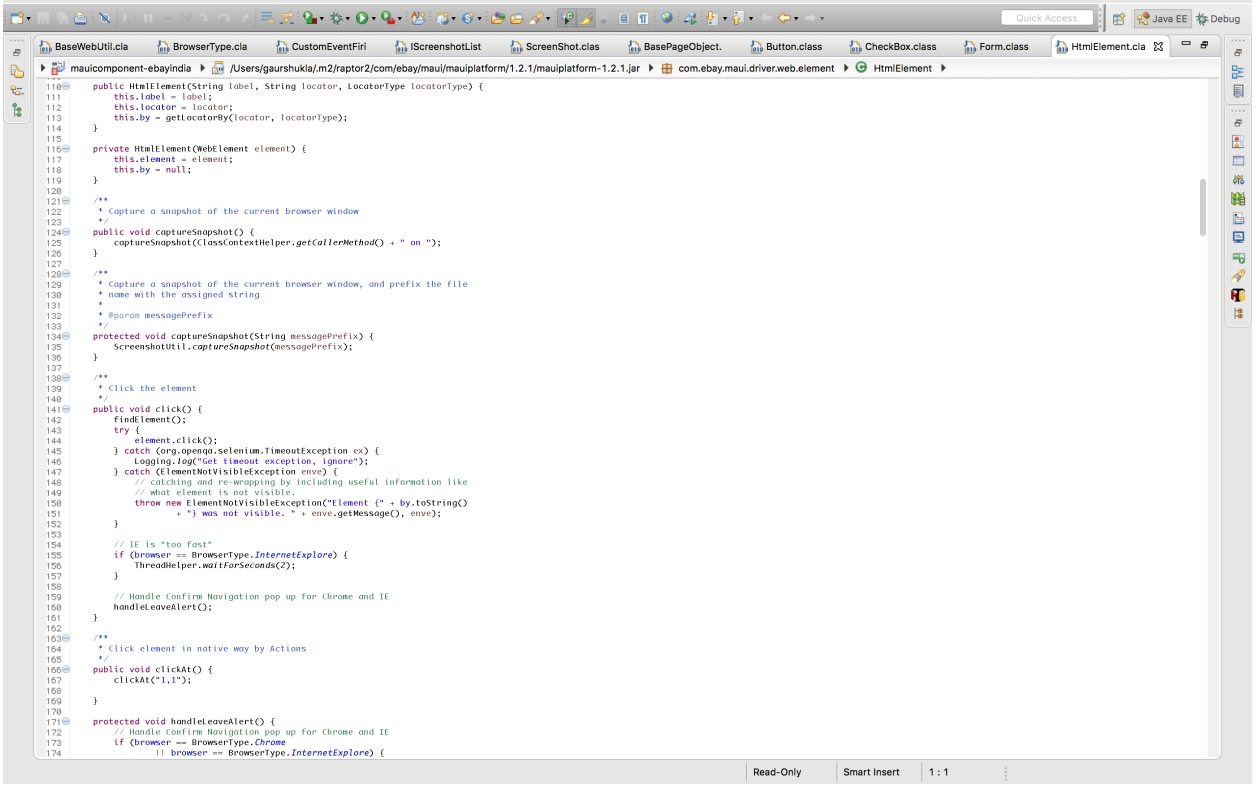


```
    public void waitForElementToBeClickable(HtmlElement element) {
        Assert.assertNotNull(element, "Element can't be null");
        Logging.logWebStep(null, "wait for " + element.toString()
                + " to be clickable.", false);
        WebDriverWait wait = new WebDriverWait(driver, EXPLICIT_WAIT_TIME_OUT);
        wait.until(ExpectedConditions.elementToBeClickable(element.getBy()));
    }

    public void waitForElementToBeSelected(HtmlElement element) {
        Assert.assertNotNull(element, "Element can't be null");
        Logging.logWebStep(null, "wait for " + element.toString()
                + " to be selected.", false);
        WebDriverWait wait = new WebDriverWait(driver, EXPLICIT_WAIT_TIME_OUT);
        wait.until(ExpectedConditions.elementToBeSelected(element.getBy()));
    }

    public void waitForElementToBeVisible(HtmlElement element) {
        Assert.assertNotNull(element, "Element can't be null");
        Logging.logWebStep(null, "wait for " + element.toString()
                + " to be visible.", false);
        WebDriverWait wait = new WebDriverWait(driver, EXPLICIT_WAIT_TIME_OUT);
        wait.until(ExpectedConditions.visibilityOfElementLocated(element
                .getBy()));
    }

    public void waitForElementToDisappear(HtmlElement element) {
        Assert.assertNotNull(element, "Element can't be null");
        Logging.logWebStep(null, "wait for " + element.toString()
                + " to disappear.", false);
        WebDriverWait wait = new WebDriverWait(driver, EXPLICIT_WAIT_TIME_OUT);
        wait.until(ExpectedConditions.invisibilityOfElementLocated(element
                .getBy()));
    }

    public void waitForElementToPresent(HtmlElement element) {
        Assert.assertNotNull(element, "Element can't be null");
        Logging.logWebStep(null, "wait for " + element.toString()
                + " to present.", false);
        WebDriverWait wait = new WebDriverWait(driver, EXPLICIT_WAIT_TIME_OUT);
        wait.until(ExpectedConditions.presenceOfElementLocated(element.getBy()));
    }

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

    public void waitForTextToPresent(HtmlElement element, String text) {
        Assert.assertNotNull(text, "Text can't be null");
        Logging.logWebStep(null, "wait for text \"" + text + "\" to present.",
                false);
        WebDriverWait wait = new WebDriverWait(driver, EXPLICIT_WAIT_TIME_OUT);
        wait.until(ExpectedConditions.textToBePresentInElementLocated(
                element.getBy(), text));
    }

    public String getHtmlFilePath() {
        return htmlFilePath;
    }

    public String getImageFilePath() {
        return imageFilePath;
    }
}
```



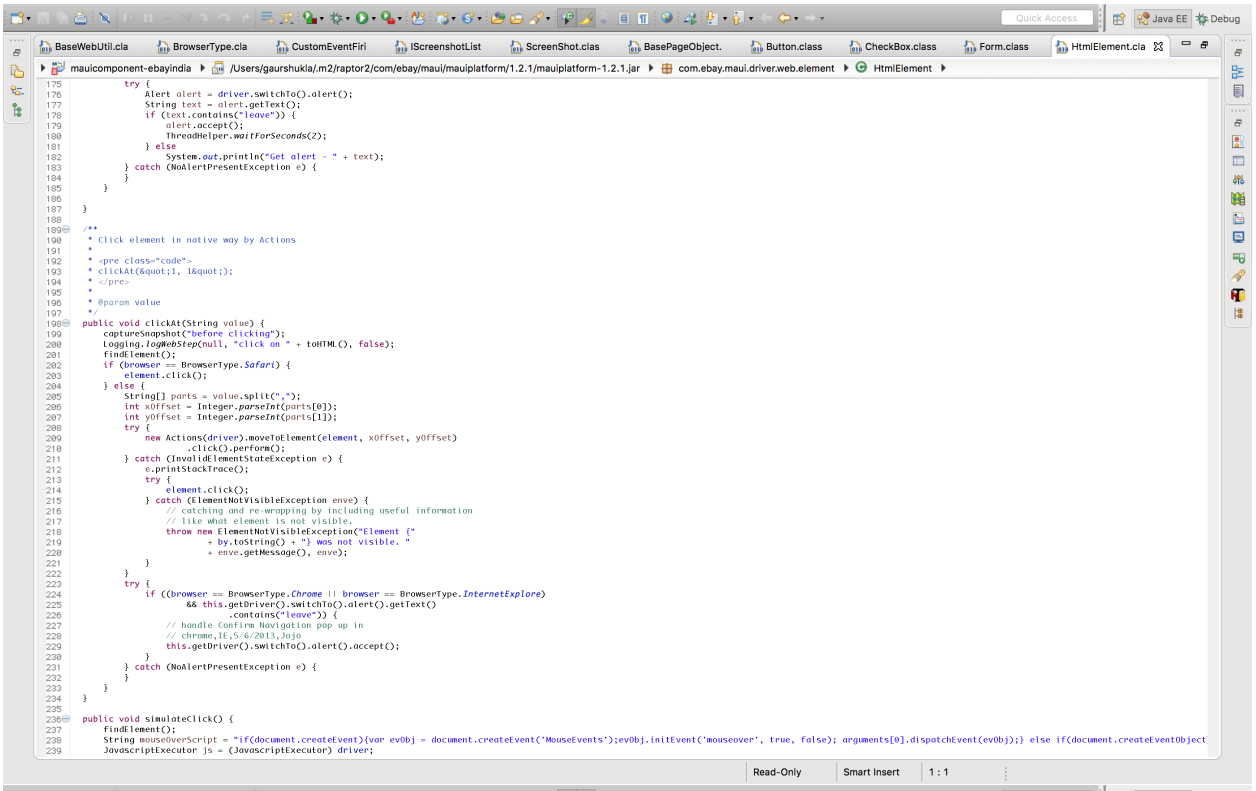
```
43  public class HtmlElement {
44
45      private static enum LocatorType {
46          ID, NAME, CLASS_NAME, LINK_TEXT, PARTIAL_LINK_TEXT, CSS_SELECTOR, TAG_NAME, XPATH,
47      }
48
49      private static final int EXPLICIT_WAIT_TIME_OUT = WebUXDriver
50              .getWebUXDriver().getExplicitWait();
51      protected static final Logger logger = Logging.getLogger(HtmlElement.class);
52      protected WebDriver driver = WebUXDriver.getWebDriver();
53      protected WebUXDriver webUXDriver = WebUXDriver.getWebUXDriver();
54      protected BrowserType browser = WebUXDriver.getWebUXDriver().getConfig()
55              .getBrowser();
56      protected WebElement element = null;
57      private String label = null;
58      private String locator = null;
59      private By by = null;
60
61      /**
62       * Find element using BY locator. Make sure to initialize the driver before
63       * calling findElement()
64       *
65       * @param label
66       *            - element name for logging
67       * @param by
68       *            - By type
69       * @sample {@code new HtmlElement("UserId", By.id(userid))}
70       */
71      public HtmlElement(String label, By by) {
72          this.label = label;
73          this.by = by;
74      }
75
76      /**
77       * This constructor will attempt to locate the element using the locator
78       * string. It will accept "xpath=[xpath]", or anything starting with / or (
79       * as-is, and us it properly. If anything else is passed, it will attempt to
80       * locate the element using the @id or @name as given
81       *
82       * Make sure to initialize the driver before calling findElement().
83       *
84       *
85       * @param label
86       *            - element name for logging
87       * @param locator
88       *            - xpath locator
89       * @sample {@code new HtmlElement("UserId", "//input[@id='userid']")}
90       */
91      public HtmlElement(String label, String locator) {
92          this.label = label;
93          // For backward compatibility
94          if (locator.startsWith("xpath=")) {
95              locator = locator.substring(6);
96          } else if (!locator.startsWith("/") && !locator.startsWith("(")
97              locator = "//*[@id='" + locator + "' or @name='" + locator + "']";
98
99          this.locator = locator;
100         this.by = By.xpath(locator);
101     }
102
103     /**
104      * This constructor locates the element using locator and locator type
105      *
106      * @param label
107      * @param locator
```

Read-Only | Smart Insert | 1 : 1

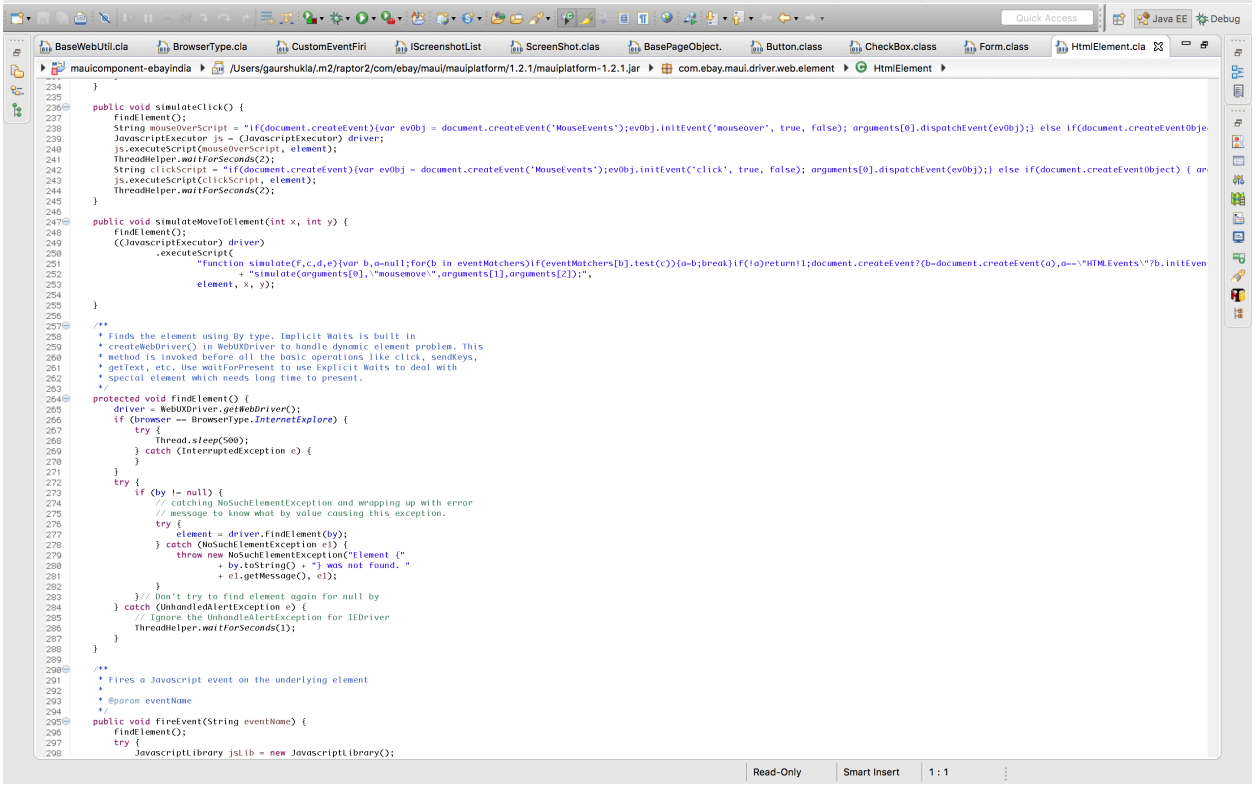


```
110     public HtmlElement(String label, String locator, LocatorType locatorType) {
111         this.label = label;
112         this.locator = locator;
113         this.by = getLocatorBy(locator, locatorType);
114     }
115
116     private HtmlElement(WebElement element) {
117         this.element = element;
118         this.by = null;
119     }
120
121     /**
122      * Capture a snapshot of the current browser window
123      */
124     public void captureSnapshot() {
125         captureSnapshot(ClassContextHelper.getCallerMethod() + " on ");
126     }
127
128     /**
129      * Capture a snapshot of the current browser window, and prefix the file
130      * name with the assigned string
131      *
132      * @param messagePrefix
133      */
134     protected void captureSnapshot(String messagePrefix) {
135         ScreenshotUtil.captureSnapshot(messagePrefix);
136     }
137
138     /**
139      * Click the element
140      */
141     public void click() {
142         findElement();
143         try {
144             element.click();
145         } catch (org.openqa.selenium.TimeoutException ex) {
146             Logging.log("Get timeout exception, ignore");
147         } catch (ElementNotVisibleException enve) {
148             // catching and re-wrapping by including useful information like
149             // what element is not visible.
150             throw new ElementNotVisibleException("Element (" + by.toString()
151                     + ") was not visible. " + enve.getMessage(), enve);
152         }
153
154         // IE is "too fast"
155         if (browser == BrowserType.InternetExplore) {
156             ThreadHelper.waitForSeconds(2);
157         }
158
159         // Handle Confirm Navigation pop up for Chrome and IE
160         handleLeaveAlert();
161     }
162
163     /**
164      * Click element in native way by Actions
165      */
166     public void clickAt() {
167         clickAt("1,1");
168
169     }
170
171     protected void handleLeaveAlert() {
172         // Handle Confirm Navigation pop up for Chrome and IE
173         if (browser == BrowserType.Chrome
174                 || browser == BrowserType.InternetExplore) {
```

Read-Only | Smart Insert | 1 : 1



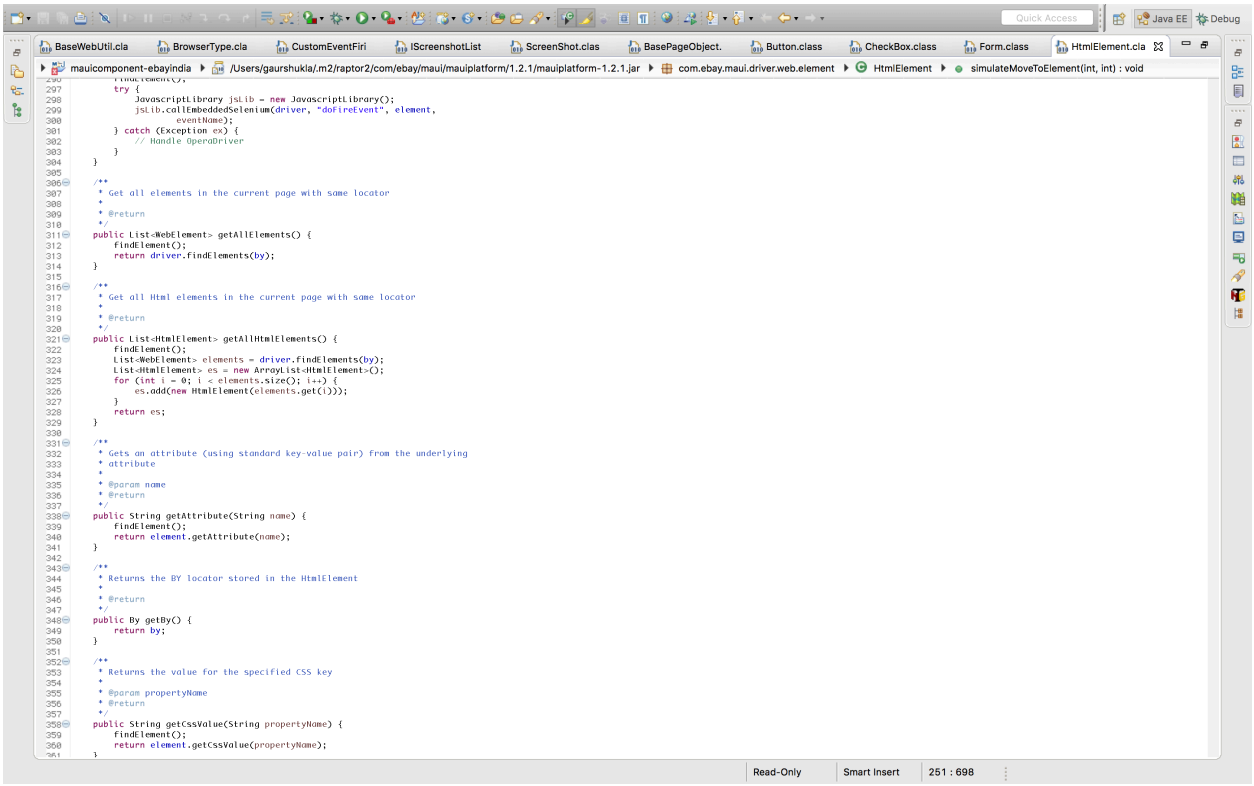
```
        try {
            Alert alert = driver.switchTo().alert();
            String text = alert.getText();
            if (text.contains("leave")) {
                alert.accept();
                ThreadHelper.waitForSeconds(2);
            } else
                System.out.println("Get alert - " + text);
        } catch (NoAlertPresentException e) {
        }
    }

}

/**
 * Click element in native way by Actions
 *
 * <pre class="code">
 * clickAt(&quot;1, 1&quot;);
 * </pre>
 *
 * @param value
 */
public void clickAt(String value) {
    captureSnapshot("before clicking");
    Logging.logWebStep(null, "click on " + toHTML(), false);
    findElement();
    if (browser == BrowserType.Safari) {
        element.click();
    } else {
        String[] parts = value.split(",");
        int xOffset = Integer.parseInt(parts[0]);
        int yOffset = Integer.parseInt(parts[1]);
        try {
            new Actions(driver).moveToElement(element, xOffset, yOffset)
                    .click().perform();
        } catch (InvalidElementStateException e) {
            e.printStackTrace();
            try {
                element.click();
            } catch (ElementNotVisibleException enve) {
                // catching and re-wrapping by including useful information
                // like what element is not visible.
                throw new ElementNotVisibleException("Element {"
                        + by.toString() + "} was not visible. "
                        + enve.getMessage(), enve);
            }
        }
        try {
            if ((browser == BrowserType.Chrome || browser == BrowserType.InternetExplore)
                    && this.getDriver().switchTo().alert().getText()
                            .contains("leave")) {
                // handle Confirm Navigation pop up in
                // chrome,IE,5/6/2013,Jojo
                this.getDriver().switchTo().alert().accept();
            }
        } catch (NoAlertPresentException e) {
        }
    }
}

public void simulateClick() {
    findElement();
    String mouseOverScript = "if(document.createEvent){var evObj = document.createEvent('MouseEvents');evObj.initEvent('mouseover', true, false); arguments[0].dispatchEvent(evObj);} else if(document.createEventObject
    JavascriptExecutor js = (JavascriptExecutor) driver;
```

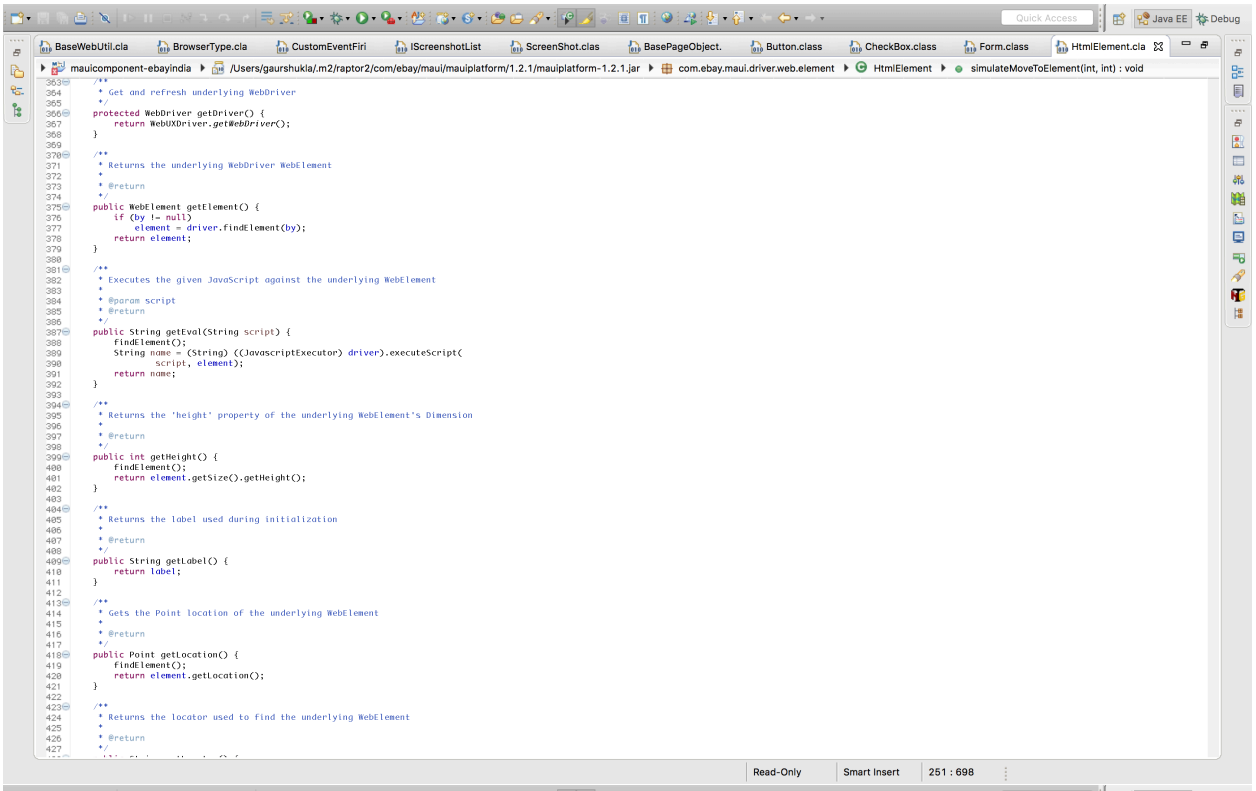


```
}

public void simulateClick() {
    findElement();
    String mouseOverScript = "if(document.createEvent){var evObj = document.createEvent('MouseEvents');evObj.initEvent('mouseover', true, false); arguments[0].dispatchEvent(evObj);} else if(document.createEventObje
    JavascriptExecutor js = (JavascriptExecutor) driver;
    js.executeScript(mouseOverScript, element);
    ThreadHelper.waitForSeconds(2);
    String clickScript = "if(document.createEvent){var evObj = document.createEvent('MouseEvents');evObj.initEvent('click', true, false); arguments[0].dispatchEvent(evObj);} else if(document.createEventObject) { ar
    js.executeScript(clickScript, element);
    ThreadHelper.waitForSeconds(2);
}

public void simulateMoveToElement(int x, int y) {
    findElement();
    ((JavascriptExecutor) driver)
            .executeScript(
                    "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.initEven
                            + "simulate(arguments[0],\"mousemove\",arguments[1],arguments[2]);",
                    element, x, y);

}

/**
 * Finds the element using By type. Implicit Waits is built in
 * createWebDriver() in WebUXDriver to handle dynamic element problem. This
 * method is invoked before all the basic operations like click, sendKeys,
 * getText, etc. Use waitForPresent to use Explicit Waits to deal with
 * special element which needs long time to present.
 */
protected void findElement() {
    driver = WebUXDriver.getWebDriver();
    if (browser == BrowserType.InternetExplore) {
        try {
            Thread.sleep(500);
        } catch (InterruptedException e) {
        }
    }
    try {
        if (by != null) {
            // catching NoSuchElementException and wrapping up with error
            // message to know what by value causing this exception.
            try {
                element = driver.findElement(by);
            } catch (NoSuchElementException e1) {
                throw new NoSuchElementException("Element {"
                        + by.toString() + "} was not found. "
                        + e1.getMessage(), e1);
            }
        }// Don't try to find element again for null by
    } catch (UnhandledAlertException e) {
        // Ignore the UnhandleAlertException for IEDriver
        ThreadHelper.waitForSeconds(1);
    }
}

/**
 * Fires a Javascript event on the underlying element
 *
 * @param eventName
 */
public void fireEvent(String eventName) {
    findElement();
    try {
        JavascriptLibrary jsLib = new JavascriptLibrary();
```

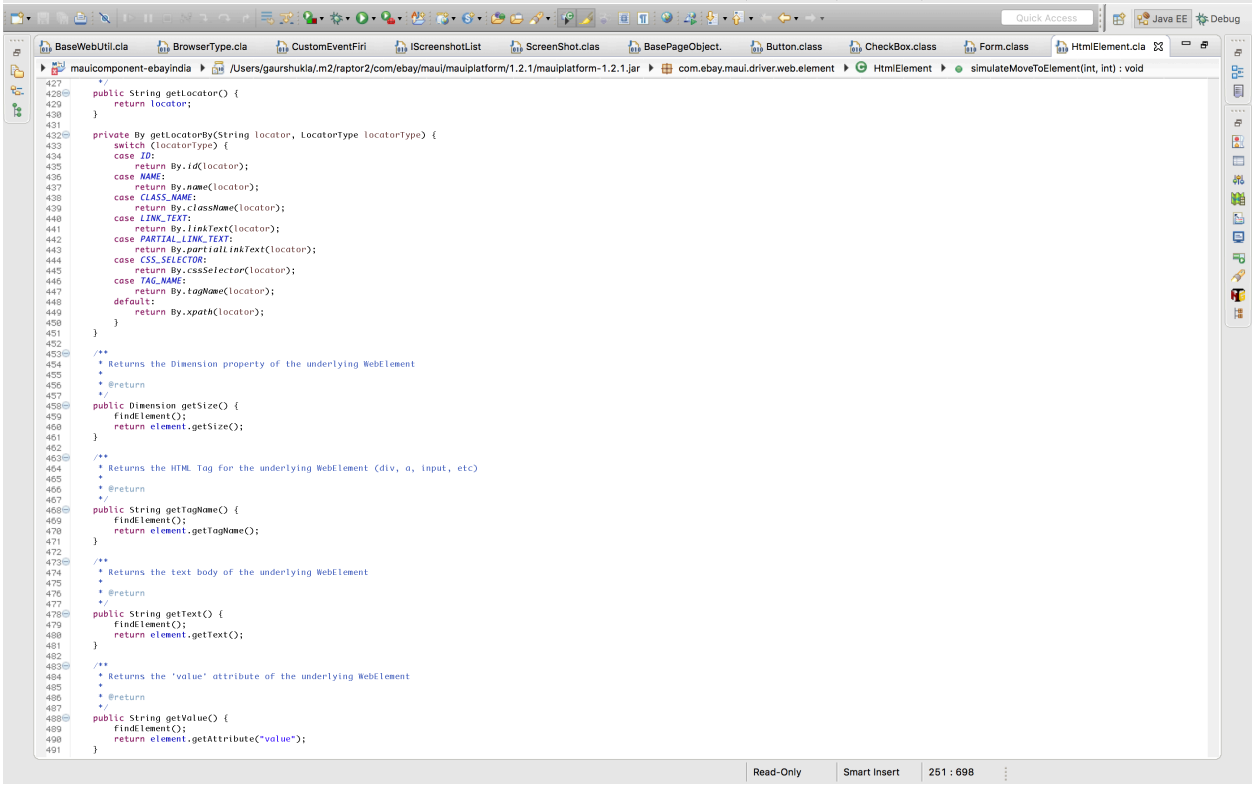
```
if(document.createEvent){var evObj = document.createEvent('MouseEvents');evObj.initEvent('mouseover', true, false);
arguments[0].dispatchEvent(evObj);} else if(document.createEventObject) { arguments[0].fireEvent('onmouseover');}"

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)):(a=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)$/,MouseEven
ts:/^(?:click|dblclick|mouse(?:down|up|over|move|out))$/}; "
```

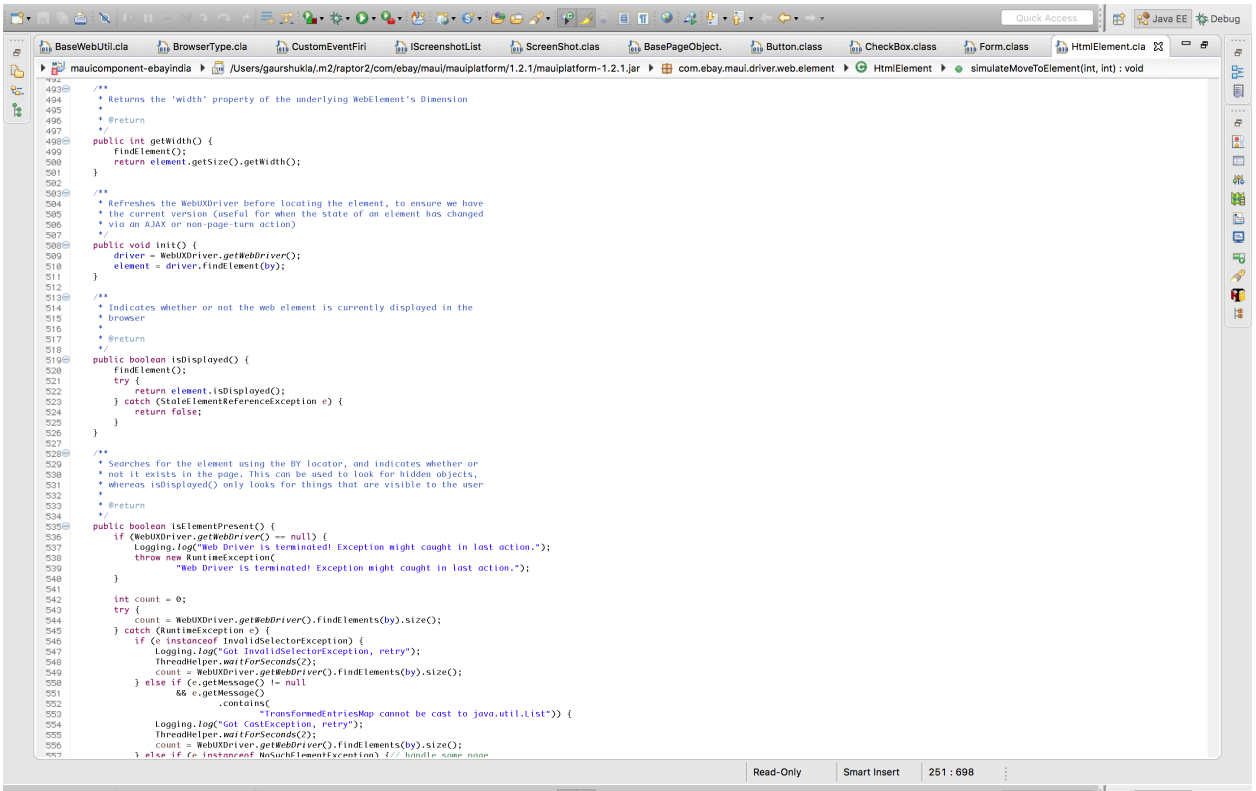




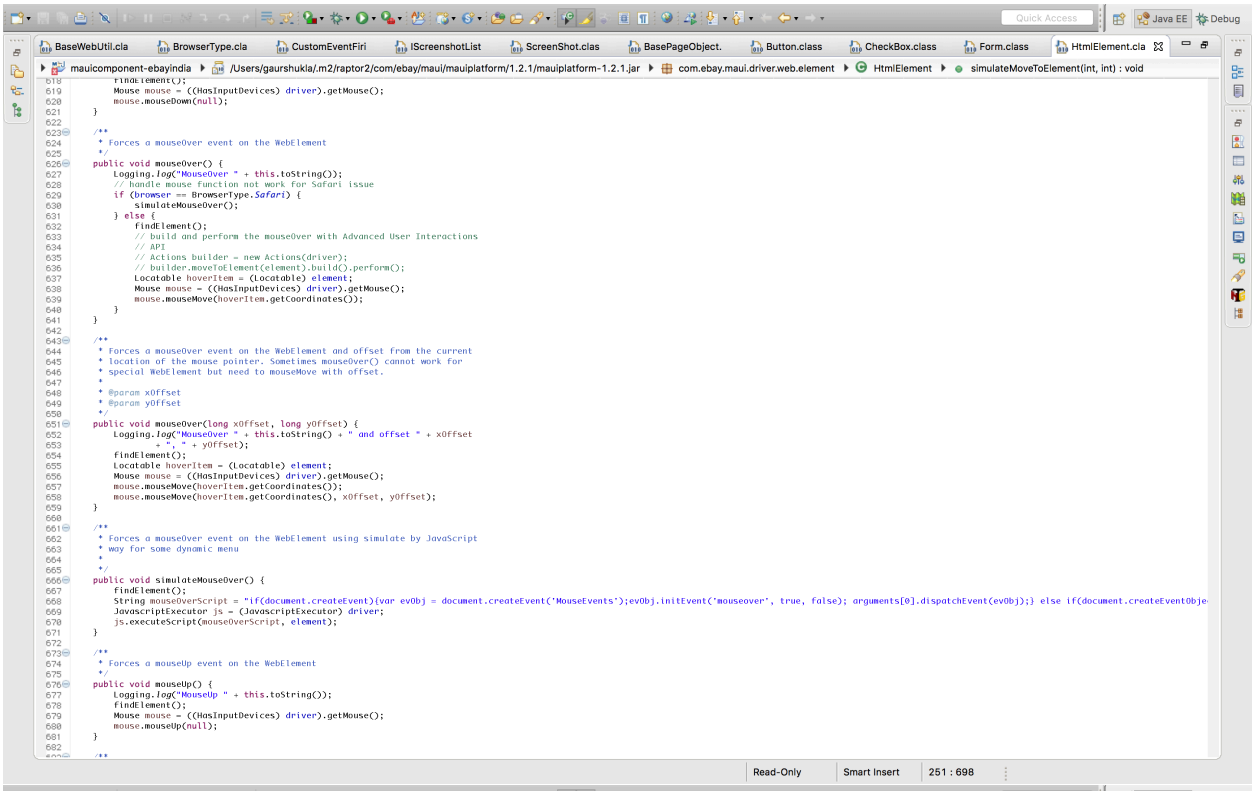

```
363     /**
364      * Get and refresh underlying WebDriver
365      */
366     protected WebDriver getDriver() {
367         return WebUXDriver.getWebDriver();
368     }
369
370     /**
371      * Returns the underlying WebDriver WebElement
372      *
373      * @return
374      */
375     public WebElement getElement() {
376         if (by != null)
377             element = driver.findElement(by);
378         return element;
379     }
380
381     /**
382      * Executes the given JavaScript against the underlying WebElement
383      *
384      * @param script
385      * @return
386      */
387     public String getEval(String script) {
388         findElement();
389         String name = (String) ((JavascriptExecutor) driver).executeScript(
390                 script, element);
391         return name;
392     }
393
394     /**
395      * Returns the 'height' property of the underlying WebElement's Dimension
396      *
397      * @return
398      */
399     public int getHeight() {
400         findElement();
401         return element.getSize().getHeight();
402     }
403
404     /**
405      * Returns the label used during initialization
406      *
407      * @return
408      */
409     public String getLabel() {
410         return label;
411     }
412
413     /**
414      * Gets the Point location of the underlying WebElement
415      *
416      * @return
417      */
418     public Point getLocation() {
419         findElement();
420         return element.getLocation();
421     }
422
423     /**
424      * Returns the locator used to find the underlying WebElement
425      *
426      * @return
427      */
```



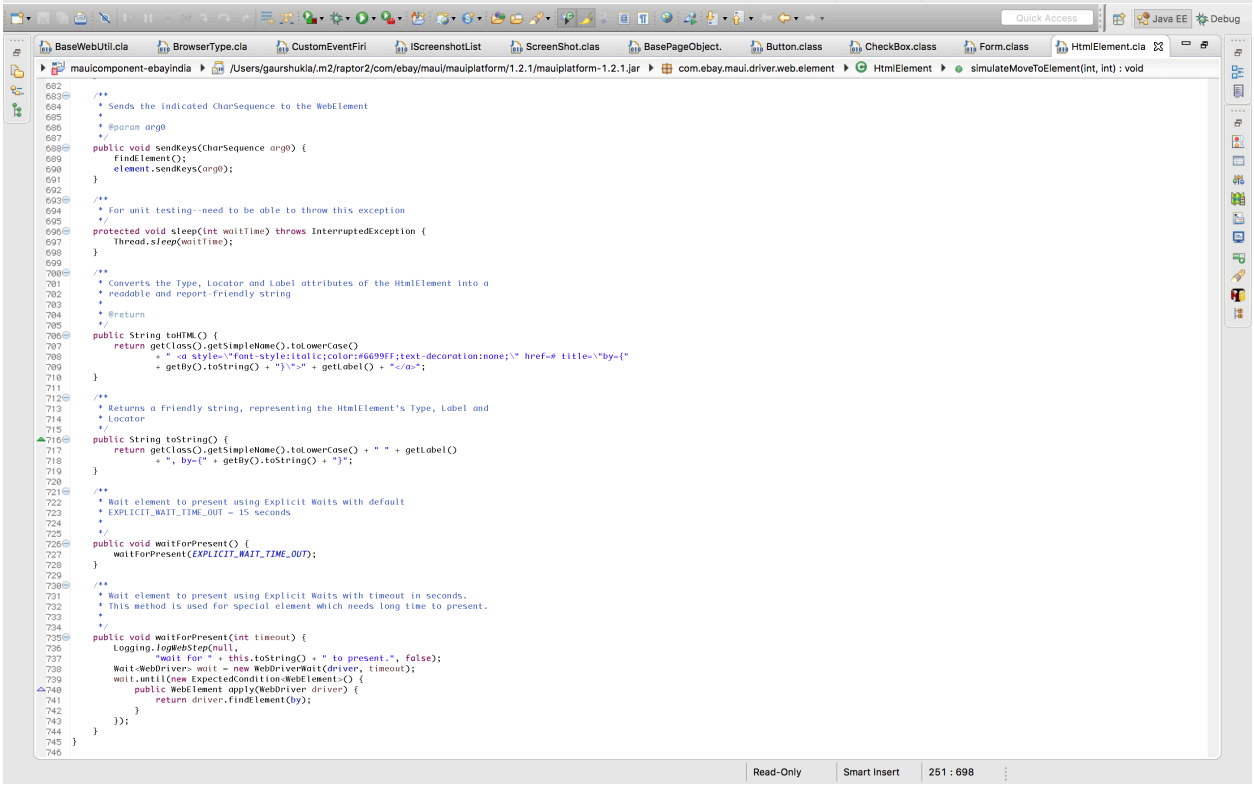

```
427      */
428     public String getLocator() {
429         return locator;
430     }
431
432     private By getLocatorBy(String locator, LocatorType locatorType) {
433         switch (locatorType) {
434         case ID:
435             return By.id(locator);
436         case NAME:
437             return By.name(locator);
438         case CLASS_NAME:
439             return By.className(locator);
440         case LINK_TEXT:
441             return By.linkText(locator);
442         case PARTIAL_LINK_TEXT:
443             return By.partialLinkText(locator);
444         case CSS_SELECTOR:
445             return By.cssSelector(locator);
446         case TAG_NAME:
447             return By.tagName(locator);
448         default:
449             return By.xpath(locator);
450         }
451     }
452
453     /**
454      * Returns the Dimension property of the underlying WebElement
455      *
456      * @return
457      */
458     public Dimension getSize() {
459         findElement();
460         return element.getSize();
461     }
462
463     /**
464      * Returns the HTML Tag for the underlying WebElement (div, a, input, etc)
465      *
466      * @return
467      */
468     public String getTagName() {
469         findElement();
470         return element.getTagName();
471     }
472
473     /**
474      * Returns the text body of the underlying WebElement
475      *
476      * @return
477      */
478     public String getText() {
479         findElement();
480         return element.getText();
481     }
482
483     /**
484      * Returns the 'value' attribute of the underlying WebElement
485      *
486      * @return
487      */
488     public String getValue() {
489         findElement();
490         return element.getAttribute("value");
491     }
```



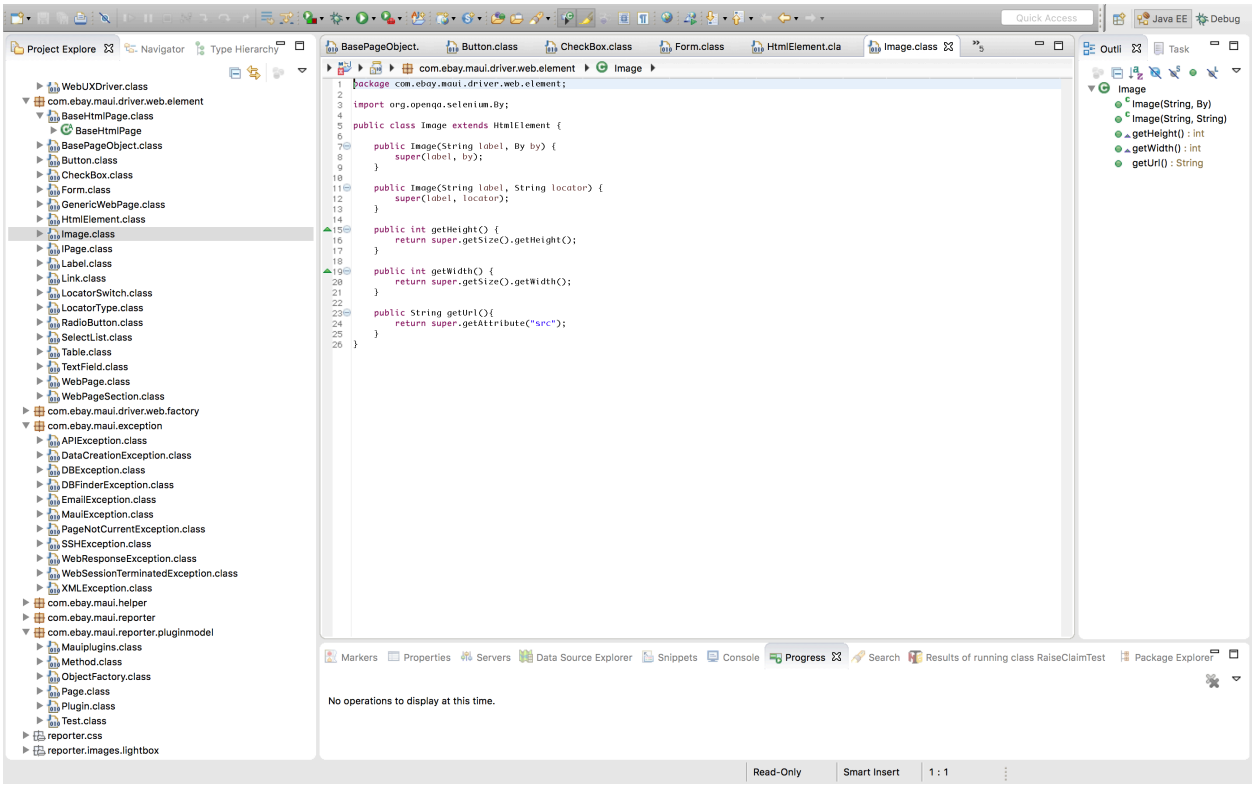
```
/**
 * Returns the 'width' property of the underlying WebElement's Dimension
 *
 * @return
 */
public int getWidth() {
    findElement();
    return element.getSize().getWidth();
}

/**
 * Refreshes the WebUXDriver before locating the element, to ensure we have
 * the current version (useful for when the state of an element has changed
 * via an AJAX or non-page-turn action)
 */
public void init() {
    driver = WebUXDriver.getWebDriver();
    element = driver.findElement(by);
}

/**
 * Indicates whether or not the web element is currently displayed in the
 * browser
 *
 * @return
 */
public boolean isDisplayed() {
    findElement();
    try {
        return element.isDisplayed();
    } catch (StaleElementReferenceException e) {
        return false;
    }
}

/**
 * Searches for the element using the BY locator, and indicates whether or
 * not it exists in the page. This can be used to look for hidden objects,
 * whereas isDisplayed() only looks for things that are visible to the user
 *
 * @return
 */
public boolean isElementPresent() {
    if (WebUXDriver.getWebDriver() == null) {
        Logging.log("Web Driver is terminated! Exception might caught in last action.");
        throw new RuntimeException(
                "Web Driver is terminated! Exception might caught in last action.");
    }

    int count = 0;
    try {
        count = WebUXDriver.getWebDriver().findElements(by).size();
    } catch (RuntimeException e) {
        if (e instanceof InvalidSelectorException) {
            Logging.log("Got InvalidSelectorException, retry");
            ThreadHelper.waitForSeconds(2);
            count = WebUXDriver.getWebDriver().findElements(by).size();
        } else if (e.getMessage() != null
                && e.getMessage()
                        .contains(
                                "TransformedEntriesMap cannot be cast to java.util.List")) {
            Logging.log("Got CastException, retry");
            ThreadHelper.waitForSeconds(2);
            count = WebUXDriver.getWebDriver().findElements(by).size();
        } else if (e instanceof NoSuchElementException) {// handle some page
                                                         // throwing this
                                                         // error for
                                                         // findElements
                                                         // in IE

            if (e.getMessage().contains(
                    "Finding elements returned an unexpected error")) {

                try {
                    if (driver.findElement(by) != null)
                        count = 1;
                } catch (NoSuchElementException ex2) {
                    // ignore
                }
            }
        } else {
            throw e;
        }
    }
    if (count == 0)
        return false;
    return true;
}

/**
 * Indicates whether or not the element is enabled in the browser
 *
 * @return
 */
public boolean isEnabled() {
    findElement();
    return element.isEnabled();
}

/**
 * Indicates whether or not the element is selected in the browser
 *
 * @return
 */
public boolean isSelected() {
    findElement();
    return element.isSelected();
}

/**
 * Whether or not the indicated text is contained in the element's getText()
 * attribute
 *
 * @param text
 * @return
 */
public boolean isTextPresent(String text) {
    findElement();
    return element.getText().contains(text);
}

/**
 * Forces a mouseDown event on the WebElement
 */
public void mouseDown() {
    Logging.log("MouseDown " + this.toString());
    findElement();
    Mouse mouse = ((HasInputDevices) driver).getMouse();
```

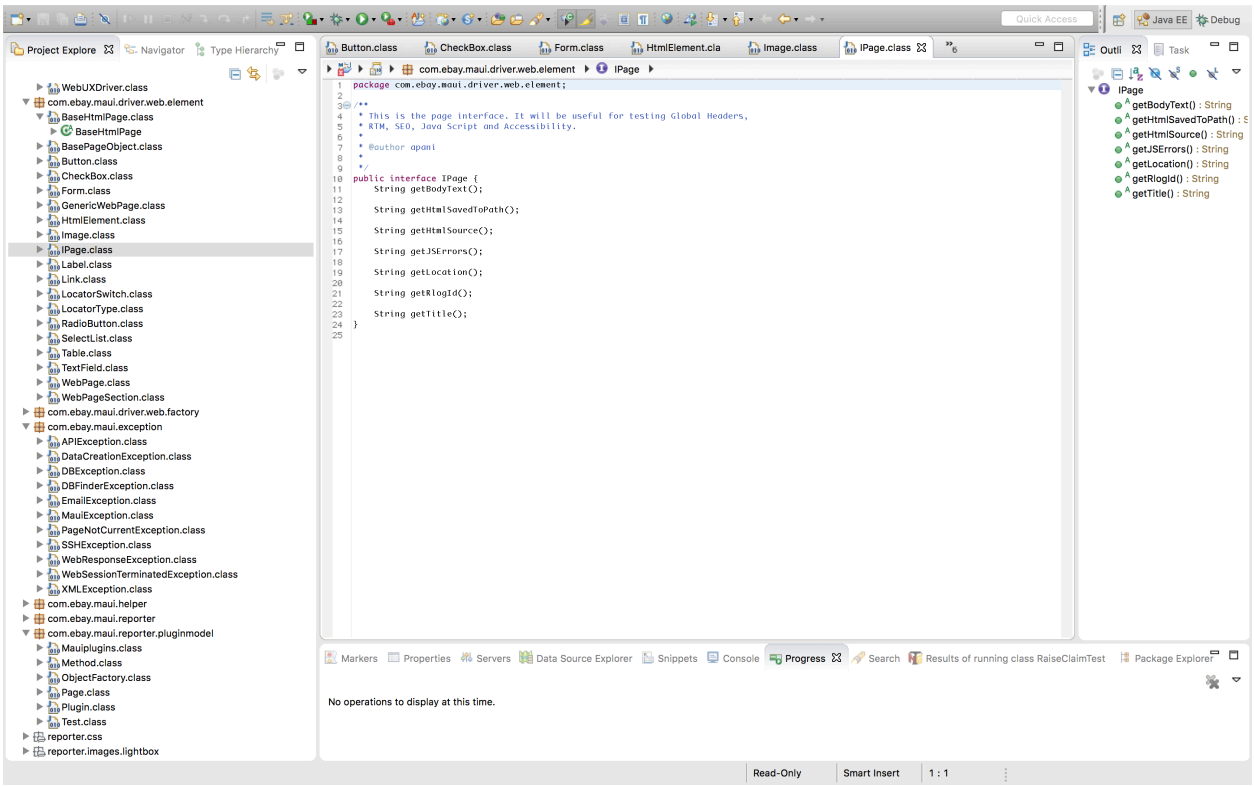


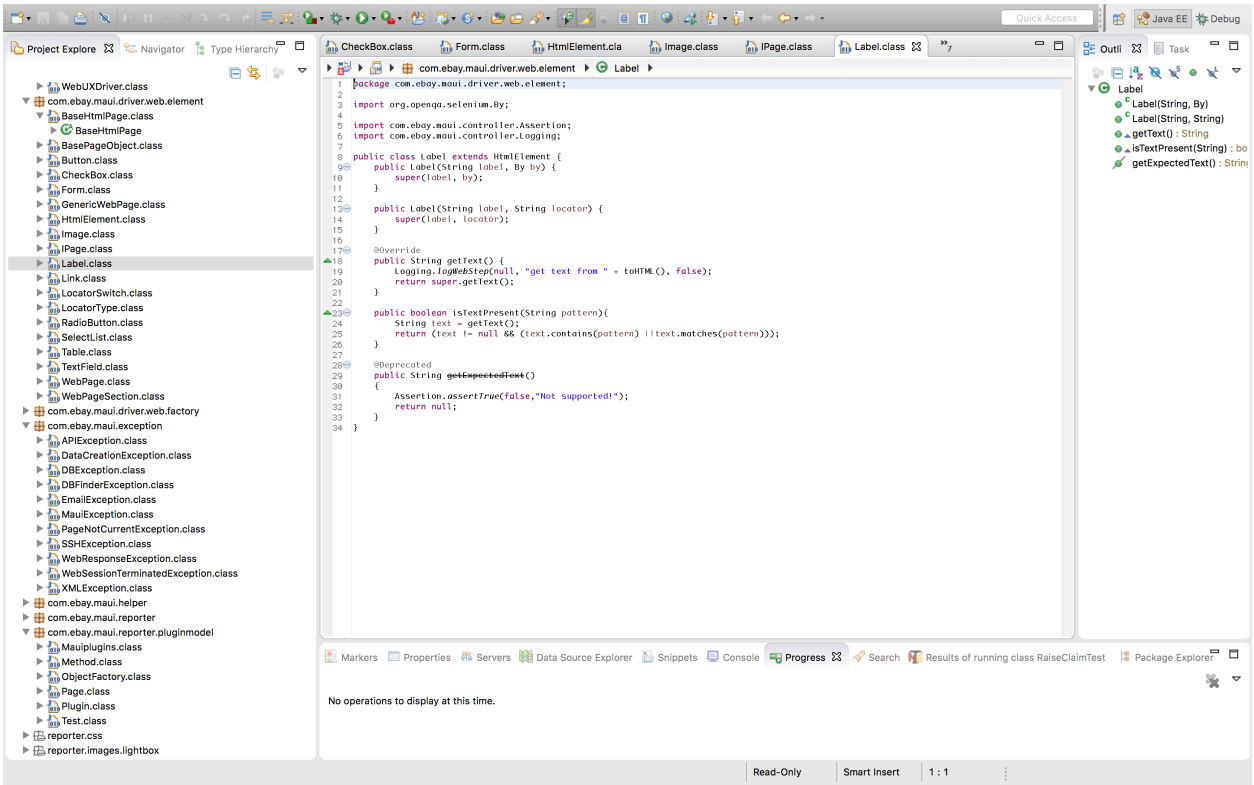
```
        findElement();
        Mouse mouse = ((HasInputDevices) driver).getMouse();
        mouse.mouseDown(null);
    }

    /**
     * Forces a mouseOver event on the WebElement
     */
    public void mouseOver() {
        Logging.log("MouseOver " + this.toString());
        // handle mouse function not work for Safari issue
        if (browser == BrowserType.Safari) {
            simulateMouseOver();
        } else {
            findElement();
            // build and perform the mouseOver with Advanced User Interactions
            // API
            // Actions builder = new Actions(driver);
            // builder.moveToElement(element).build().perform();
            Locatable hoverItem = (Locatable) element;
            Mouse mouse = ((HasInputDevices) driver).getMouse();
            mouse.mouseMove(hoverItem.getCoordinates());
        }
    }

    /**
     * Forces a mouseOver event on the WebElement and offset from the current
     * location of the mouse pointer. Sometimes mouseOver() cannot work for
     * special WebElement but need to mouseMove with offset.
     *
     * @param xOffset
     * @param yOffset
     */
    public void mouseOver(long xOffset, long yOffset) {
        Logging.log("MouseOver " + this.toString() + " and offset " + xOffset
                + ", " + yOffset);
        findElement();
        Locatable hoverItem = (Locatable) element;
        Mouse mouse = ((HasInputDevices) driver).getMouse();
        mouse.mouseMove(hoverItem.getCoordinates());
        mouse.mouseMove(hoverItem.getCoordinates(), xOffset, yOffset);
    }

    /**
     * Forces a mouseOver event on the WebElement using simulate by JavaScript
     * way for some dynamic menu
     *
     */
    public void simulateMouseOver() {
        findElement();
        String mouseOverScript = "if(document.createEvent){var evObj = document.createEvent('MouseEvents');evObj.initEvent('mouseover', true, false); arguments[0].dispatchEvent(evObj);} else if(document.createEventObje
        JavascriptExecutor js = (JavascriptExecutor) driver;
        js.executeScript(mouseOverScript, element);
    }

    /**
     * Forces a mouseUp event on the WebElement
     */
    public void mouseUp() {
        Logging.log("MouseUp " + this.toString());
        findElement();
        Mouse mouse = ((HasInputDevices) driver).getMouse();
        mouse.mouseUp(null);
    }
```

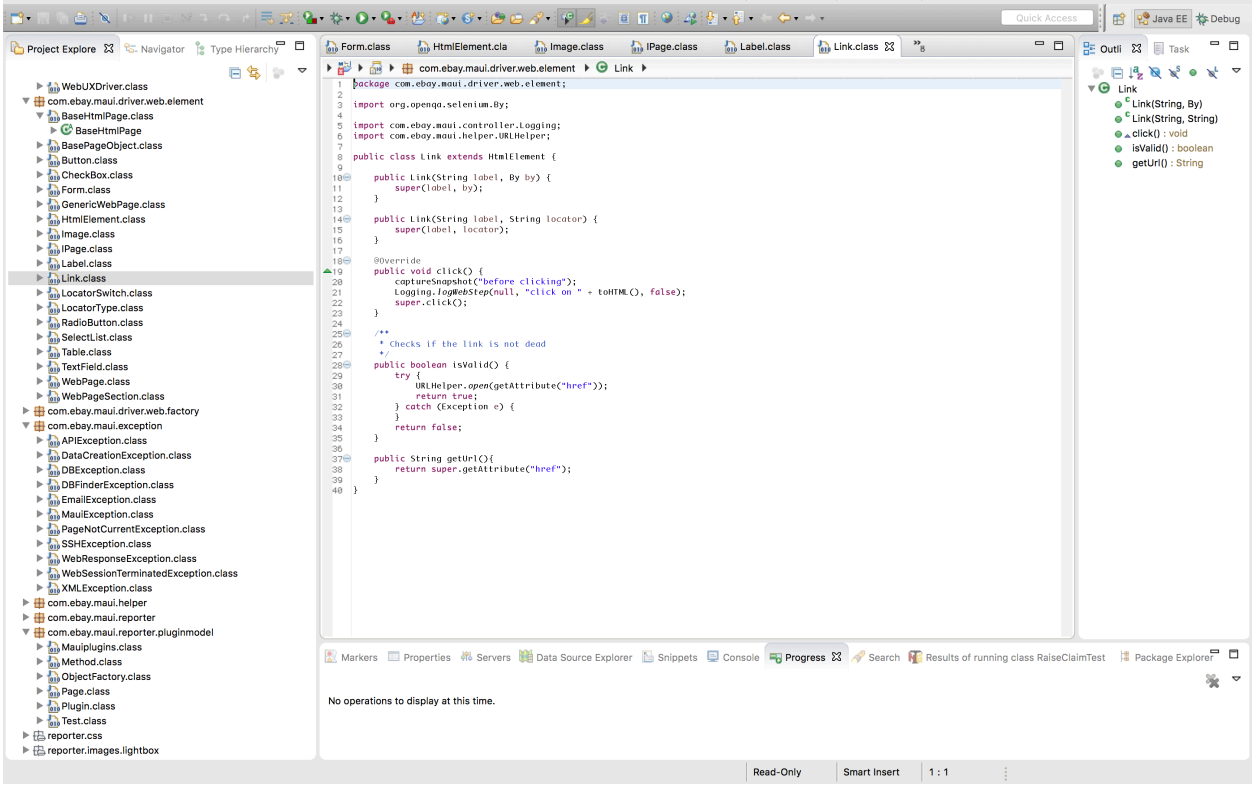


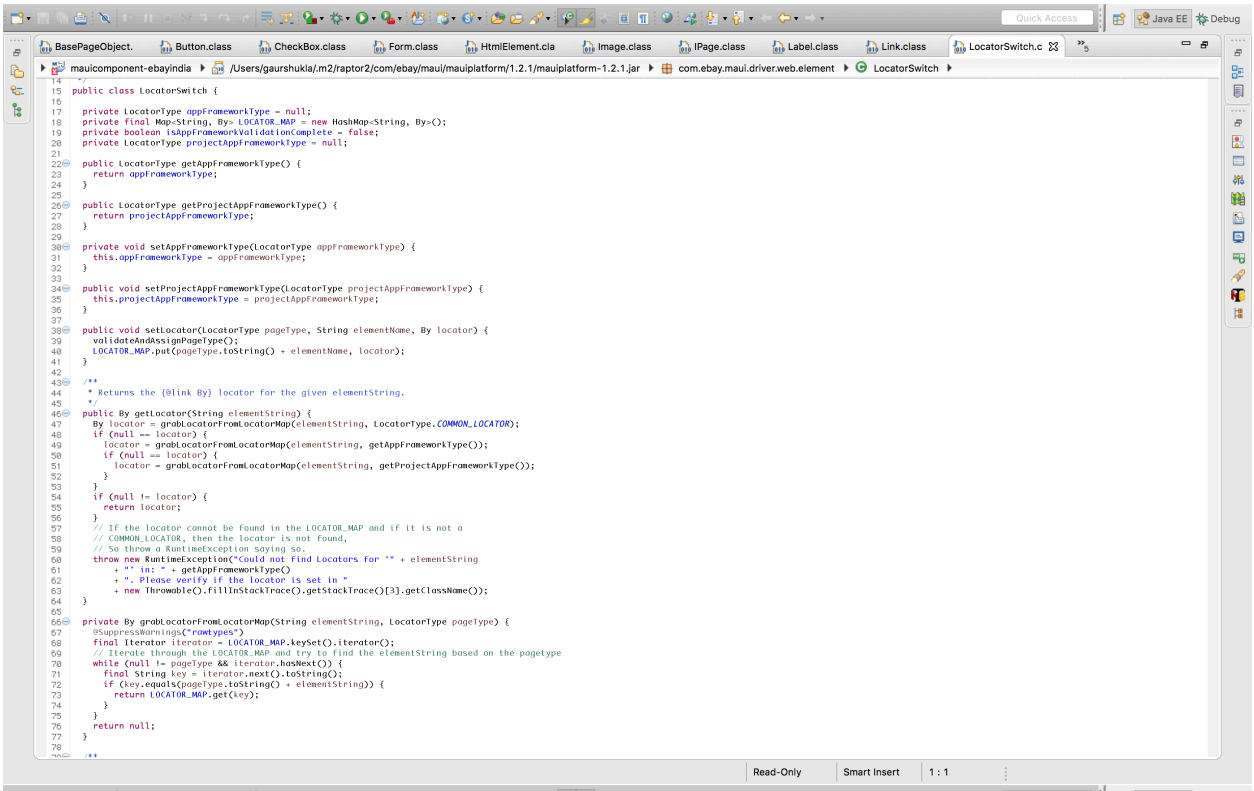
```
    /**
     * Sends the indicated CharSequence to the WebElement
     *
     * @param arg0
     */
    public void sendKeys(CharSequence arg0) {
        findElement();
        element.sendKeys(arg0);
    }

    /**
     * For unit testing--need to be able to throw this exception
     */
    protected void sleep(int waitTime) throws InterruptedException {
        Thread.sleep(waitTime);
    }

    /**
     * Converts the Type, Locator and Label attributes of the HtmlElement into a
     * readable and report-friendly string
     *
     * @return
     */
    public String toHTML() {
        return getClass().getSimpleName().toLowerCase()
                + " <a style=\"font-style:italic;color:#6699FF;text-decoration:none;\" href=# title=\"by={"
                + getBy().toString() + "}\">" + getLabel() + "</a>";
    }

    /**
     * Returns a friendly string, representing the HtmlElement's Type, Label and
     * Locator
     */
    public String toString() {
        return getClass().getSimpleName().toLowerCase() + " " + getLabel()
                + ", by={" + getBy().toString() + "}";
    }

    /**
     * Wait element to present using Explicit Waits with default
     * EXPLICIT_WAIT_TIME_OUT = 15 seconds
     *
     */
    public void waitForPresent() {
        waitForPresent(EXPLICIT_WAIT_TIME_OUT);
    }

    /**
     * Wait element to present using Explicit Waits with timeout in seconds.
     * This method is used for special element which needs long time to present.
     *
     */
    public void waitForPresent(int timeout) {
        Logging.logWebStep(null,
                "wait for " + this.toString() + " to present.", false);
        Wait<WebDriver> wait = new WebDriverWait(driver, timeout);
        wait.until(new ExpectedCondition<WebElement>() {
            public WebElement apply(WebDriver driver) {
                return driver.findElement(by);
            }
        });
    }
}
```


Project Explore
Navigator
Type Hierarchy
WebUXDriver.class
com.ebay.maui.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
com.ebay.maui.driver.web.factory
com.ebay.maui.exception
APIException.class
DataCreationException.class
DBException.class
DBFinderException.class
EmailException.class
MauiException.class
PageNotCurrentException.class
SSHException.class
WebResponseException.class
WebSessionTerminatedException.class
XMLException.class
com.ebay.maui.helper
com.ebay.maui.reporter
com.ebay.maui.reporter.pluginmodel
Mauiplugins.class
Method.class
ObjectFactory.class
Page.class
Plugin.class
Test.class
reporter.css
reporter.images.lightbox
BasePageObject.
Button.class
CheckBox.class
Form.class
HtmlElement.cla
Image.class
com.ebay.maui.driver.web.element
Image
package com.ebay.maui.driver.web.element;
import org.openqa.selenium.By;
public class Image extends HtmlElement {
public Image(String label, By by) {
super(label, by);
}
public Image(String label, String locator) {
super(label, locator);
}
public int getHeight() {
return super.getSize().getHeight();
}
public int getWidth() {
return super.getSize().getWidth();
}
public String getUrl(){
return super.getAttribute("src");
}
}
Outli
Task
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

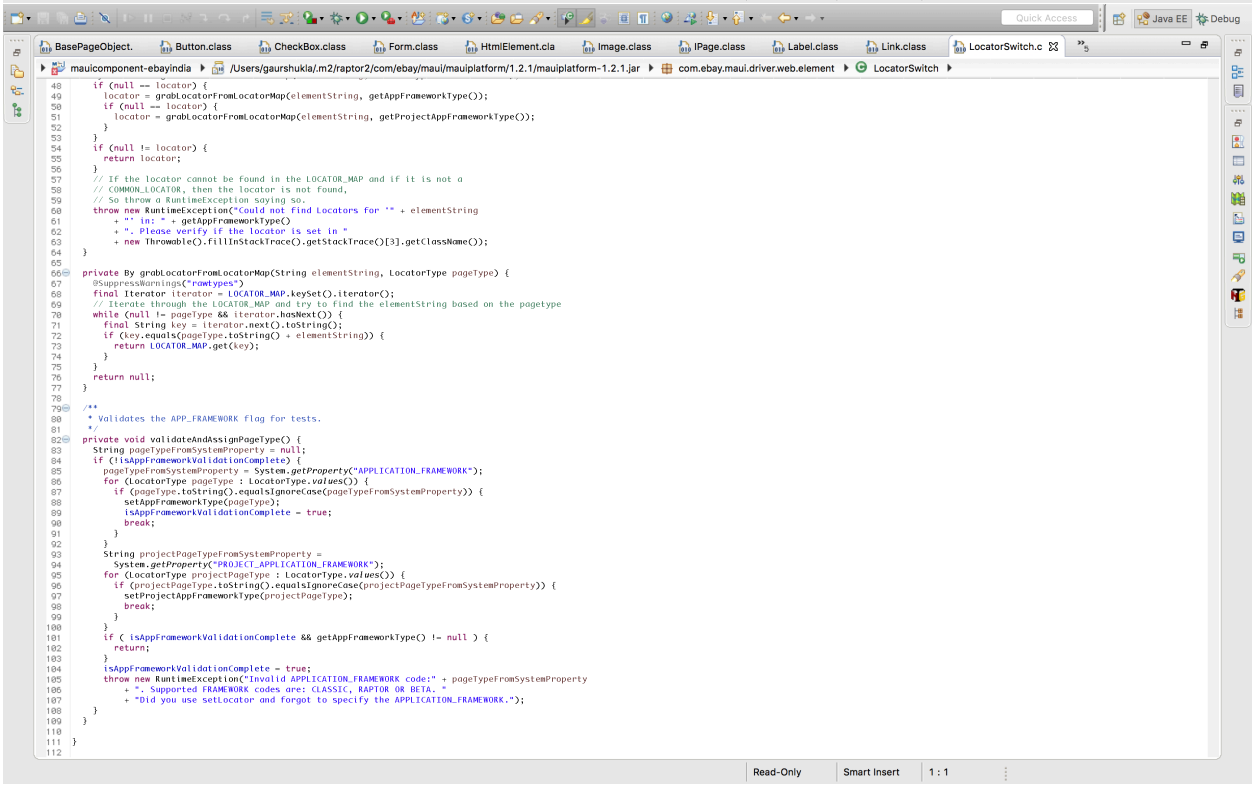

Project Explore
Navigator
Type Hierarchy
WebUXDriver.class
com.ebay.maui.driver.web.element
BaseHtmlPage.class
BaseHtmlPage
BasePageObject.class
Button.class
CheckBox.class
Form.class
GenericWebPage.class
HtmlElement.class
Image.class
IPage.class
Label.class
Link.class
LocatorSwitch.class
LocatorType.class
RadioButton.class
SelectList.class
Table.class
TextField.class
WebPage.class
WebPageSection.class
com.ebay.maui.driver.web.factory
com.ebay.maui.exception
APIException.class
DataCreationException.class
DBException.class
DBFinderException.class
EmailException.class
MauiException.class
PageNotCurrentException.class
SSHException.class
WebResponseException.class
WebSessionTerminatedException.class
XMLException.class
com.ebay.maui.helper
com.ebay.maui.reporter
com.ebay.maui.reporter.pluginmodel
Mauiplugins.class
Method.class
ObjectFactory.class
Page.class
Plugin.class
Test.class
reporter.css
reporter.images.lightbox
Button.class
CheckBox.class
Form.class
HtmlElement.cla
Image.class
IPage.class
com.ebay.maui.driver.web.element
IPage
package com.ebay.maui.driver.web.element;
/**
* This is the page interface. It will be useful for testing Global Headers,
* RTM, SEO, Java Script and Accessibility.
*
* @author apani
*
*/
public interface IPage {
String getBodyText();
String getHtmlSavedToPath();
String getHtmlSource();
String getJSErrors();
String getLocation();
String getRlogId();
String getTitle();
}
Outli
Task
IPage
getBodyText() : String
getHtmlSavedToPath() :
getHtmlSource() : String
getJSErrors() : String
getLocation() : String
getRlogId() : 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


Project Explore
Navigator
Type Hierarchy
CheckBox.class
Form.class
HtmlElement.cla
Image.class
IPage.class
Label.class
com.ebay.maui.driver.web.element
Label
Outli
Task
Label
Label(String, By)
Label(String, String)
getText() : String
isTextPresent(String) : bo
getExpectedText() : Strin
package com.ebay.maui.driver.web.element;
import org.openqa.selenium.By;
import com.ebay.maui.controller.Assertion;
import com.ebay.maui.controller.Logging;
public class Label extends HtmlElement {
public Label(String label, By by) {
super(label, by);
}
public Label(String label, String locator) {
super(label, locator);
}
@Override
public String getText() {
Logging.logWebStep(null, "get text from " + toHTML(), false);
return super.getText();
}
public boolean isTextPresent(String pattern){
String text = getText();
return (text != null && (text.contains(pattern) ||text.matches(pattern)));
}
@Deprecated
public String getExpectedText()
{
Assertion.assertTrue(false,"Not supported!");
return null;
}
}
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

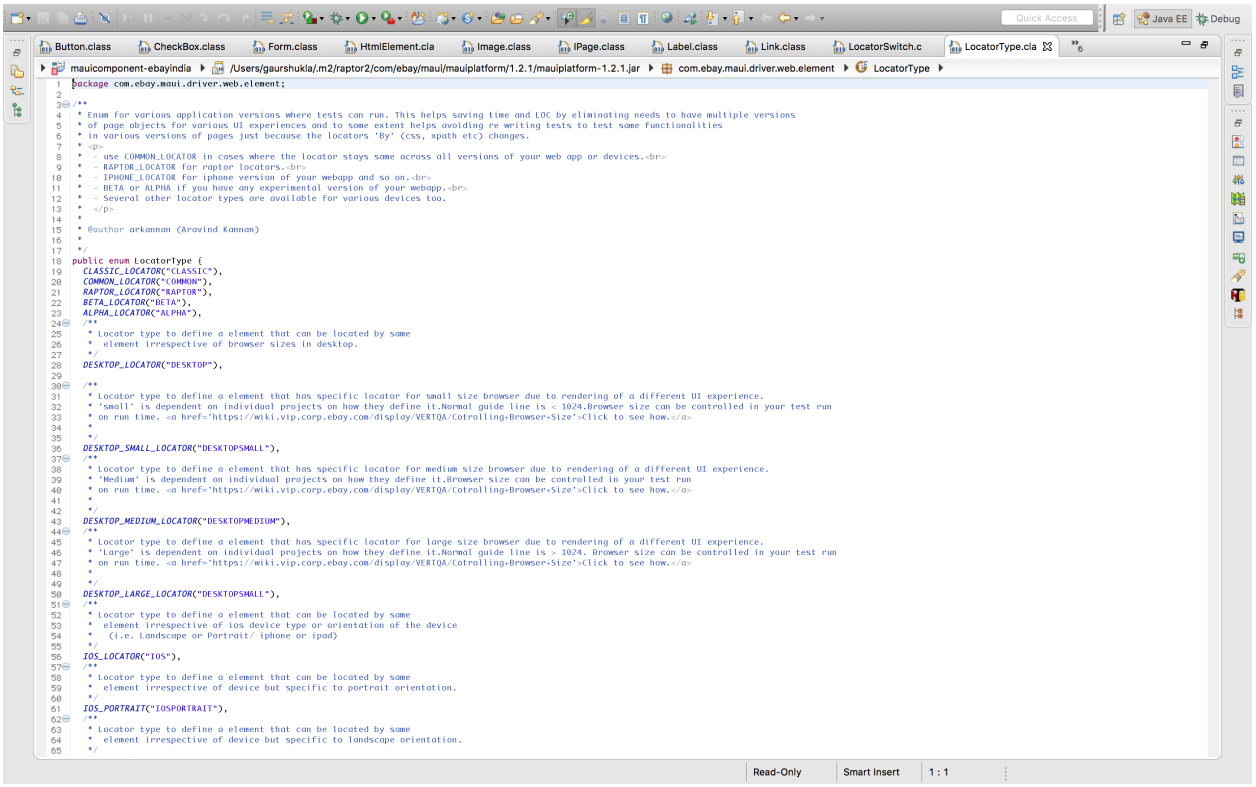

Project Explore
Navigator
Type Hierarchy
Form.class
HtmlElement.cla
Image.class
IPage.class
Label.class
Link.class
com.ebay.maui.driver.web.element
Link
Outli
Task
Link
Link(String, By)
Link(String, String)
click() : void
isValid() : boolean
getUrl() : String
package com.ebay.maui.driver.web.element;
import org.openqa.selenium.By;
import com.ebay.maui.controller.Logging;
import com.ebay.maui.helper.URLHelper;
public class Link extends HtmlElement {
public Link(String label, By by) {
super(label, by);
}
public Link(String label, String locator) {
super(label, locator);
}
@Override
public void click() {
captureSnapshot("before clicking");
Logging.logWebStep(null, "click on " + toHTML(), false);
super.click();
}
/**
* Checks if the link is not dead
*/
public boolean isValid() {
try {
URLHelper.open(getAttribute("href"));
return true;
} catch (Exception e) {
}
return false;
}
public String getUrl(){
return super.getAttribute("href");
}
}
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



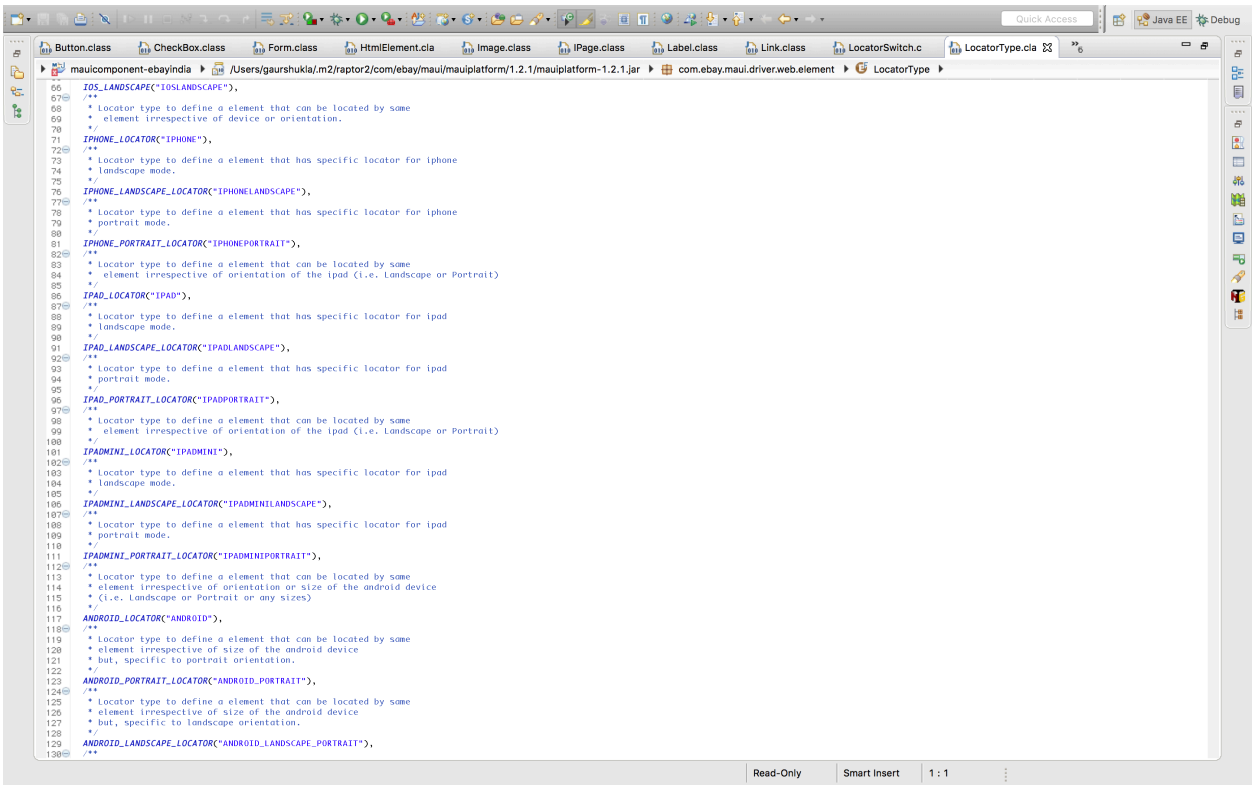
```
public class LocatorSwitch {

  private LocatorType appFrameworkType = null;
  private final Map<String, By> LOCATOR_MAP = new HashMap<String, By>();
  private boolean isAppFrameworkValidationComplete = false;
  private LocatorType projectAppFrameworkType = null;

  public LocatorType getAppFrameworkType() {
    return appFrameworkType;
  }

  public LocatorType getProjectAppFrameworkType() {
    return projectAppFrameworkType;
  }

  private void setAppFrameworkType(LocatorType appFrameworkType) {
    this.appFrameworkType = appFrameworkType;
  }

  public void setProjectAppFrameworkType(LocatorType projectAppFrameworkType) {
    this.projectAppFrameworkType = projectAppFrameworkType;
  }

  public void setLocator(LocatorType pageType, String elementName, By locator) {
    validateAndAssignPageType();
    LOCATOR_MAP.put(pageType.toString() + elementName, locator);
  }

  /**
   * Returns the {@link By} locator for the given elementString.
   */
  public By getLocator(String elementString) {
    By locator = grabLocatorFromLocatorMap(elementString, LocatorType.COMMON_LOCATOR);
    if (null == locator) {
      locator = grabLocatorFromLocatorMap(elementString, getAppFrameworkType());
      if (null == locator) {
        locator = grabLocatorFromLocatorMap(elementString, getProjectAppFrameworkType());
      }
    }
    if (null != locator) {
      return locator;
    }
    // If the locator cannot be found in the LOCATOR_MAP and if it is not a
    // COMMON_LOCATOR, then the locator is not found,
    // So throw a RuntimeException saying so.
    throw new RuntimeException("Could not find Locators for '" + elementString
        + "' in: " + getAppFrameworkType()
        + ". Please verify if the locator is set in "
        + new Throwable().fillInStackTrace().getStackTrace()[3].getClassName());
  }

  private By grabLocatorFromLocatorMap(String elementString, LocatorType pageType) {
    @SuppressWarnings("rawtypes")
    final Iterator iterator = LOCATOR_MAP.keySet().iterator();
    // Iterate through the LOCATOR_MAP and try to find the elementString based on the pagetype
    while (null != pageType && iterator.hasNext()) {
      final String key = iterator.next().toString();
      if (key.equals(pageType.toString() + elementString)) {
        return LOCATOR_MAP.get(key);
      }
    }
    return null;
  }
```



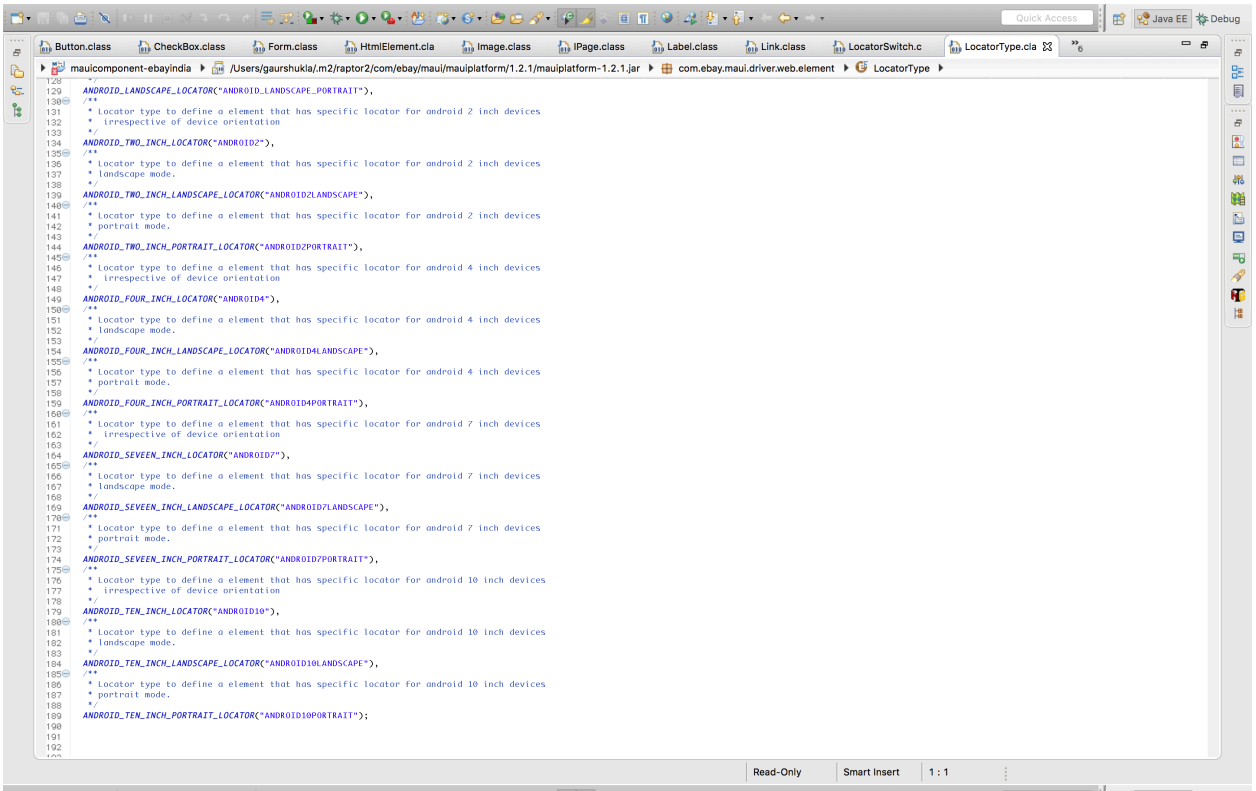
```
    if (null == locator) {
      locator = grabLocatorFromLocatorMap(elementString, getAppFrameworkType());
      if (null == locator) {
        locator = grabLocatorFromLocatorMap(elementString, getProjectAppFrameworkType());
      }
    }
    if (null != locator) {
      return locator;
    }
    // If the locator cannot be found in the LOCATOR_MAP and if it is not a
    // COMMON_LOCATOR, then the locator is not found,
    // So throw a RuntimeException saying so.
    throw new RuntimeException("Could not find Locators for '" + elementString
        + "' in: " + getAppFrameworkType()
        + ". Please verify if the locator is set in "
        + new Throwable().fillInStackTrace().getStackTrace()[3].getClassName());
  }

  private By grabLocatorFromLocatorMap(String elementString, LocatorType pageType) {
    @SuppressWarnings("rawtypes")
    final Iterator iterator = LOCATOR_MAP.keySet().iterator();
    // Iterate through the LOCATOR_MAP and try to find the elementString based on the pagetype
    while (null != pageType && iterator.hasNext()) {
      final String key = iterator.next().toString();
      if (key.equals(pageType.toString() + elementString)) {
        return LOCATOR_MAP.get(key);
      }
    }
    return null;
  }

  /**
   * Validates the APP_FRAMEWORK flag for tests.
   */
  private void validateAndAssignPageType() {
    String pageTypeFromSystemProperty = null;
    if (!isAppFrameworkValidationComplete) {
      pageTypeFromSystemProperty = System.getProperty("APPLICATION_FRAMEWORK");
      for (LocatorType pageType : LocatorType.values()) {
        if (pageType.toString().equalsIgnoreCase(pageTypeFromSystemProperty)) {
          setAppFrameworkType(pageType);
          isAppFrameworkValidationComplete = true;
          break;
        }
      }
      String projectPageTypeFromSystemProperty =
        System.getProperty("PROJECT_APPLICATION_FRAMEWORK");
      for (LocatorType projectPageType : LocatorType.values()) {
        if (projectPageType.toString().equalsIgnoreCase(projectPageTypeFromSystemProperty)) {
          setProjectAppFrameworkType(projectPageType);
          break;
        }
      }
      if ( isAppFrameworkValidationComplete && getAppFrameworkType() != null ) {
        return;
      }
      isAppFrameworkValidationComplete = true;
      throw new RuntimeException("Invalid APPLICATION_FRAMEWORK code:" + pageTypeFromSystemProperty
          + ". Supported FRAMEWORK codes are: CLASSIC, RAPTOR OR BETA. "
          + "Did you use setLocator and forgot to specify the APPLICATION_FRAMEWORK.");
    }
  }
}
```

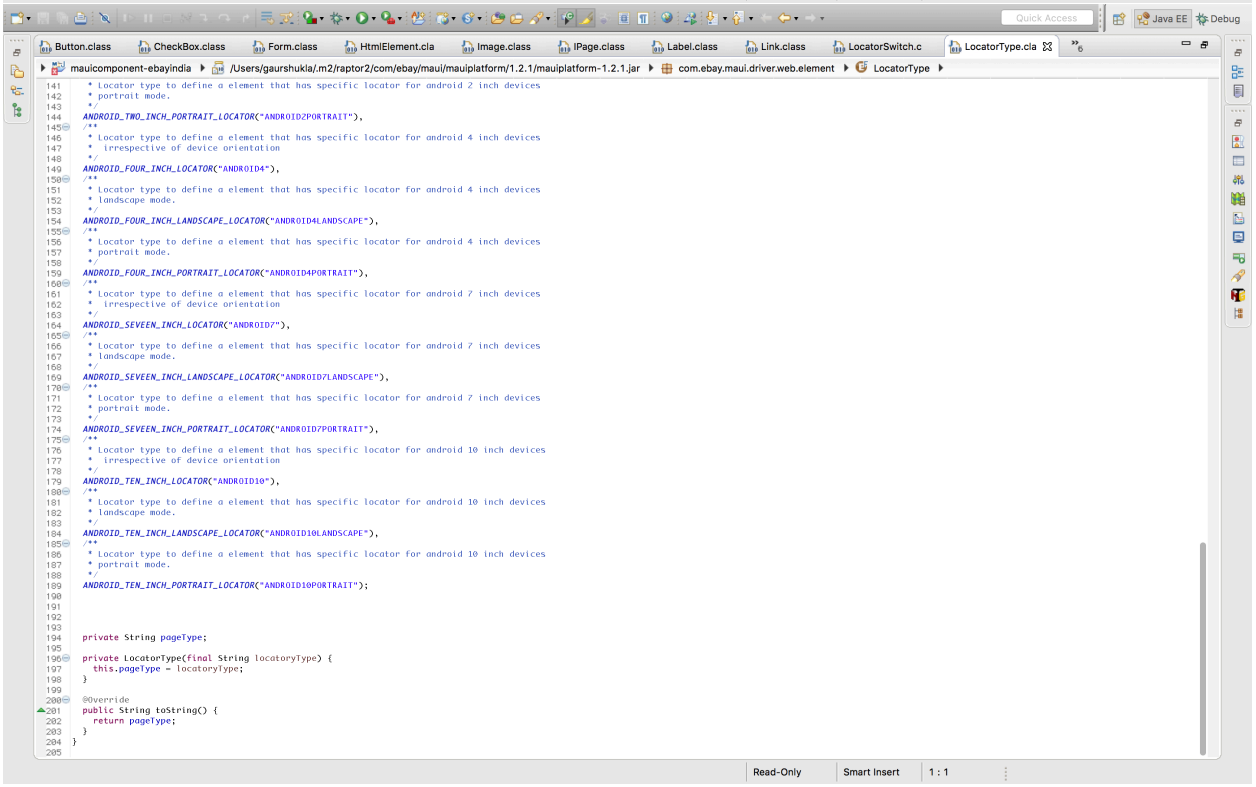


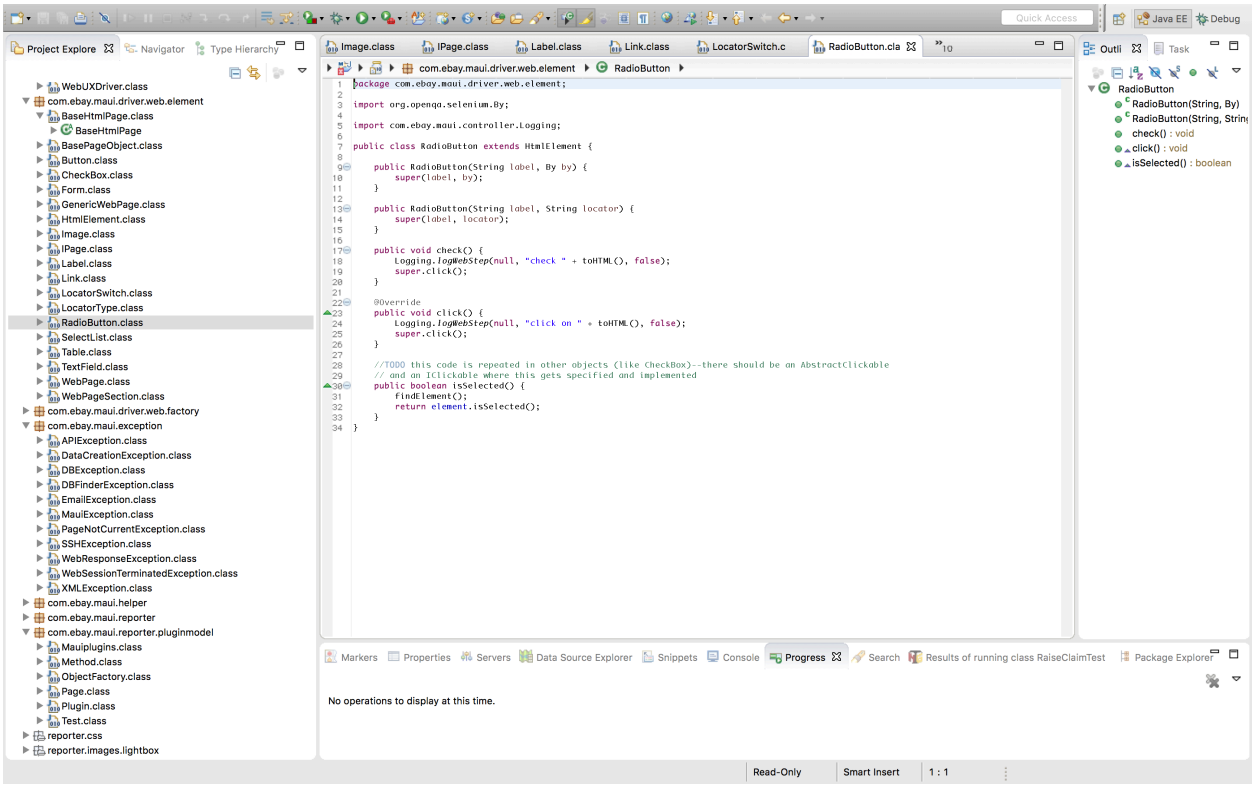
```
package com.ebay.maui.driver.web.element;

/**
 * Enum for various application versions where tests can run. This helps saving time and LOC by eliminating needs to have multiple versions
 * of page objects for various UI experiences and to some extent helps avoiding re writing tests to test same functionalities
 * in various versions of pages just because the locators 'By' (css, xpath etc) changes.
 * <p>
 *  - use COMMON_LOCATOR in cases where the locator stays same across all versions of your web app or devices.<br>
 *  - RAPTOR_LOCATOR for raptor locators.<br>
 *  - IPHONE_LOCATOR for iphone version of your webapp and so on.<br>
 *  - BETA or ALPHA if you have any experimental version of your webapp.<br>
 *  - Several other locator types are available for various devices too.
 *  </p>
 *
 * @author arkannan (Aravind Kannan)
 *
 */
public enum LocatorType {
  CLASSIC_LOCATOR("CLASSIC"),
  COMMON_LOCATOR("COMMON"),
  RAPTOR_LOCATOR("RAPTOR"),
  BETA_LOCATOR("BETA"),
  ALPHA_LOCATOR("ALPHA"),
  /**
   * Locator type to define a element that can be located by same
   *  element irrespective of browser sizes in desktop.
   */
  DESKTOP_LOCATOR("DESKTOP"),

  /**
   * Locator type to define a element that has specific locator for small size browser due to rendering of a different UI experience.
   * '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
   * on run time. <a href='https://wiki.vip.corp.ebay.com/display/VERTQA/Cotrolling+Browser+Size'>Click to see how.</a>
   *
   */
  DESKTOP_SMALL_LOCATOR("DESKTOPSMALL"),
  /**
   * Locator type to define a element that has specific locator for medium size browser due to rendering of a different UI experience.
   * 'Medium' is dependent on individual projects on how they define it.Browser size can be controlled in your test run
   * on run time. <a href='https://wiki.vip.corp.ebay.com/display/VERTQA/Cotrolling+Browser+Size'>Click to see how.</a>
   *
   */
  DESKTOP_MEDIUM_LOCATOR("DESKTOPMEDIUM"),
  /**
   * Locator type to define a element that has specific locator for large size browser due to rendering of a different UI experience.
   * '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
   * on run time. <a href='https://wiki.vip.corp.ebay.com/display/VERTQA/Cotrolling+Browser+Size'>Click to see how.</a>
   *
   */
  DESKTOP_LARGE_LOCATOR("DESKTOPSMALL"),
  /**
   * Locator type to define a element that can be located by same
   *  element irrespective of ios device type or orientation of the device
   *    (i.e. Landscape or Portrait/ iphone or ipad)
   */
  IOS_LOCATOR("IOS"),
  /**
   * Locator type to define a element that can be located by same
   *  element irrespective of device but specific to portrait orientation.
   */
  IOS_PORTRAIT("IOSPORTRAIT"),
  /**
   * Locator type to define a element that can be located by same
   *  element irrespective of device but specific to landscape orientation.
   */
```



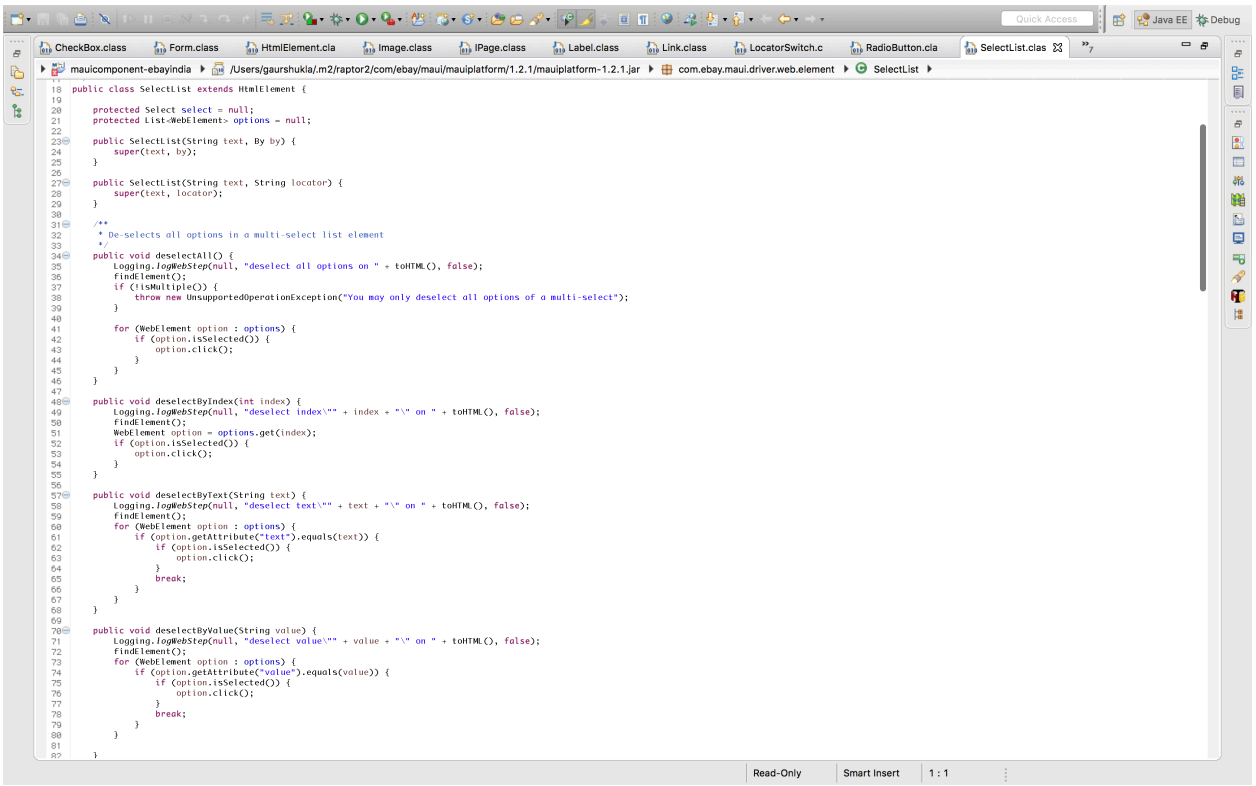
```
  IOS_LANDSCAPE("IOSLANDSCAPE"),
  /**
   * Locator type to define a element that can be located by same
   *  element irrespective of device or orientation.
   */
  IPHONE_LOCATOR("IPHONE"),
  /**
   * Locator type to define a element that has specific locator for iphone
   * landscape mode.
   */
  IPHONE_LANDSCAPE_LOCATOR("IPHONELANDSCAPE"),
  /**
   * Locator type to define a element that has specific locator for iphone
   * portrait mode.
   */
  IPHONE_PORTRAIT_LOCATOR("IPHONEPORTRAIT"),
  /**
   * Locator type to define a element that can be located by same
   *  element irrespective of orientation of the ipad (i.e. Landscape or Portrait)
   */
  IPAD_LOCATOR("IPAD"),
  /**
   * Locator type to define a element that has specific locator for ipad
   * landscape mode.
   */
  IPAD_LANDSCAPE_LOCATOR("IPADLANDSCAPE"),
  /**
   * Locator type to define a element that has specific locator for ipad
   * portrait mode.
   */
  IPAD_PORTRAIT_LOCATOR("IPADPORTRAIT"),
  /**
   * Locator type to define a element that can be located by same
   *  element irrespective of orientation of the ipad (i.e. Landscape or Portrait)
   */
  IPADMINI_LOCATOR("IPADMINI"),
  /**
   * Locator type to define a element that has specific locator for ipad
   * landscape mode.
   */
  IPADMINI_LANDSCAPE_LOCATOR("IPADMINILANDSCAPE"),
  /**
   * Locator type to define a element that has specific locator for ipad
   * portrait mode.
   */
  IPADMINI_PORTRAIT_LOCATOR("IPADMINIPORTRAIT"),
  /**
   * Locator type to define a element that can be located by same
   * element irrespective of orientation or size of the android device
   * (i.e. Landscape or Portrait or any sizes)
   */
  ANDROID_LOCATOR("ANDROID"),
  /**
   * Locator type to define a element that can be located by same
   * element irrespective of size of the android device
   * but, specific to portrait orientation.
   */
  ANDROID_PORTRAIT_LOCATOR("ANDROID_PORTRAIT"),
  /**
   * Locator type to define a element that can be located by same
   * element irrespective of size of the android device
   * but, specific to landscape orientation.
   */
  ANDROID_LANDSCAPE_LOCATOR("ANDROID_LANDSCAPE_PORTRAIT"),
  /**
```


Button.class
CheckBox.class
Form.class
HtmlElement.cla
Image.class
IPage.class
Label.class
Link.class
LocatorSwitch.c
LocatorType.cla
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web.element
LocatorType
Read-Only
Smart Insert
1 : 1

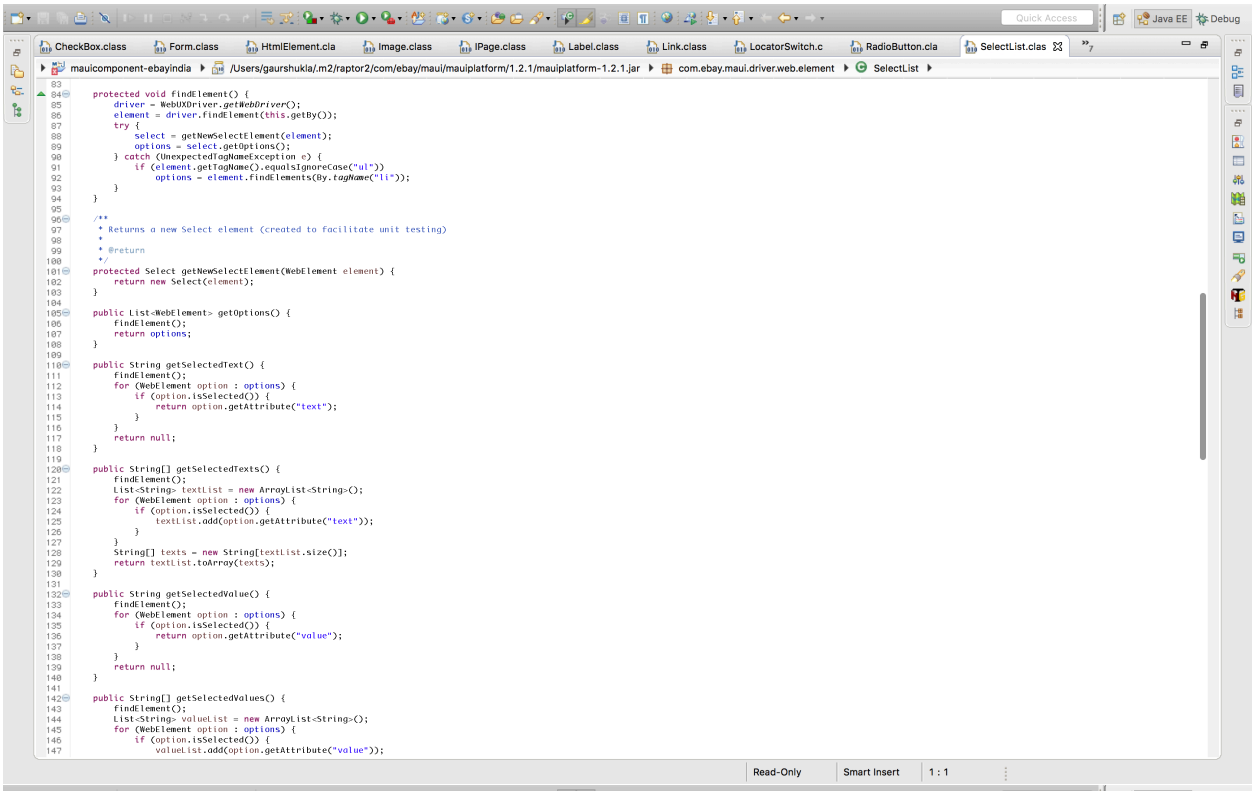

Button.class
CheckBox.class
Form.class
HtmlElement.cla
Image.class
IPage.class
Label.class
Link.class
LocatorSwitch.c
LocatorType.cla
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web.element
LocatorType
Read-Only
Smart Insert
1 : 1



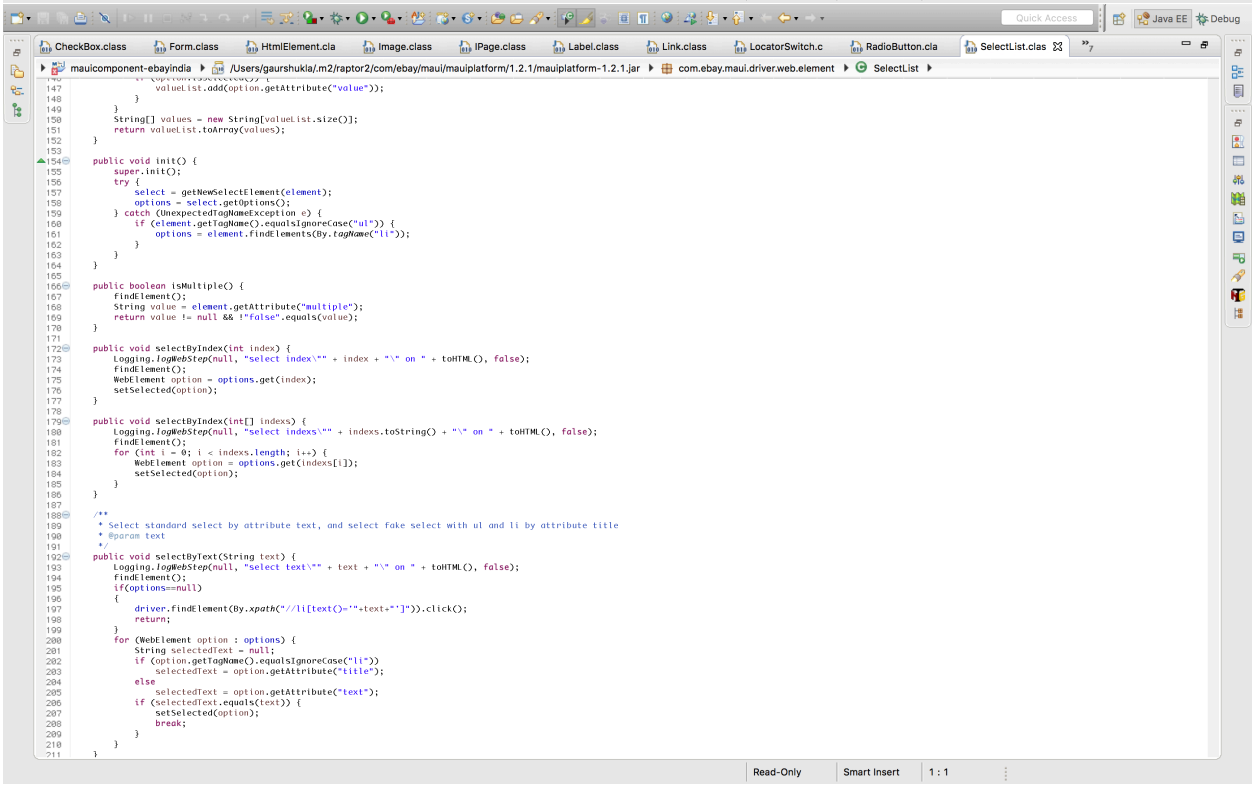
```
package com.ebay.maui.driver.web.element;

import org.openqa.selenium.By;

import com.ebay.maui.controller.Logging;

public class RadioButton extends HtmlElement {

    public RadioButton(String label, By by) {
        super(label, by);
    }

    public RadioButton(String label, String locator) {
        super(label, locator);
    }

    public void check() {
        Logging.logWebStep(null, "check " + toHTML(), false);
        super.click();
    }

    @Override
    public void click() {
        Logging.logWebStep(null, "click on " + toHTML(), false);
        super.click();
    }

    //TODO this code is repeated in other objects (like CheckBox)--there should be an AbstractClickable
    // and an IClickable where this gets specified and implemented
    public boolean isSelected() {
        findElement();
        return element.isSelected();
    }
}
```



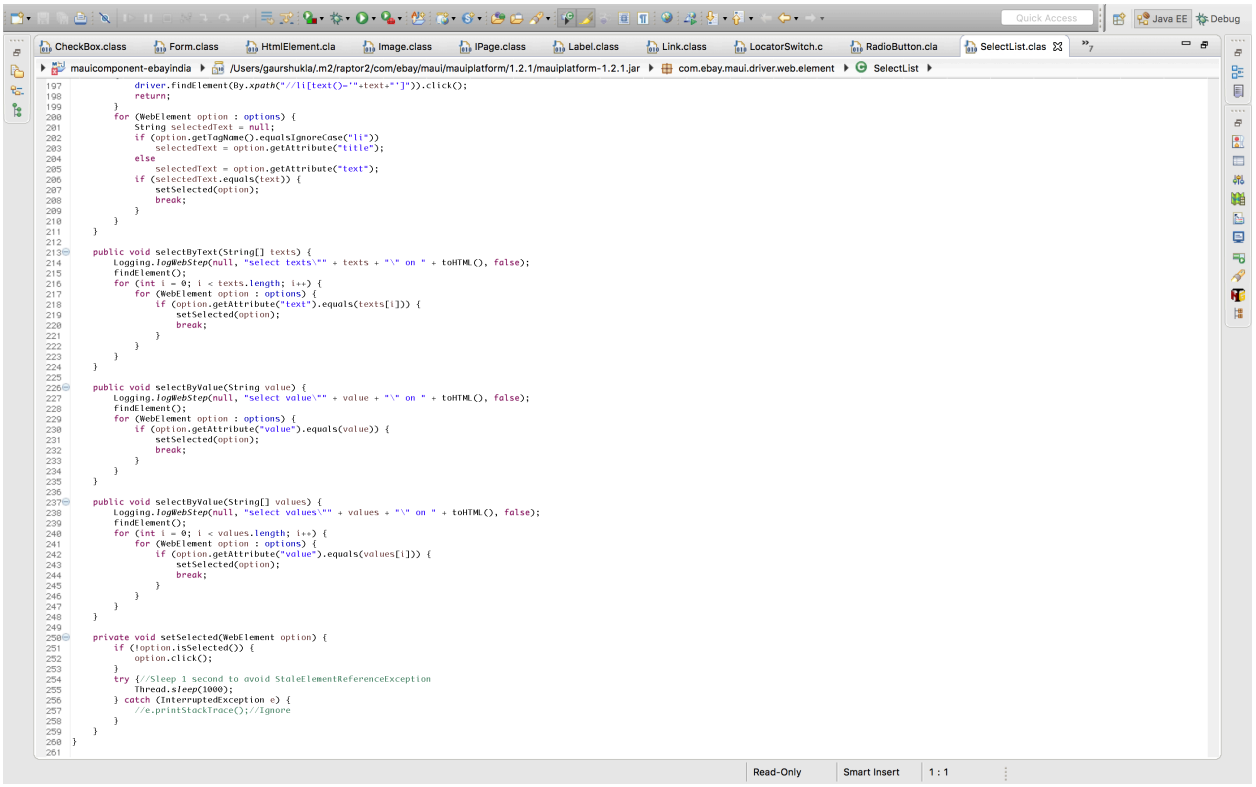
```
public class SelectList extends HtmlElement {

    protected Select select = null;
    protected List<WebElement> options = null;

    public SelectList(String text, By by) {
        super(text, by);
    }

    public SelectList(String text, String locator) {
        super(text, locator);
    }

    /**
     * De-selects all options in a multi-select list element
     */
    public void deselectAll() {
        Logging.logWebStep(null, "deselect all options on " + toHTML(), false);
        findElement();
        if (!isMultiple()) {
            throw new UnsupportedOperationException("You may only deselect all options of a multi-select");
        }

        for (WebElement option : options) {
            if (option.isSelected()) {
                option.click();
            }
        }
    }

    public void deselectByIndex(int index) {
        Logging.logWebStep(null, "deselect index\"" + index + "\" on " + toHTML(), false);
        findElement();
        WebElement option = options.get(index);
        if (option.isSelected()) {
            option.click();
        }
    }

    public void deselectByText(String text) {
        Logging.logWebStep(null, "deselect text\"" + text + "\" on " + toHTML(), false);
        findElement();
        for (WebElement option : options) {
            if (option.getAttribute("text").equals(text)) {
                if (option.isSelected()) {
                    option.click();
                }
                break;
            }
        }
    }

    public void deselectByValue(String value) {
        Logging.logWebStep(null, "deselect value\"" + value + "\" on " + toHTML(), false);
        findElement();
        for (WebElement option : options) {
            if (option.getAttribute("value").equals(value)) {
                if (option.isSelected()) {
                    option.click();
                }
                break;
            }
        }
    }
```

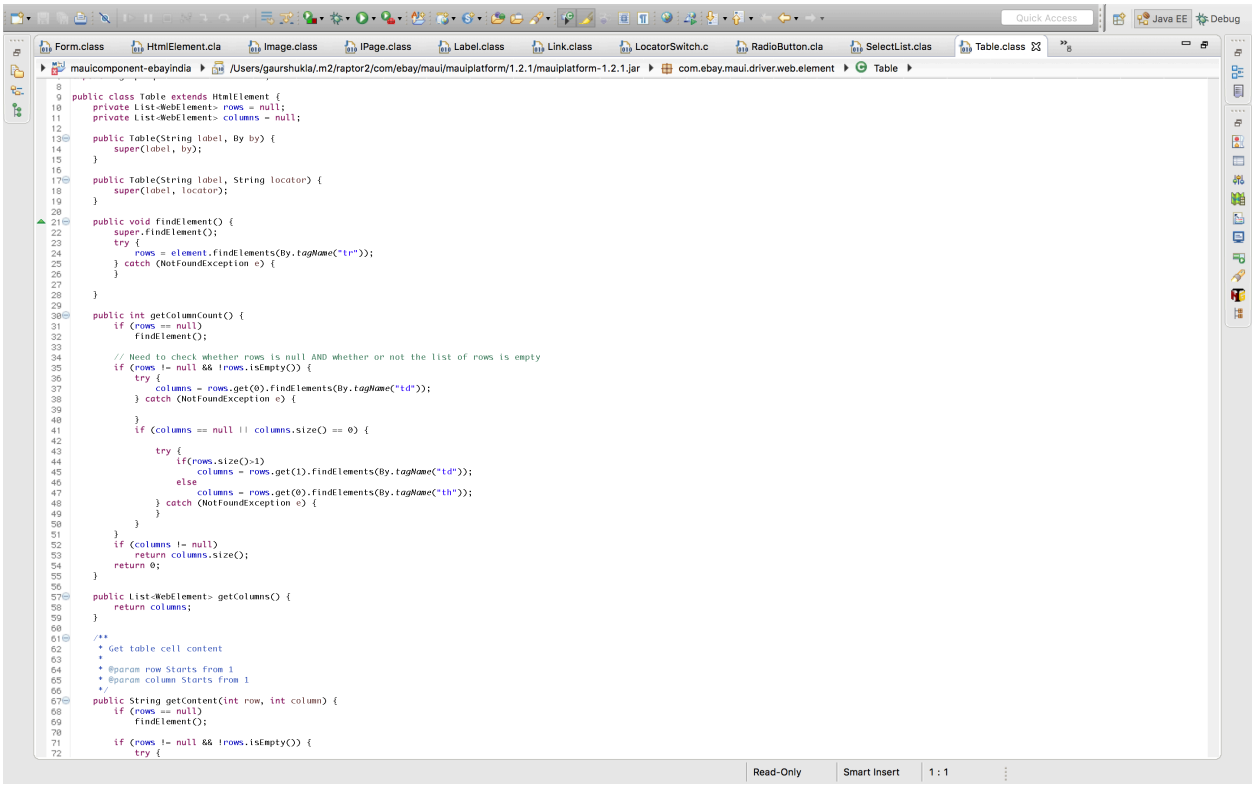


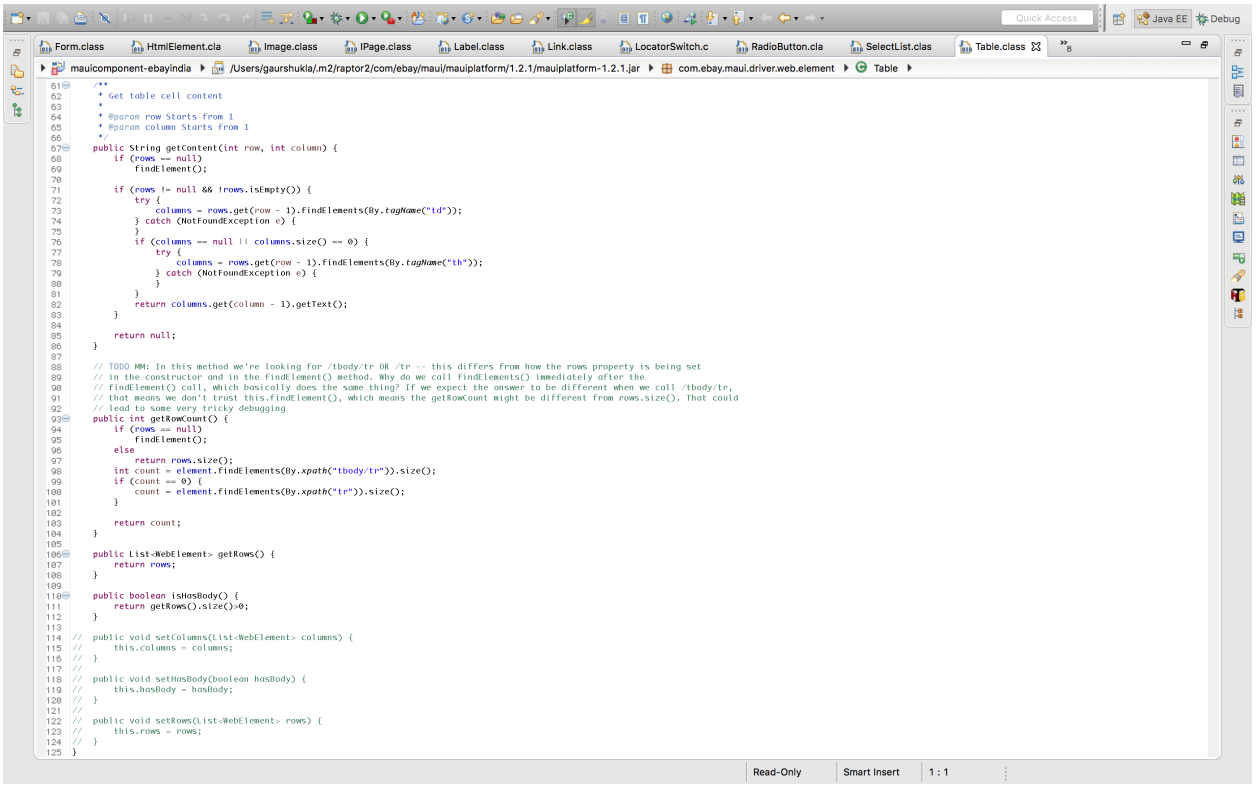
```
protected void findElement() {
    driver = WebUIDriver.getWebDriver();
    element = driver.findElement(this.getBy());
    try {
        select = getNewSelectElement(element);
        options = select.getOptions();
    } catch (UnexpectedTagNameException e) {
        if (element.getTagName().equalsIgnoreCase("ul"))
            options = element.findElements(By.tagName("li"));
    }
}

/**
 * Returns a new Select element (created to facilitate unit testing)
 *
 * @return
 */
protected Select getNewSelectElement(WebElement element) {
    return new Select(element);
}

public List<WebElement> getOptions() {
    findElement();
    return options;
}

public String getSelectedText() {
    findElement();
    for (WebElement option : options) {
        if (option.isSelected()) {
            return option.getAttribute("text");
        }
    }
    return null;
}

public String[] getSelectedTexts() {
    findElement();
    List<String> textList = new ArrayList<String>();
    for (WebElement option : options) {
        if (option.isSelected()) {
            textList.add(option.getAttribute("text"));
        }
    }
    String[] texts = new String[textList.size()];
    return textList.toArray(texts);
}

public String getSelectedValue() {
    findElement();
    for (WebElement option : options) {
        if (option.isSelected()) {
            return option.getAttribute("value");
        }
    }
    return null;
}

public String[] getSelectedValues() {
    findElement();
    List<String> valueList = new ArrayList<String>();
    for (WebElement option : options) {
        if (option.isSelected()) {
            valueList.add(option.getAttribute("value"));
```

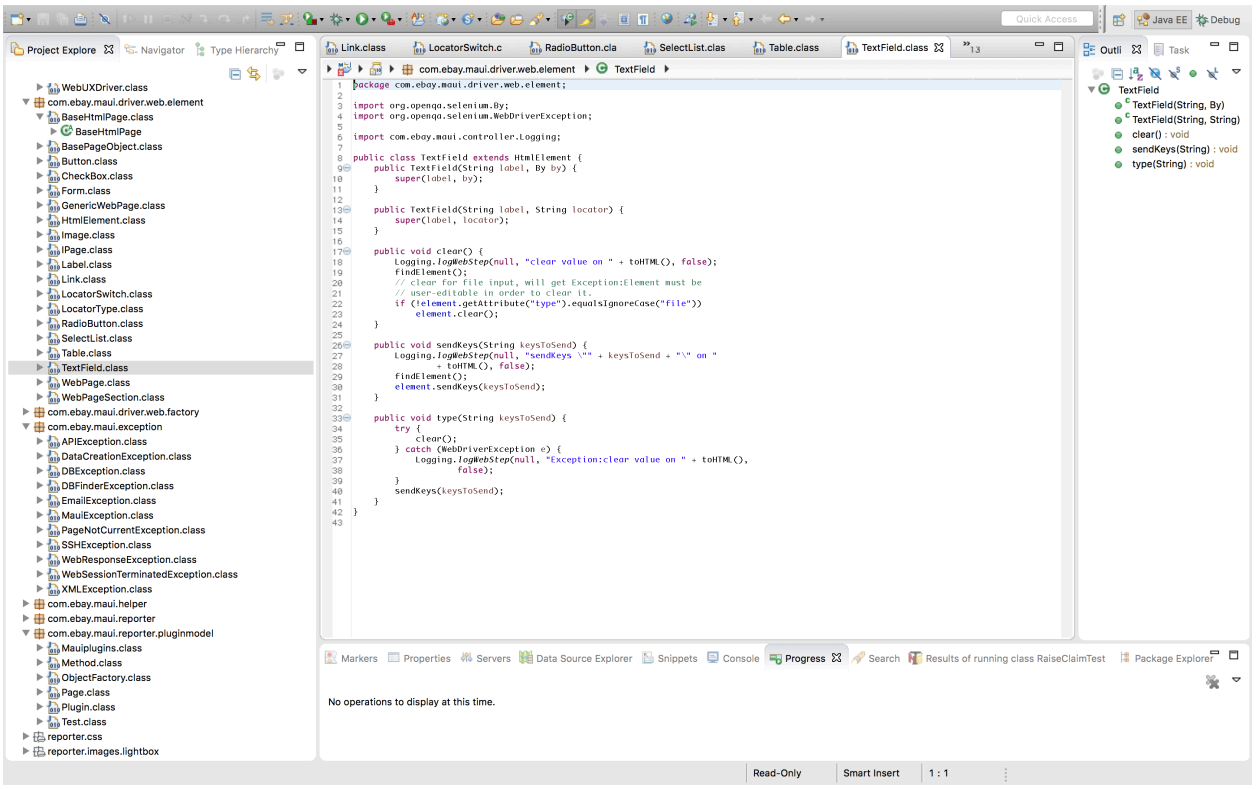


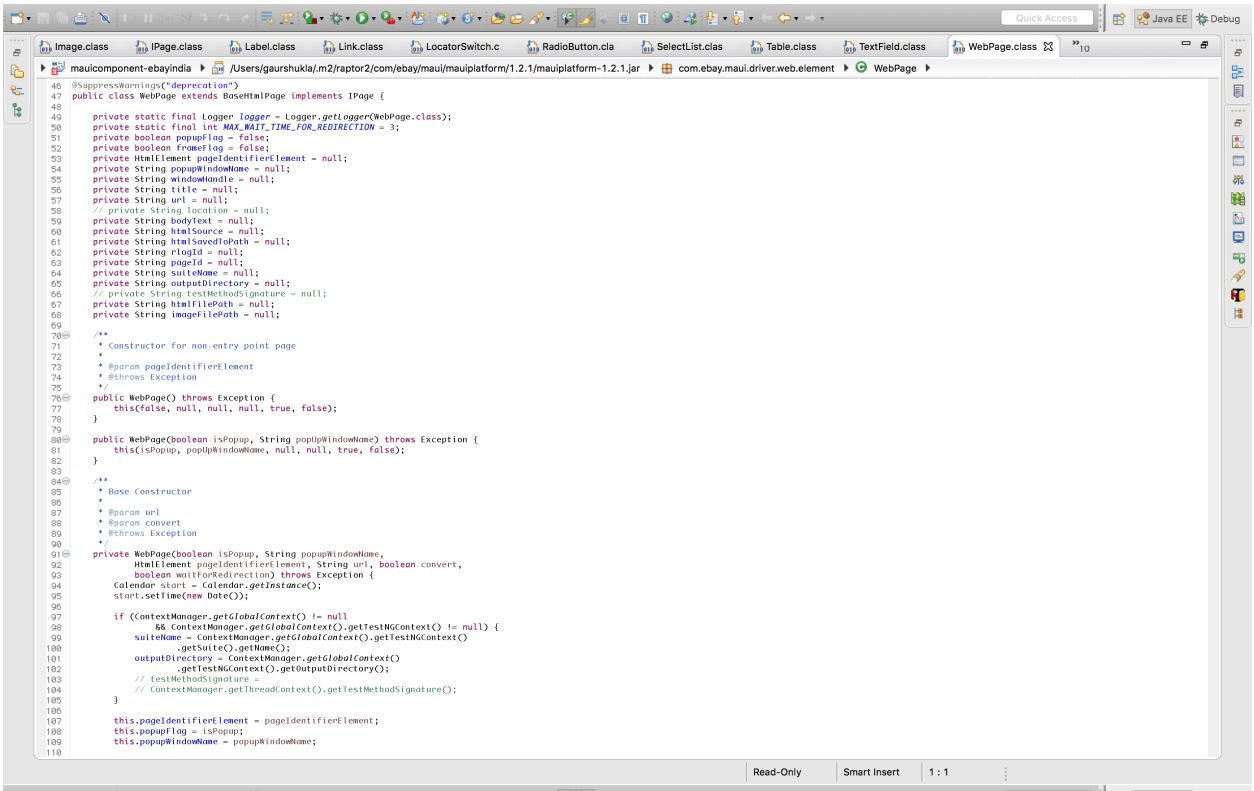
```
            valueList.add(option.getAttribute("value"));
        }
    }
    String[] values = new String[valueList.size()];
    return valueList.toArray(values);
}

public void init() {
    super.init();
    try {
        select = getNewSelectElement(element);
        options = select.getOptions();
    } catch (UnexpectedTagNameException e) {
        if (element.getTagName().equalsIgnoreCase("ul")) {
            options = element.findElements(By.tagName("li"));
        }
    }
}

public boolean isMultiple() {
    findElement();
    String value = element.getAttribute("multiple");
    return value != null && !"false".equals(value);
}

public void selectByIndex(int index) {
    Logging.logWebStep(null, "select index\"" + index + "\" on " + toHTML(), false);
    findElement();
    WebElement option = options.get(index);
    setSelected(option);
}

public void selectByIndex(int[] indexs) {
    Logging.logWebStep(null, "select indexs\"" + indexs.toString() + "\" on " + toHTML(), false);
    findElement();
    for (int i = 0; i < indexs.length; i++) {
        WebElement option = options.get(indexs[i]);
        setSelected(option);
    }
}

/**
 * Select standard select by attribute text, and select fake select with ul and li by attribute title
 * @param text
 */
public void selectByText(String text) {
    Logging.logWebStep(null, "select text\"" + text + "\" on " + toHTML(), false);
    findElement();
    if(options==null)
    {
        driver.findElement(By.xpath("//li[text()='"+text+"']")).click();
        return;
    }
    for (WebElement option : options) {
        String selectedText = null;
        if (option.getTagName().equalsIgnoreCase("li"))
            selectedText = option.getAttribute("title");
        else
            selectedText = option.getAttribute("text");
        if (selectedText.equals(text)) {
            setSelected(option);
            break;
        }
    }
}
```


CheckBox.class
Form.class
HtmlElement.cla
Image.class
IPage.class
Label.class
Link.class
LocatorSwitch.c
RadioButton.cla
SelectList.clas
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web.element
SelectList
Read-Only
Smart Insert
1 : 1

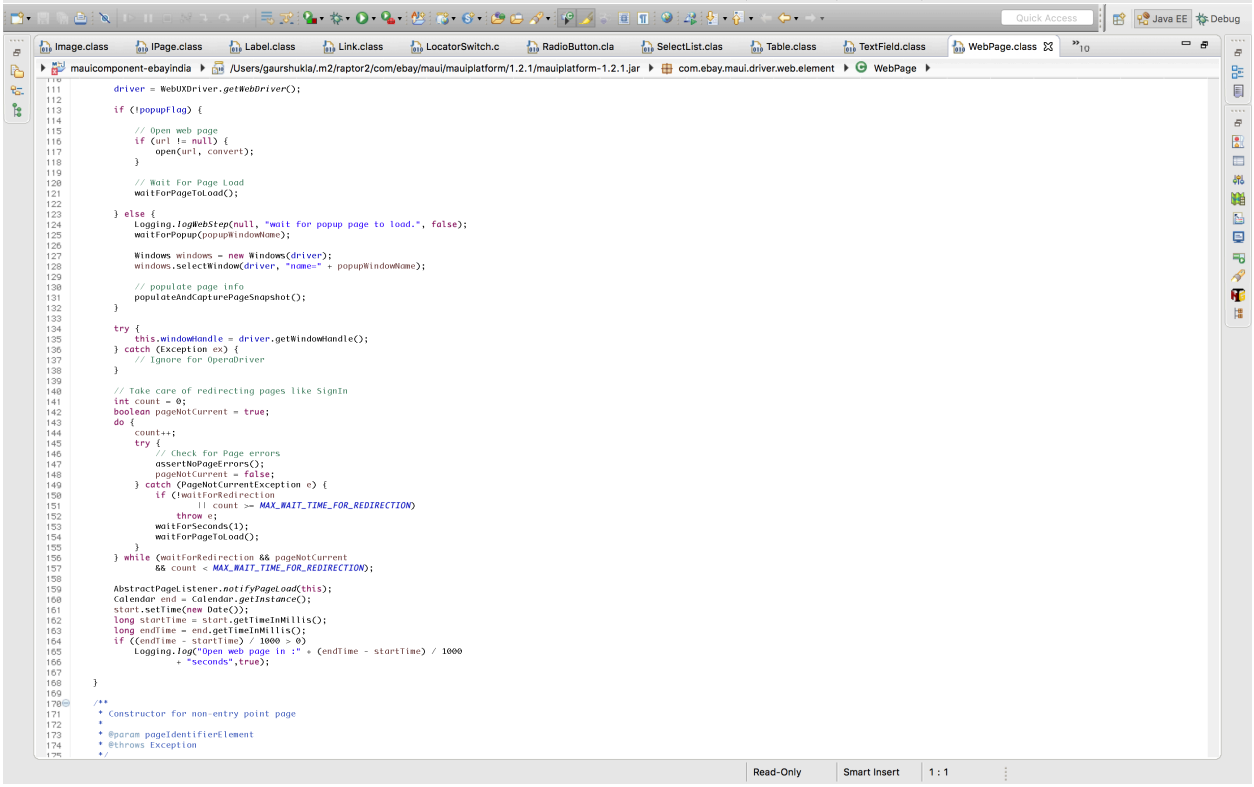

Form.class
HtmlElement.cla
Image.class
IPage.class
Label.class
Link.class
LocatorSwitch.c
RadioButton.cla
SelectList.clas
Table.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web.element
Table
Read-Only
Smart Insert
1 : 1


Form.class
HtmlElement.cla
Image.class
IPage.class
Label.class
Link.class
LocatorSwitch.c
RadioButton.cla
SelectList.clas
Table.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web.element
Table
Read-Only
Smart Insert
1 : 1

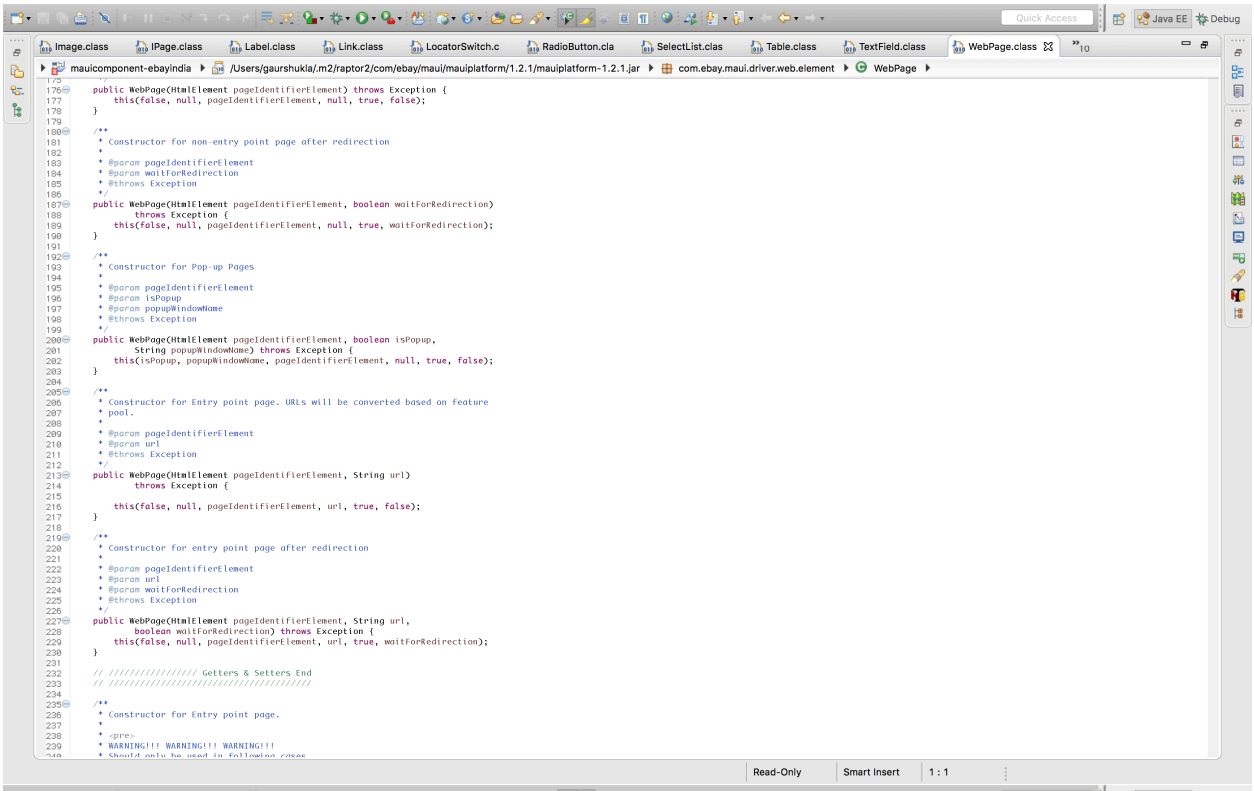

Project Explore
Navigator
Type Hierarchy
WebUXDriver.class
com.ebay.maui.driver.web.element
BaseHtmlPage.class
BaseHtmlPage
BasePageObject.class
Button.class
CheckBox.class
Form.class
GenericWebPage.class
HtmlElement.class
Image.class
IPage.class
Label.class
Link.class
LocatorSwitch.class
LocatorType.class
RadioButton.class
SelectList.class
Table.class
TextField.class
WebPage.class
WebPageSection.class
com.ebay.maui.driver.web.factory
com.ebay.maui.exception
APIException.class
DataCreationException.class
DBException.class
DBFinderException.class
EmailException.class
MauiException.class
PageNotCurrentException.class
SSHException.class
WebResponseException.class
WebSessionTerminatedException.class
XMLException.class
com.ebay.maui.helper
com.ebay.maui.reporter
com.ebay.maui.reporter.pluginmodel
Mauiplugins.class
Method.class
ObjectFactory.class
Page.class
Plugin.class
Test.class
reporter.css
reporter.images.lightbox
Link.class
LocatorSwitch.c
RadioButton.cla
SelectList.clas
Table.class
TextField.class
com.ebay.maui.driver.web.element
TextField
Outli
Task
TextField
TextField(String, By)
TextField(String, String)
clear() : void
sendKeys(String) : void
type(String) : void
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

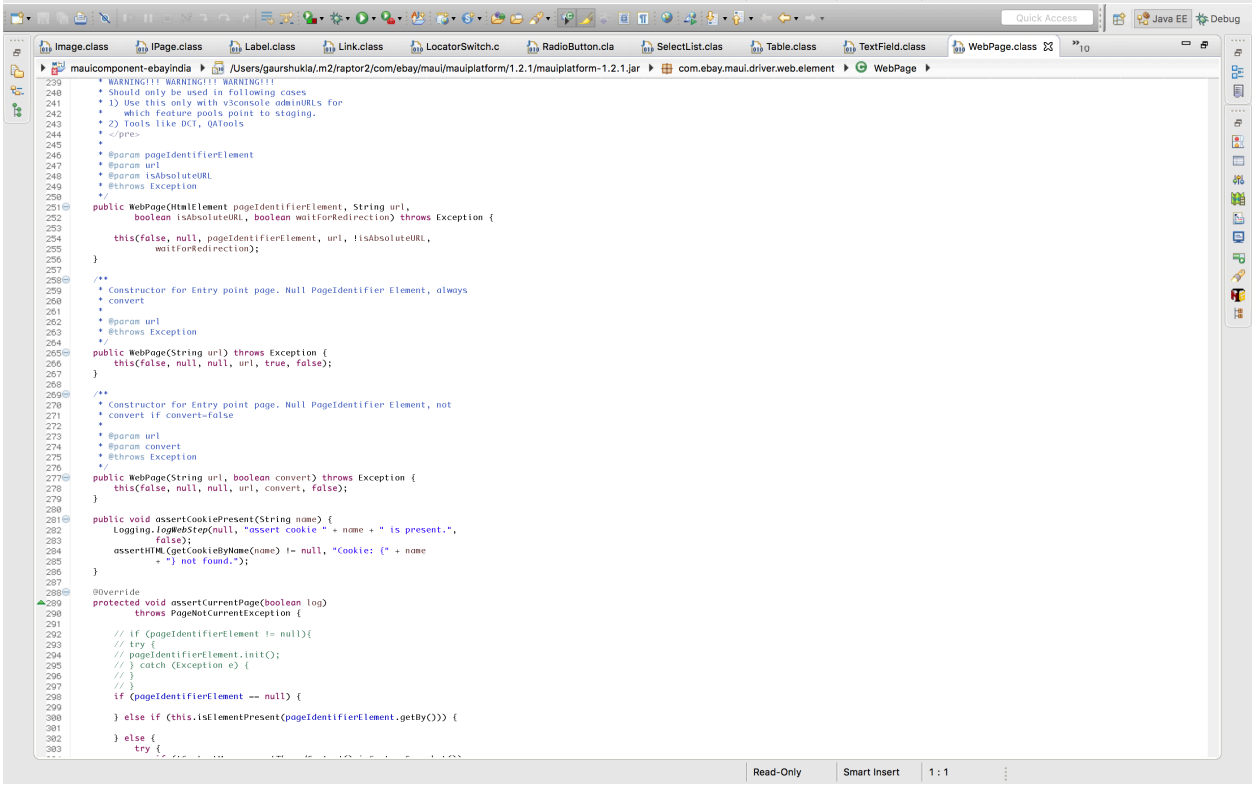


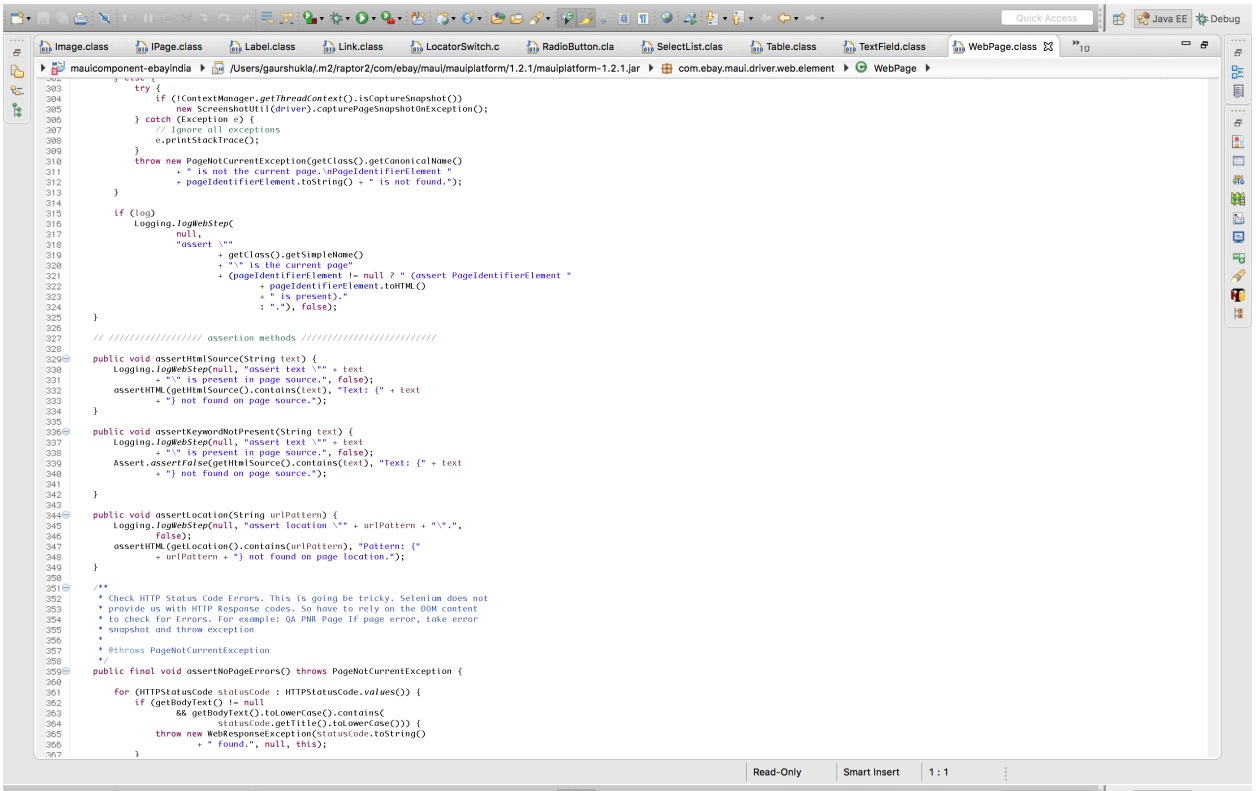
```
@SuppressWarnings("deprecation")
public class WebPage extends BaseHtmlPage implements IPage {

    private static final Logger logger = Logger.getLogger(WebPage.class);
    private static final int MAX_WAIT_TIME_FOR_REDIRECTION = 3;
    private boolean popupFlag = false;
    private boolean frameFlag = false;
    private HtmlElement pageIdentifierElement = null;
    private String popupWindowName = null;
    private String windowHandle = null;
    private String title = null;
    private String url = null;
    // private String location = null;
    private String bodyText = null;
    private String htmlSource = null;
    private String htmlSavedToPath = null;
    private String rlogId = null;
    private String pageId = null;
    private String suiteName = null;
    private String outputDirectory = null;
    // private String testMethodSignature = null;
    private String htmlFilePath = 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, String popupWindowName,
            HtmlElement pageIdentifierElement, String url, boolean convert,
            boolean waitForRedirection) throws Exception {
        Calendar start = Calendar.getInstance();
        start.setTime(new Date());

        if (ContextManager.getGlobalContext() != null
                && ContextManager.getGlobalContext().getTestNGContext() != null) {
            suiteName = ContextManager.getGlobalContext().getTestNGContext()
                    .getSuite().getName();
            outputDirectory = ContextManager.getGlobalContext()
                    .getTestNGContext().getOutputDirectory();
            // testMethodSignature =
            // ContextManager.getThreadContext().getTestMethodSignature();
        }

        this.pageIdentifierElement = pageIdentifierElement;
        this.popupFlag = isPopup;
        this.popupWindowName = popupWindowName;
```



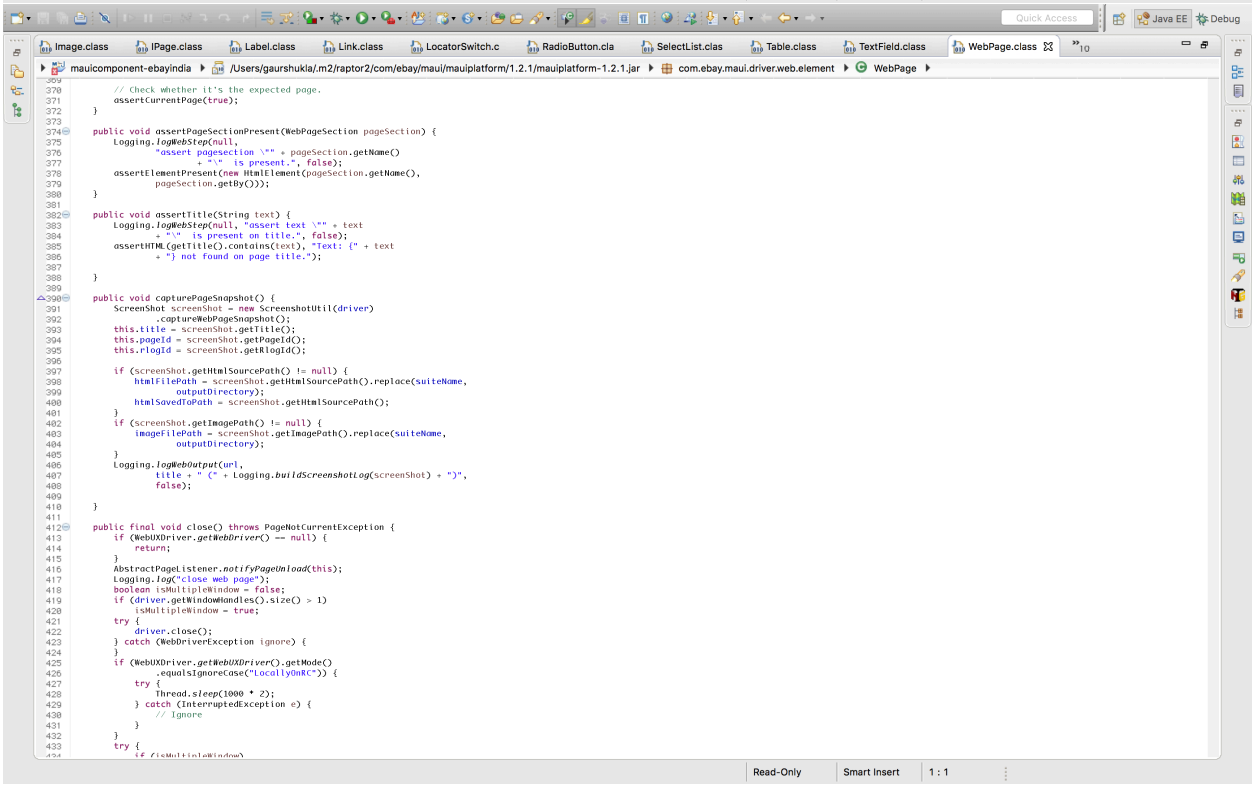
```
        driver = WebUXDriver.getWebDriver();

        if (!popupFlag) {

            // Open web page
            if (url != null) {
                open(url, convert);
            }

            // Wait For Page Load
            waitForPageToLoad();

        } else {
            Logging.logWebStep(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 OperaDriver
        }

        // 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
                        || count >= MAX_WAIT_TIME_FOR_REDIRECTION)
                    throw e;
                waitForSeconds(1);
                waitForPageToLoad();
            }
        } 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
     */
```


Quick Access
Java EE
Debug
Image.class
IPage.class
Label.class
Link.class
LocatorSwitch.c
RadioButton.cla
SelectList.clas
Table.class
TextField.class
WebPage.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web.element
WebPage
Read-Only
Smart Insert
1 : 1

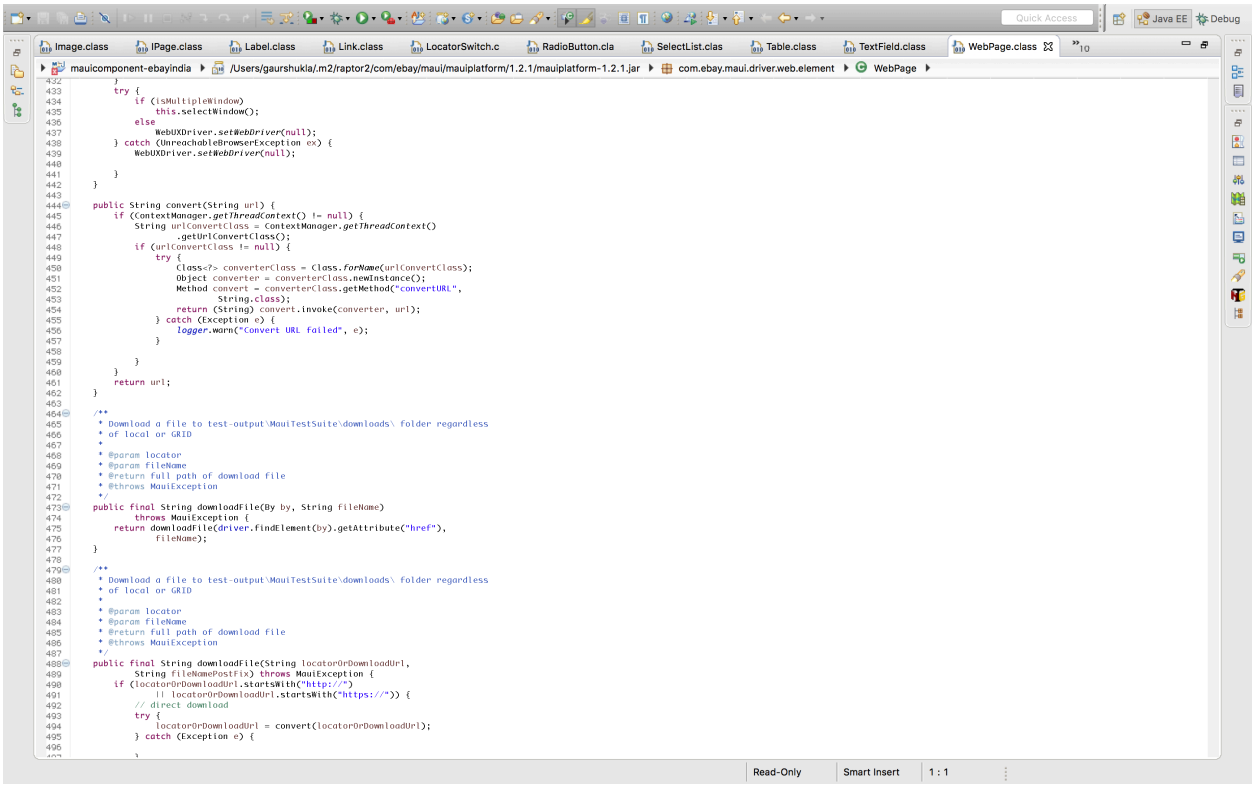

Quick Access
Java EE
Debug
Image.class
IPage.class
Label.class
Link.class
LocatorSwitch.c
RadioButton.cla
SelectList.clas
Table.class
TextField.class
WebPage.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web.element
WebPage
Read-Only
Smart Insert
1 : 1



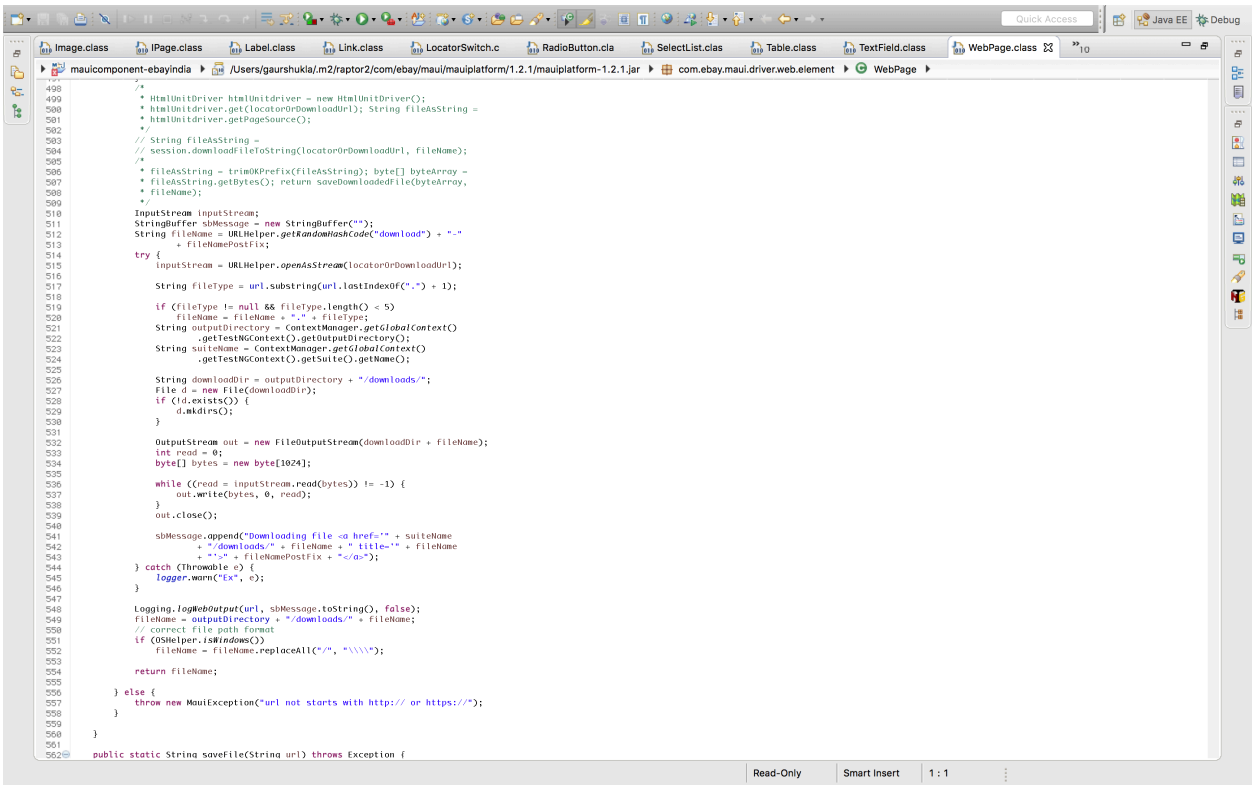
```
303         try {
304             if (!ContextManager.getThreadContext().isCaptureSnapshot())
305                 new ScreenshotUtil(driver).capturePageSnapshotOnException();
306         } catch (Exception e) {
307             // Ignore all exceptions
308             e.printStackTrace();
309         }
310         throw new PageNotCurrentException(getClass().getCanonicalName()
311                 + " is not the current page.\nPageIdentifierElement "
312                 + pageIdentifierElement.toString() + " is not found.");
313     }
314
315     if (log)
316         Logging.logWebStep(
317                 null,
318                 "assert \""
319                         + getClass().getSimpleName()
320                         + "\" is the current page"
321                         + (pageIdentifierElement != null ? " (assert PageIdentifierElement "
322                                 + pageIdentifierElement.toHTML()
323                                 + " is present)."
324                                 : "."), false);
325 }
326
327 // ////////////////// assertion methods //////////////////////////
328
329 public void assertHtmlSource(String text) {
330     Logging.logWebStep(null, "assert text \"" + text
331             + "\" is present in page source.", false);
332     assertHTML(getHtmlSource().contains(text), "Text: {" + text
333             + "} not found on page source.");
334 }
335
336 public void assertKeywordNotPresent(String text) {
337     Logging.logWebStep(null, "assert text \"" + text
338             + "\" is present in page source.", false);
339     Assert.assertFalse(getHtmlSource().contains(text), "Text: {" + text
340             + "} not found on page source.");
341
342 }
343
344 public void assertLocation(String urlPattern) {
345     Logging.logWebStep(null, "assert location \"" + urlPattern + "\".",
346             false);
347     assertHTML(getLocation().contains(urlPattern), "Pattern: {"
348             + urlPattern + "} not found on page location.");
349 }
350
351 /**
352  * Check HTTP Status Code Errors. This is going be tricky. Selenium does not
353  * provide us with HTTP Response codes. So have to rely on the DOM content
354  * to check for Errors. For example: QA PNR Page If page error, take error
355  * snapshot and throw exception
356  *
357  * @throws PageNotCurrentException
358  */
359 public final void assertNoPageErrors() throws PageNotCurrentException {
360
361     for (HTTPStatusCode statusCode : HTTPStatusCode.values()) {
362         if (getBodyText() != null
363                 && getBodyText().toLowerCase().contains(
364                         statusCode.getTitle().toLowerCase())) {
365             throw new WebResponseException(statusCode.toString()
366                     + " found.", null, this);
367         }
```



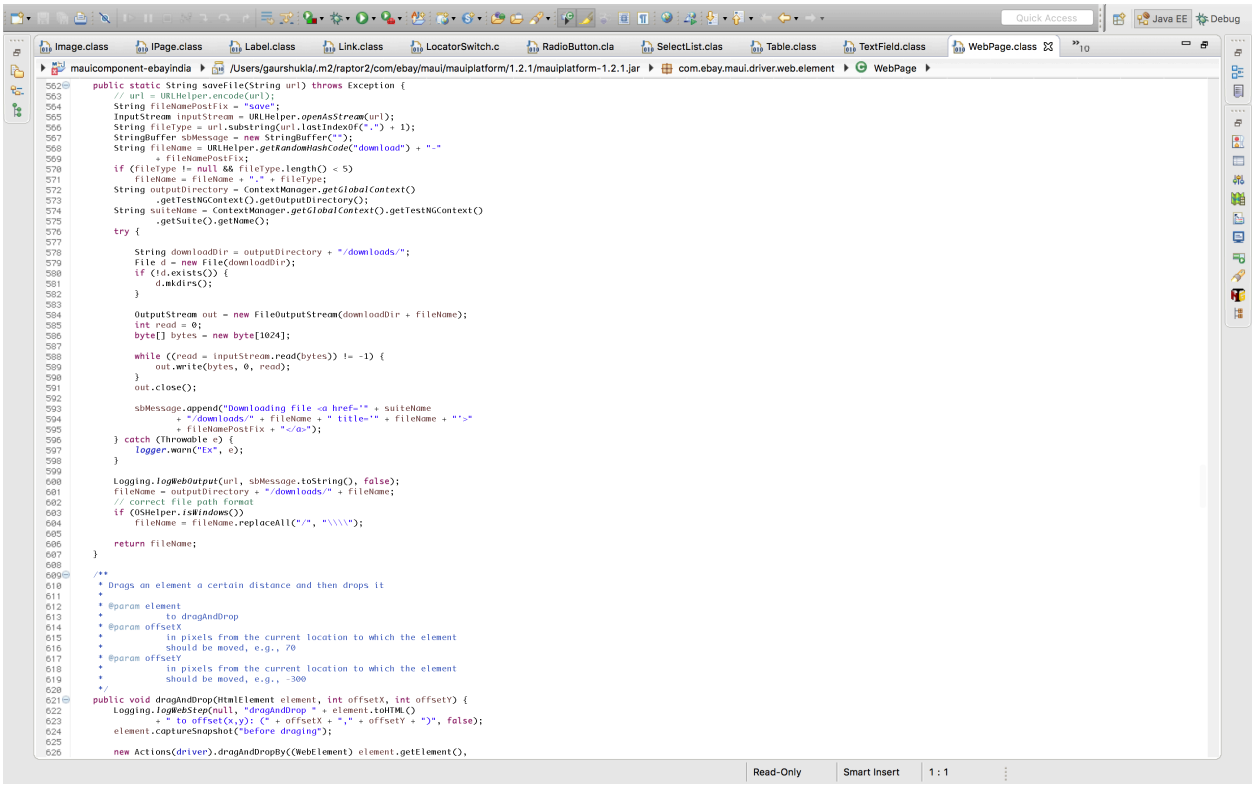
```
370     // Check whether it's the expected page.
371     assertCurrentPage(true);
372 }
373
374 public void assertPageSectionPresent(WebPageSection pageSection) {
375     Logging.logWebStep(null,
376             "assert pagesection \"" + pageSection.getName()
377                     + "\" is present.", false);
378     assertElementPresent(new HtmlElement(pageSection.getName(),
379             pageSection.getBy()));
380 }
381
382 public void assertTitle(String text) {
383     Logging.logWebStep(null, "assert text \"" + text
384             + "\"  is present on title.", false);
385     assertHTML(getTitle().contains(text), "Text: {" + text
386             + "} not found on page title.");
387
388 }
389
390 public void capturePageSnapshot() {
391     ScreenShot screenShot = new ScreenshotUtil(driver)
392             .captureWebPageSnapshot();
393     this.title = screenShot.getTitle();
394     this.pageId = screenShot.getPageId();
395     this.rlogId = screenShot.getRlogId();
396
397     if (screenShot.getHtmlSourcePath() != null) {
398         htmlFilePath = screenShot.getHtmlSourcePath().replace(suiteName,
399                 outputDirectory);
400         htmlSavedToPath = screenShot.getHtmlSourcePath();
401     }
402     if (screenShot.getImagePath() != null) {
403         imageFilePath = screenShot.getImagePath().replace(suiteName,
404                 outputDirectory);
405     }
406     Logging.logWebOutput(url,
407             title + " (" + Logging.buildScreenshotLog(screenShot) + ")",
408             false);
409
410 }
411
412 public final void close() throws PageNotCurrentException {
413     if (WebUXDriver.getWebDriver() == null) {
414         return;
415     }
416     AbstractPageListener.notifyPageUnload(this);
417     Logging.log("close web page");
418     boolean isMultipleWindow = false;
419     if (driver.getWindowHandles().size() > 1)
420         isMultipleWindow = true;
421     try {
422         driver.close();
423     } catch (WebDriverException ignore) {
424     }
425     if (WebUXDriver.getWebUXDriver().getMode()
426             .equalsIgnoreCase("LocallyOnRC")) {
427         try {
428             Thread.sleep(1000 * 2);
429         } catch (InterruptedException e) {
430             // Ignore
431         }
432     }
433     try {
434         if (isMultipleWindow)
```



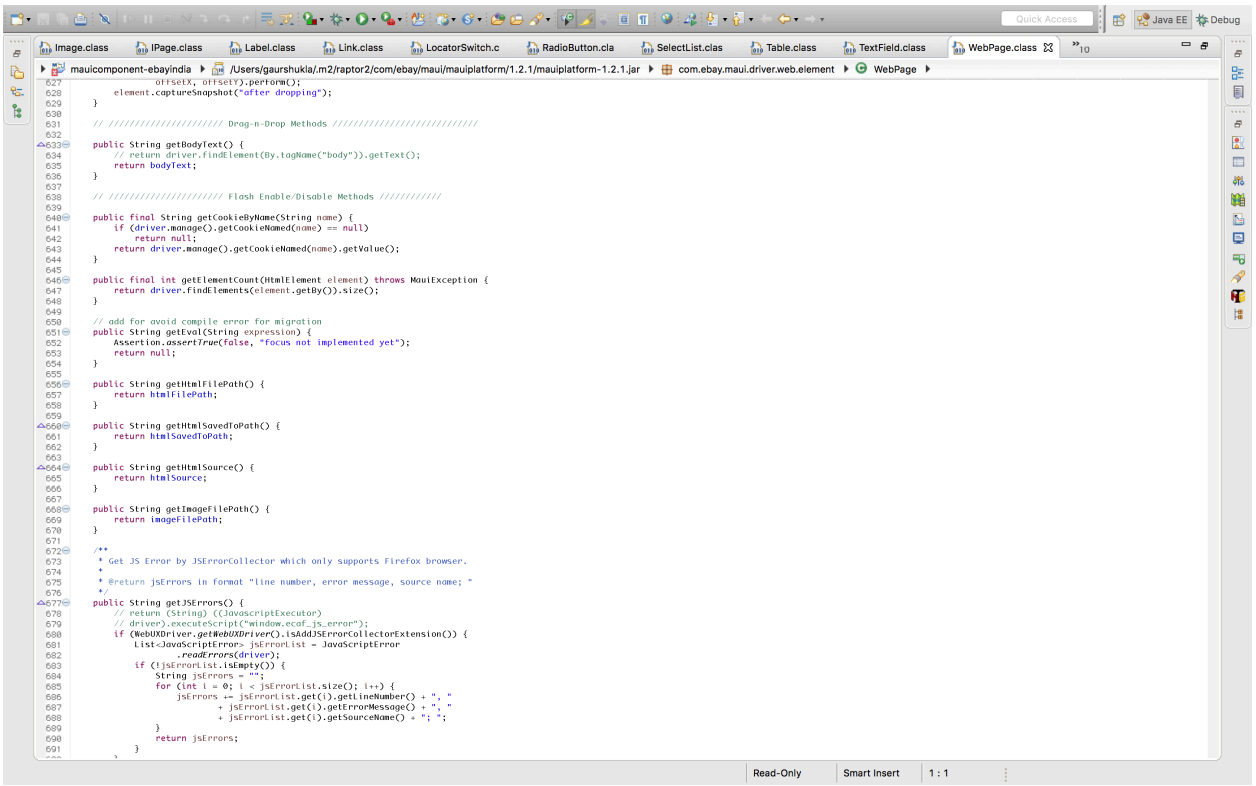
```
        }
        try {
            if (isMultipleWindow)
                this.selectWindow();
            else
                WebUXDriver.setWebDriver(null);
        } catch (UnreachableBrowserException ex) {
            WebUXDriver.setWebDriver(null);

        }
    }

    public String convert(String url) {
        if (ContextManager.getThreadContext() != null) {
            String urlConvertClass = ContextManager.getThreadContext()
                    .getUrlConvertClass();
            if (urlConvertClass != null) {
                try {
                    Class<?> converterClass = Class.forName(urlConvertClass);
                    Object converter = converterClass.newInstance();
                    Method convert = converterClass.getMethod("convertURL",
                            String.class);
                    return (String) convert.invoke(converter, url);
                } catch (Exception e) {
                    logger.warn("Convert URL failed", e);
                }

            }
        }
        return url;
    }

    /**
     * Download a file to test-output\MauiTestSuite\downloads\ folder regardless
     * of local or GRID
     *
     * @param locator
     * @param fileName
     * @return full path of download file
     * @throws MauiException
     */
    public final String downloadFile(By by, String fileName)
            throws MauiException {
        return downloadFile(driver.findElement(by).getAttribute("href"),
                fileName);
    }

    /**
     * Download a file to test-output\MauiTestSuite\downloads\ folder regardless
     * of local or GRID
     *
     * @param locator
     * @param fileName
     * @return full path of download file
     * @throws MauiException
     */
    public final String downloadFile(String locatorOrDownloadUrl,
            String fileNamePostFix) throws MauiException {
        if (locatorOrDownloadUrl.startsWith("http://")
                || locatorOrDownloadUrl.startsWith("https://")) {
            // direct download
            try {
                locatorOrDownloadUrl = convert(locatorOrDownloadUrl);
            } catch (Exception e) {
```



```
            /*
             * HtmlUnitDriver htmlUnitdriver = new HtmlUnitDriver();
             * htmlUnitdriver.get(locatorOrDownloadUrl); String fileAsString =
             * htmlUnitdriver.getPageSource();
             */
            // String fileAsString =
            // session.downloadFileToString(locatorOrDownloadUrl, fileName);
            /*
             * fileAsString = trimOKPrefix(fileAsString); byte[] byteArray =
             * fileAsString.getBytes(); return saveDownloadedFile(byteArray,
             * fileName);
             */
            InputStream inputStream;
            StringBuffer sbMessage = new StringBuffer("");
            String fileName = URLHelper.getRandomHashCode("download") + "-"
                    + fileNamePostFix;
            try {
                inputStream = URLHelper.openAsStream(locatorOrDownloadUrl);

                String fileType = url.substring(url.lastIndexOf(".") + 1);

                if (fileType != null && fileType.length() < 5)
                    fileName = fileName + "." + fileType;
                String outputDirectory = ContextManager.getGlobalContext()
                        .getTestNGContext().getOutputDirectory();
                String suiteName = ContextManager.getGlobalContext()
                        .getTestNGContext().getSuite().getName();

                String downloadDir = outputDirectory + "/downloads/";
                File d = new File(downloadDir);
                if (!d.exists()) {
                    d.mkdirs();
                }

                OutputStream 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
                        + "'>" + fileNamePostFix + "</a>");
            } 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;

        } else {
            throw new MauiException("url not starts with http:// or https://");
        }

    }

    public static String saveFile(String url) throws Exception {
```




```
    public static String saveFile(String url) throws Exception {
        // url = URLHelper.encode(url);
        String fileNamePostFix = "save";
        InputStream inputStream = URLHelper.openAsStream(url);
        String fileType = url.substring(url.lastIndexOf(".") + 1);
        StringBuffer sbMessage = new StringBuffer("");
        String fileName = URLHelper.getRandomHashCode("download") + "-"
                + fileNamePostFix;
        if (fileType != null && fileType.length() < 5)
            fileName = fileName + "." + fileType;
        String outputDirectory = 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.mkdirs();
            }

            OutputStream 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 + "'>"
                    + fileNamePostFix + "</a>");
        } 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
     *
     * @param element
     *            to dragAndDrop
     * @param offsetX
     *            in pixels from the current location to which the element
     *            should be moved, e.g., 70
     * @param offsetY
     *            in pixels from the current location to which the element
     *            should be moved, e.g., -300
     */
    public void dragAndDrop(HtmlElement element, int offsetX, int offsetY) {
        Logging.logWebStep(null, "dragAndDrop " + element.toHTML()
                + " to offset(x,y): (" + offsetX + "," + offsetY + ")", false);
        element.captureSnapshot("before draging");

        new Actions(driver).dragAndDropBy((WebElement) element.getElement(),
```




```
                offsetX, offsetY).perform();
        element.captureSnapshot("after dropping");
    }

    // ********************* Drag-n-Drop Methods ****************************

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

    // ********************* Flash Enable/Disable Methods ***********

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

    public final int getElementCount(HtmlElement element) throws MauiException {
        return driver.findElements(element.getBy()).size();
    }

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

    public String getHtmlFilePath() {
        return htmlFilePath;
    }

    public String getHtmlSavedToPath() {
        return htmlSavedToPath;
    }

    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() {
        // return (String) ((JavascriptExecutor)
        // driver).executeScript("window.ecaf_js_error");
        if (WebUXDriver.getWebUXDriver().isAddJSErrorCollectorExtension()) {
            List<JavaScriptError> jsErrorList = JavaScriptError
                    .readErrors(driver);
            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;
            }
```

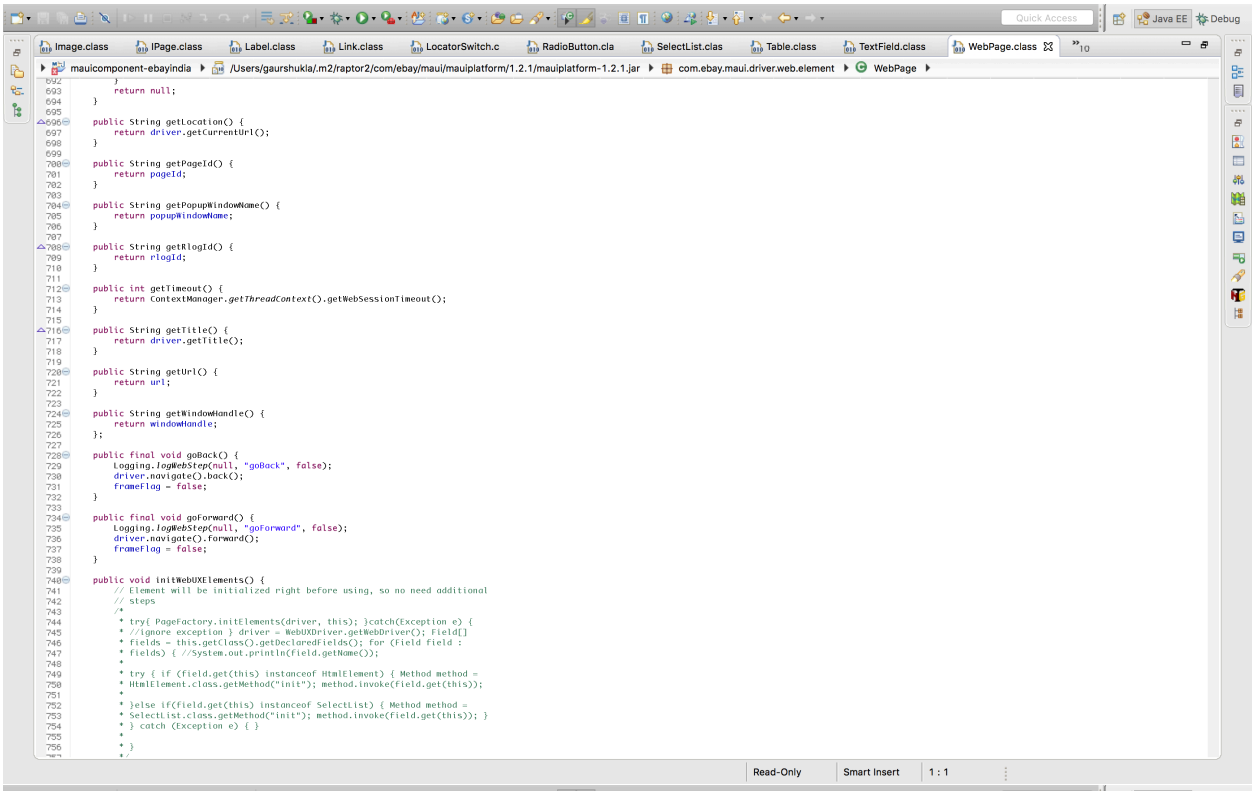


Image.class
IPage.class
Label.class
Link.class
LocatorSwitch.c
RadioButton.cla
SelectList.clas
Table.class
TextField.class
WebPage.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web.element
WebPage
Quick Access
Java EE
Debug
Read-Only
Smart Insert
1 : 1

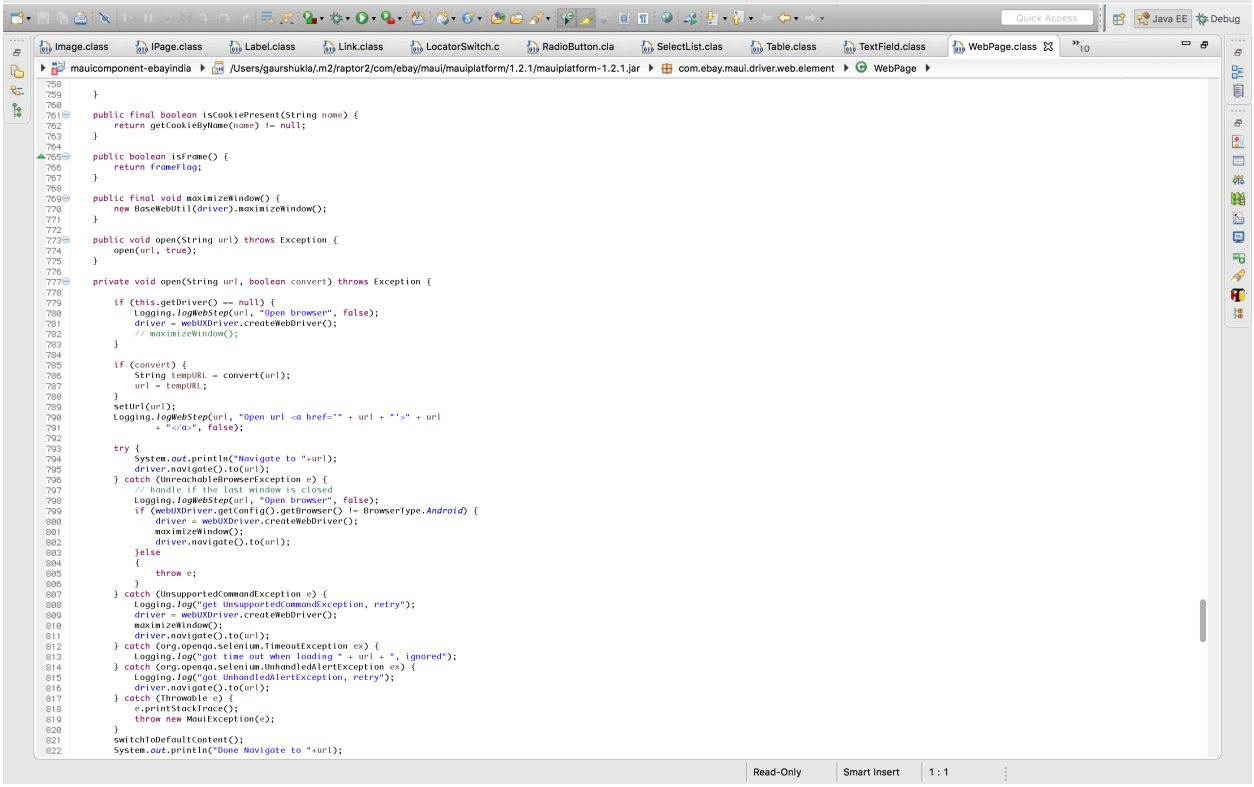
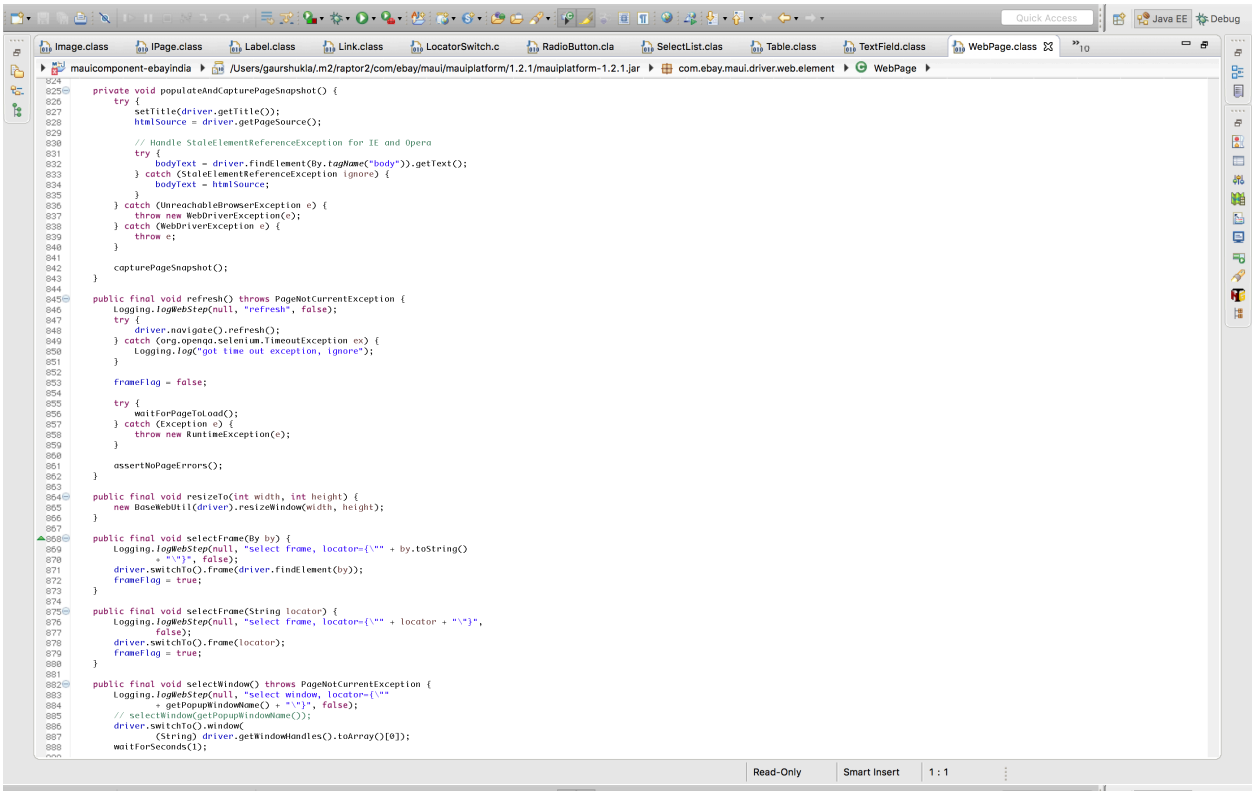
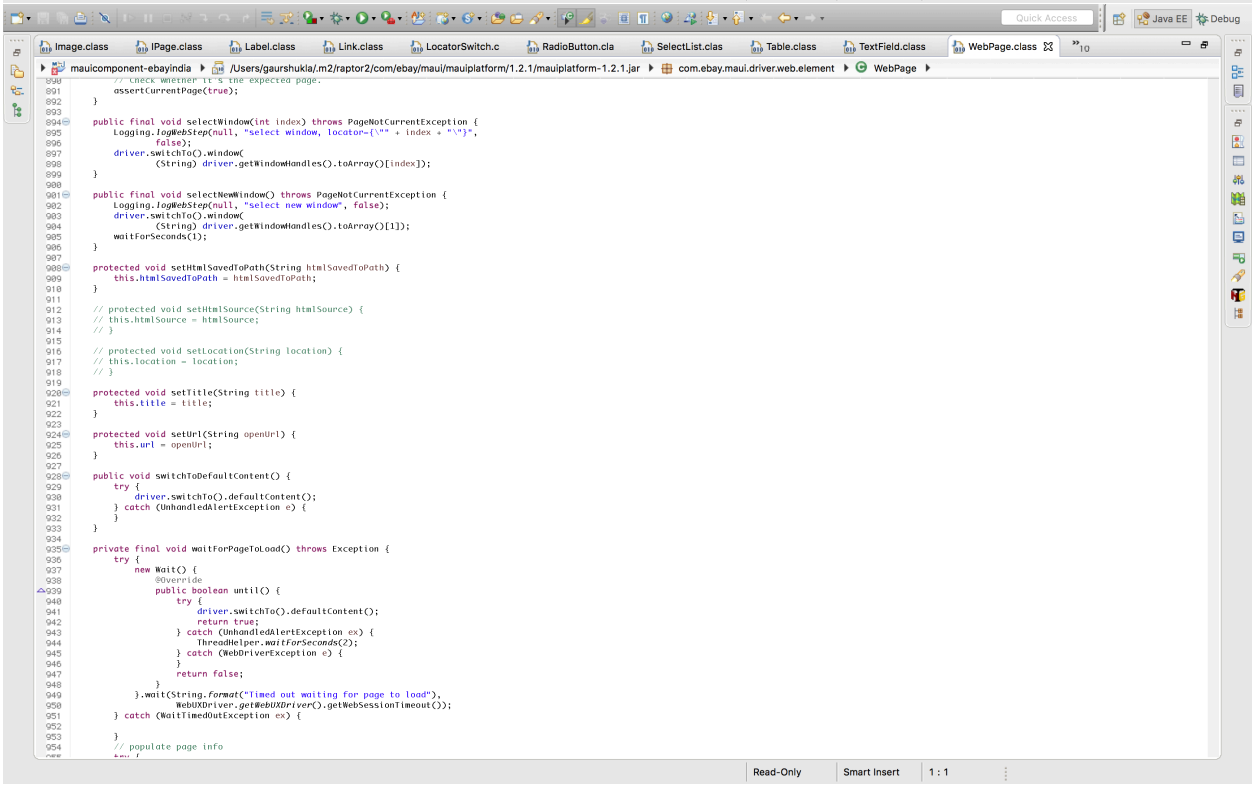


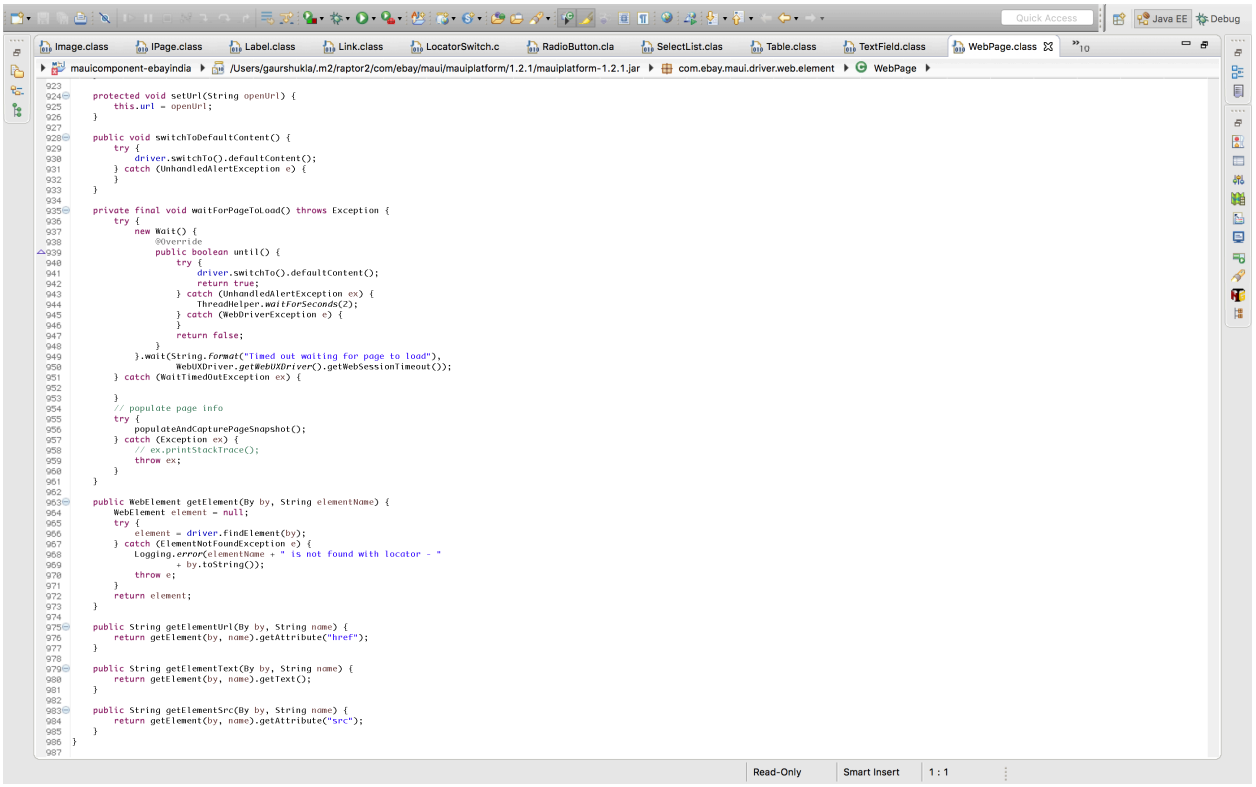
Image.class
IPage.class
Label.class
Link.class
LocatorSwitch.c
RadioButton.cla
SelectList.clas
Table.class
TextField.class
WebPage.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web.element
WebPage
Quick Access
Java EE
Debug
Read-Only
Smart Insert
1 : 1



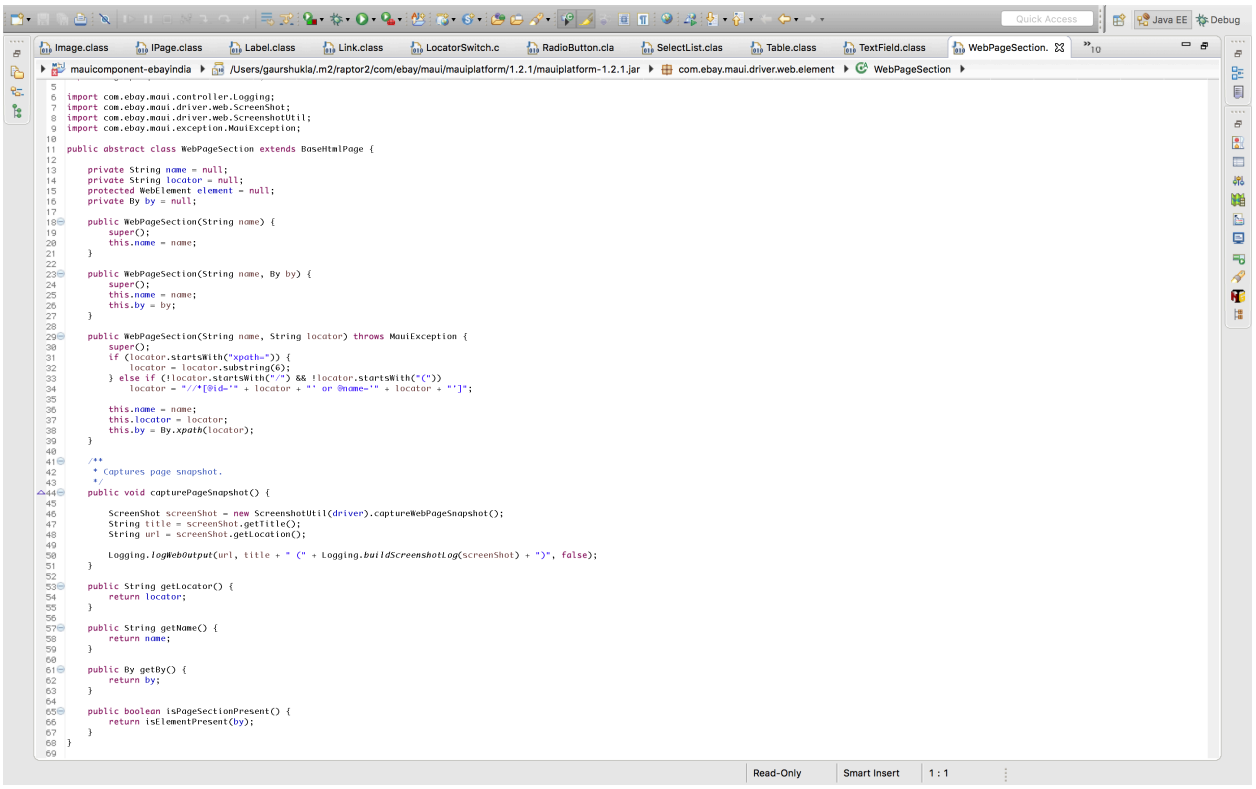
```
825    private void populateAndCapturePageSnapshot() {
826        try {
827            setTitle(driver.getTitle());
828            htmlSource = driver.getPageSource();
829
830            // Handle StaleElementReferenceException for IE and Opera
831            try {
832                bodyText = driver.findElement(By.tagName("body")).getText();
833            } catch (StaleElementReferenceException ignore) {
834                bodyText = htmlSource;
835            }
836        } catch (UnreachableBrowserException e) {
837            throw new WebDriverException(e);
838        } catch (WebDriverException e) {
839            throw e;
840        }
841
842        capturePageSnapshot();
843    }
844
845    public final void refresh() throws PageNotCurrentException {
846        Logging.logWebStep(null, "refresh", false);
847        try {
848            driver.navigate().refresh();
849        } catch (org.openqa.selenium.TimeoutException ex) {
850            Logging.log("got time out exception, ignore");
851        }
852
853        frameFlag = false;
854
855        try {
856            waitForPageToLoad();
857        } catch (Exception e) {
858            throw new RuntimeException(e);
859        }
860
861        assertNoPageErrors();
862    }
863
864    public final void resizeTo(int width, int height) {
865        new BaseWebUtil(driver).resizeWindow(width, height);
866    }
867
868    public final void selectFrame(By by) {
869        Logging.logWebStep(null, "select frame, locator={\"" + by.toString()
870                + "\"}", false);
871        driver.switchTo().frame(driver.findElement(by));
872        frameFlag = true;
873    }
874
875    public final void selectFrame(String locator) {
876        Logging.logWebStep(null, "select frame, locator={\"" + locator + "\"}",
877                false);
878        driver.switchTo().frame(locator);
879        frameFlag = true;
880    }
881
882    public final void selectWindow() throws PageNotCurrentException {
883        Logging.logWebStep(null, "select window, locator={\""
884                + getPopupWindowName() + "\"}", false);
885        // selectWindow(getPopupWindowName());
886        driver.switchTo().window(
887                (String) driver.getWindowHandles().toArray()[0]);
888        waitForSeconds(1);
```



```
891        assertCurrentPage(true);
892    }
893
894    public final void selectWindow(int index) throws PageNotCurrentException {
895        Logging.logWebStep(null, "select window, locator={\"" + index + "\"}",
896                false);
897        driver.switchTo().window(
898                (String) driver.getWindowHandles().toArray()[index]);
899    }
900
901    public final void selectNewWindow() throws PageNotCurrentException {
902        Logging.logWebStep(null, "select new window", false);
903        driver.switchTo().window(
904                (String) driver.getWindowHandles().toArray()[1]);
905        waitForSeconds(1);
906    }
907
908    protected void setHtmlSavedToPath(String htmlSavedToPath) {
909        this.htmlSavedToPath = htmlSavedToPath;
910    }
911
912    // protected void setHtmlSource(String htmlSource) {
913    // this.htmlSource = htmlSource;
914    // }
915
916    // protected void setLocation(String location) {
917    // this.location = location;
918    // }
919
920    protected void setTitle(String title) {
921        this.title = title;
922    }
923
924    protected void setUrl(String openUrl) {
925        this.url = openUrl;
926    }
927
928    public void switchToDefaultContent() {
929        try {
930            driver.switchTo().defaultContent();
931        } catch (UnhandledAlertException e) {
932        }
933    }
934
935    private final void waitForPageToLoad() throws Exception {
936        try {
937            new Wait() {
938                @Override
939                public boolean until() {
940                    try {
941                        driver.switchTo().defaultContent();
942                        return true;
943                    } catch (UnhandledAlertException ex) {
944                        ThreadHelper.waitForSeconds(2);
945                    } catch (WebDriverException e) {
946                    }
947                    return false;
948                }
949            }.wait(String.format("Timed out waiting for page to load"),
950                    WebUIDriver.getWebUIDriver().getWebSessionTimeout());
951        } catch (WaitTimedOutException ex) {
952
953        }
954        // populate page info
```

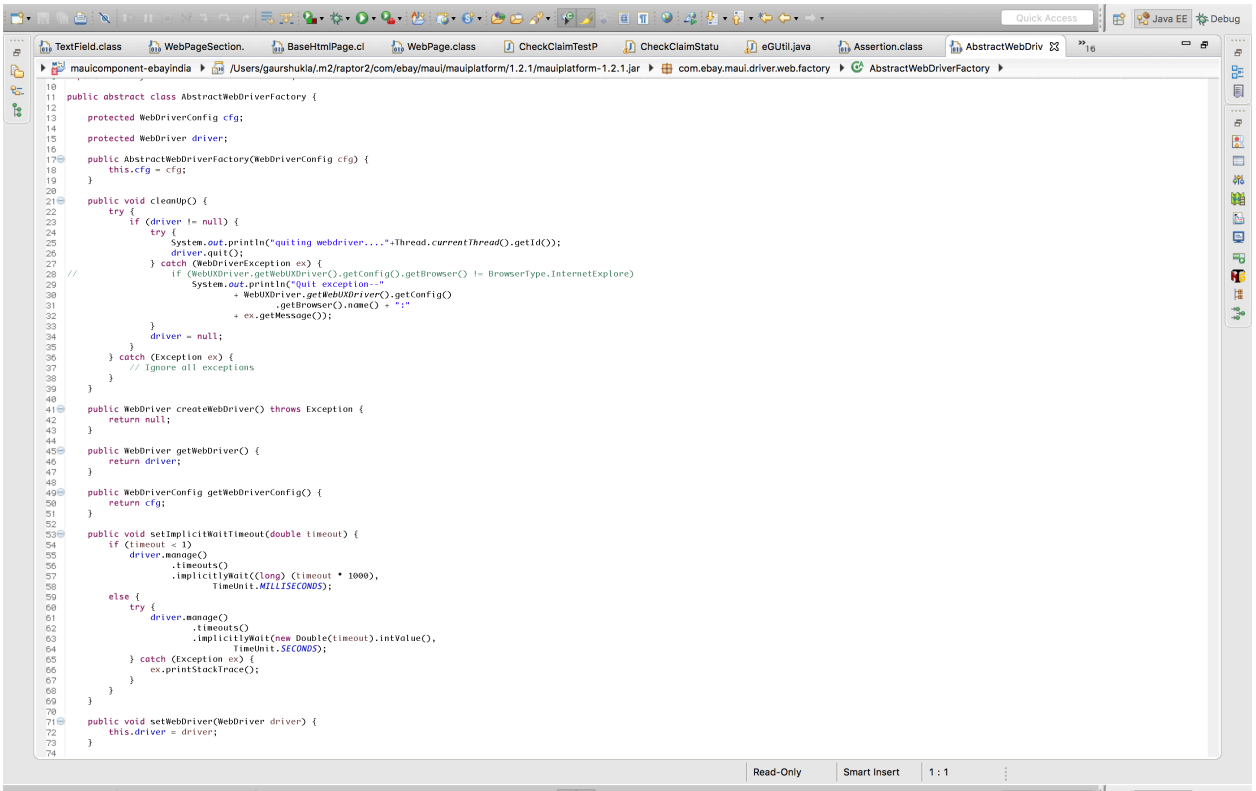


```
    protected void setUrl(String openUrl) {
        this.url = openUrl;
    }

    public void switchToDefaultContent() {
        try {
            driver.switchTo().defaultContent();
        } catch (UnhandledAlertException e) {
        }
    }

    private final void waitForPageToLoad() throws Exception {
        try {
            new Wait() {
                @Override
                public boolean until() {
                    try {
                        driver.switchTo().defaultContent();
                        return true;
                    } catch (UnhandledAlertException ex) {
                        ThreadHelper.waitForSeconds(2);
                    } catch (WebDriverException e) {
                    }
                    return false;
                }
            }.wait(String.format("Timed out waiting for page to load"),
                    WebUIDriver.getWebUIDriver().getWebSessionTimeout());
        } catch (WaitTimedOutException ex) {

        }
        // populate page info
        try {
            populateAndCapturePageSnapshot();
        } catch (Exception ex) {
            // ex.printStackTrace();
            throw ex;
        }
    }

    public WebElement getElement(By by, String elementName) {
        WebElement element = null;
        try {
            element = driver.findElement(by);
        } catch (ElementNotFoundException e) {
            Logging.error(elementName + " is not found with locator - "
                    + by.toString());
            throw e;
        }
        return element;
    }

    public String getElementUrl(By by, String name) {
        return getElement(by, name).getAttribute("href");
    }

    public String getElementText(By by, String name) {
        return getElement(by, name).getText();
    }

    public String getElementSrc(By by, String name) {
        return getElement(by, name).getAttribute("src");
    }
}
```

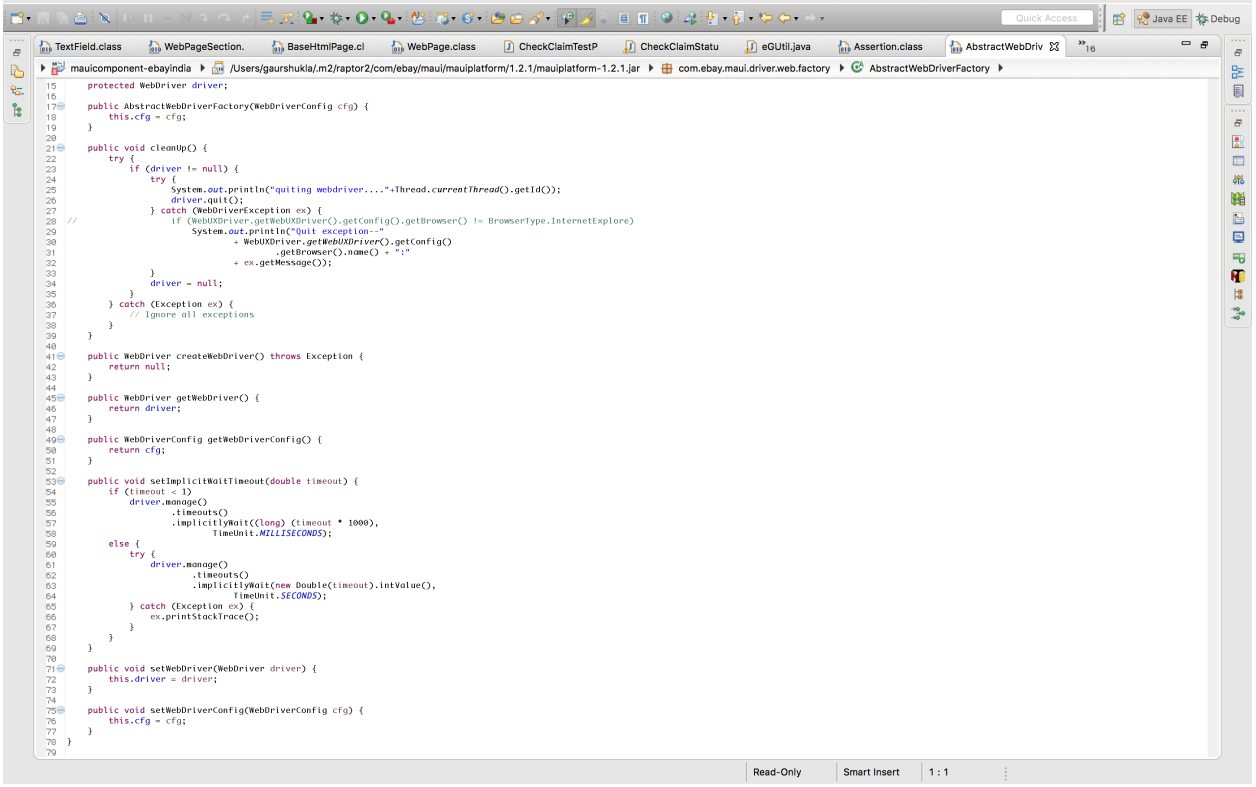


```
import com.ebay.maui.controller.Logging;
import com.ebay.maui.driver.web.ScreenShot;
import com.ebay.maui.driver.web.ScreenshotUtil;
import com.ebay.maui.exception.MauiException;

public abstract class WebPageSection extends BaseHtmlPage {

    private String name = null;
    private String locator = null;
    protected WebElement element = null;
    private By by = null;

    public WebPageSection(String name) {
        super();
        this.name = name;
    }

    public WebPageSection(String name, By by) {
        super();
        this.name = name;
        this.by = by;
    }

    public WebPageSection(String name, String locator) throws MauiException {
        super();
        if (locator.startsWith("xpath=")) {
            locator = locator.substring(6);
        } else if (!locator.startsWith("/") && !locator.startsWith("(")
            locator = "//*[@id='" + locator + "' or @name='" + locator + "']";

        this.name = name;
        this.locator = locator;
        this.by = By.xpath(locator);
    }

    /**
     * Captures page snapshot.
     */
    public void capturePageSnapshot() {

        ScreenShot screenShot = new ScreenshotUtil(driver).captureWebPageSnapshot();
        String title = screenShot.getTitle();
        String url = screenShot.getLocation();

        Logging.logWebOutput(url, title + " (" + Logging.buildScreenshotLog(screenShot) + ")", false);
    }

    public String getLocator() {
        return locator;
    }

    public String getName() {
        return name;
    }

    public By getBy() {
        return by;
    }

    public boolean isPageSectionPresent() {
        return isElementPresent(by);
    }
}
```

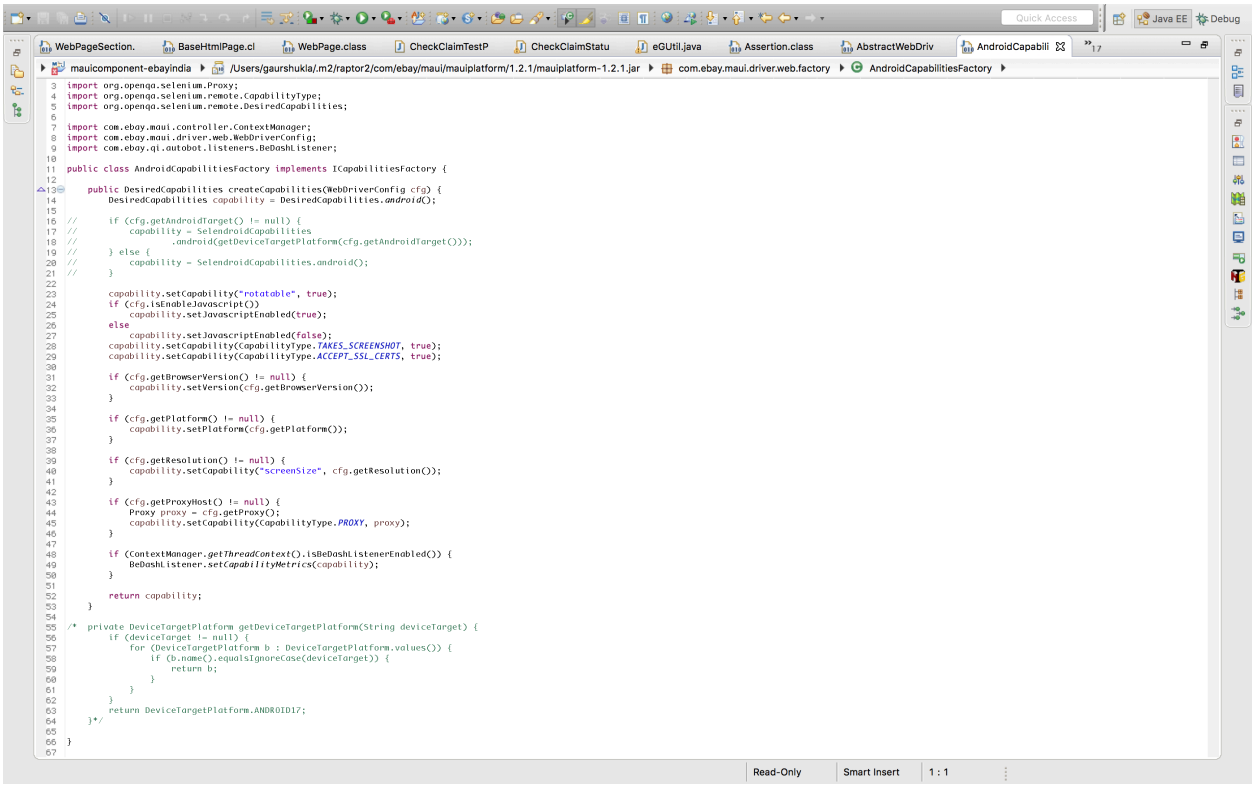
Web.factory



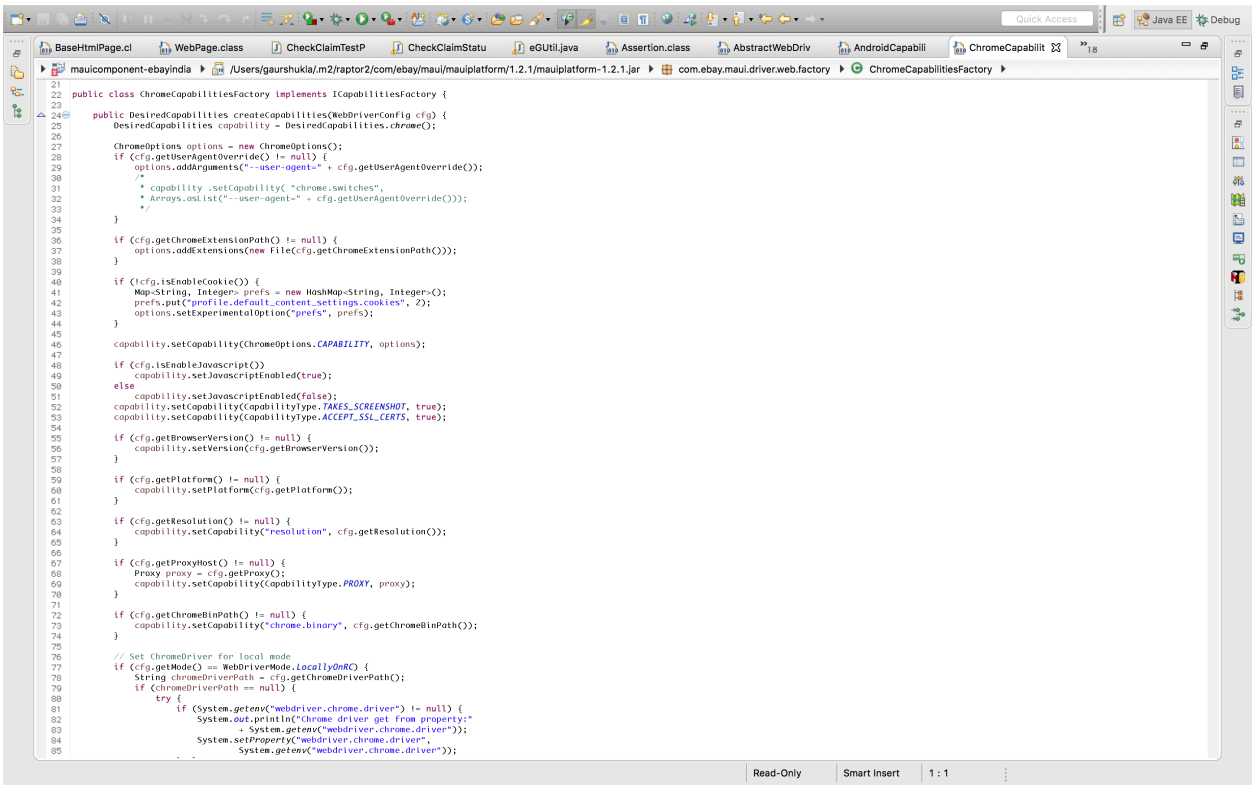
```
public abstract class AbstractWebDriverFactory {

    protected WebDriverConfig cfg;

    protected WebDriver driver;

    public AbstractWebDriverFactory(WebDriverConfig cfg) {
        this.cfg = cfg;
    }

    public void cleanUp() {
        try {
            if (driver != null) {
                try {
                    System.out.println("quiting webdriver...."+Thread.currentThread().getId());
                    driver.quit();
                } catch (WebDriverException ex) {
//                  if (WebUXDriver.getWebUXDriver().getConfig().getBrowser() != BrowserType.InternetExplore)
                        System.out.println("Quit exception--"
                                + WebUXDriver.getWebUXDriver().getConfig()
                                        .getBrowser().name() + ":"
                                + ex.getMessage());
                }
                driver = null;
            }
        } catch (Exception ex) {
            // Ignore all exceptions
        }
    }

    public WebDriver createWebDriver() throws Exception {
        return null;
    }

    public WebDriver getWebDriver() {
        return driver;
    }

    public WebDriverConfig getWebDriverConfig() {
        return cfg;
    }

    public void setImplicitWaitTimeout(double timeout) {
        if (timeout < 1)
            driver.manage()
                    .timeouts()
                    .implicitlyWait((long) (timeout * 1000),
                            TimeUnit.MILLISECONDS);
        else {
            try {
                driver.manage()
                        .timeouts()
                        .implicitlyWait(new Double(timeout).intValue(),
                                TimeUnit.SECONDS);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
    }

    public void setWebDriver(WebDriver driver) {
        this.driver = driver;
    }
```



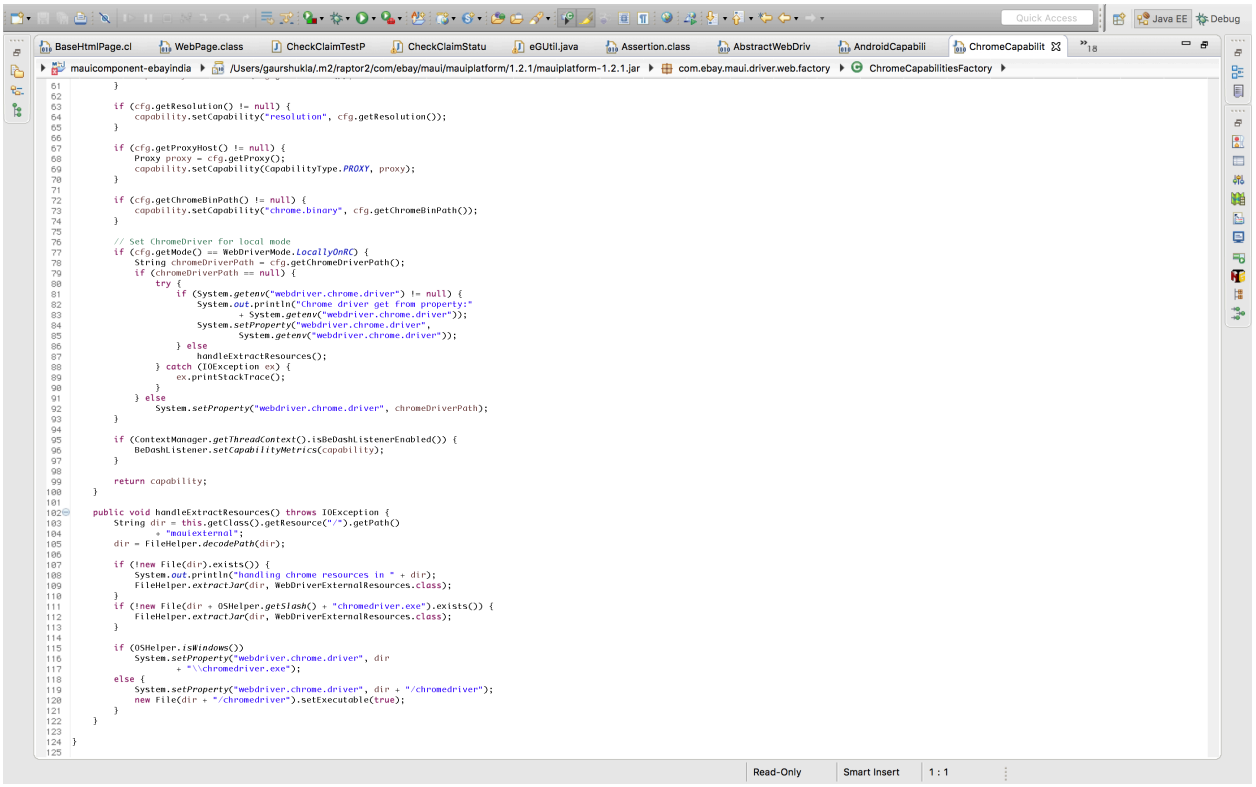
```
    protected WebDriver driver;

    public AbstractWebDriverFactory(WebDriverConfig cfg) {
        this.cfg = cfg;
    }

    public void cleanUp() {
        try {
            if (driver != null) {
                try {
                    System.out.println("quiting webdriver...."+Thread.currentThread().getId());
                    driver.quit();
                } catch (WebDriverException ex) {
//                  if (WebUXDriver.getWebUXDriver().getConfig().getBrowser() != BrowserType.InternetExplore)
                        System.out.println("Quit exception--"
                                + WebUXDriver.getWebUXDriver().getConfig()
                                        .getBrowser().name() + ":"
                                + ex.getMessage());
                }
                driver = null;
            }
        } catch (Exception ex) {
            // Ignore all exceptions
        }
    }

    public WebDriver createWebDriver() throws Exception {
        return null;
    }

    public WebDriver getWebDriver() {
        return driver;
    }

    public WebDriverConfig getWebDriverConfig() {
        return cfg;
    }

    public void setImplicitWaitTimeout(double timeout) {
        if (timeout < 1)
            driver.manage()
                    .timeouts()
                    .implicitlyWait((long) (timeout * 1000),
                            TimeUnit.MILLISECONDS);
        else {
            try {
                driver.manage()
                        .timeouts()
                        .implicitlyWait(new Double(timeout).intValue(),
                                TimeUnit.SECONDS);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
    }

    public void setWebDriver(WebDriver driver) {
        this.driver = driver;
    }

    public void setWebDriverConfig(WebDriverConfig cfg) {
        this.cfg = cfg;
    }
}
```



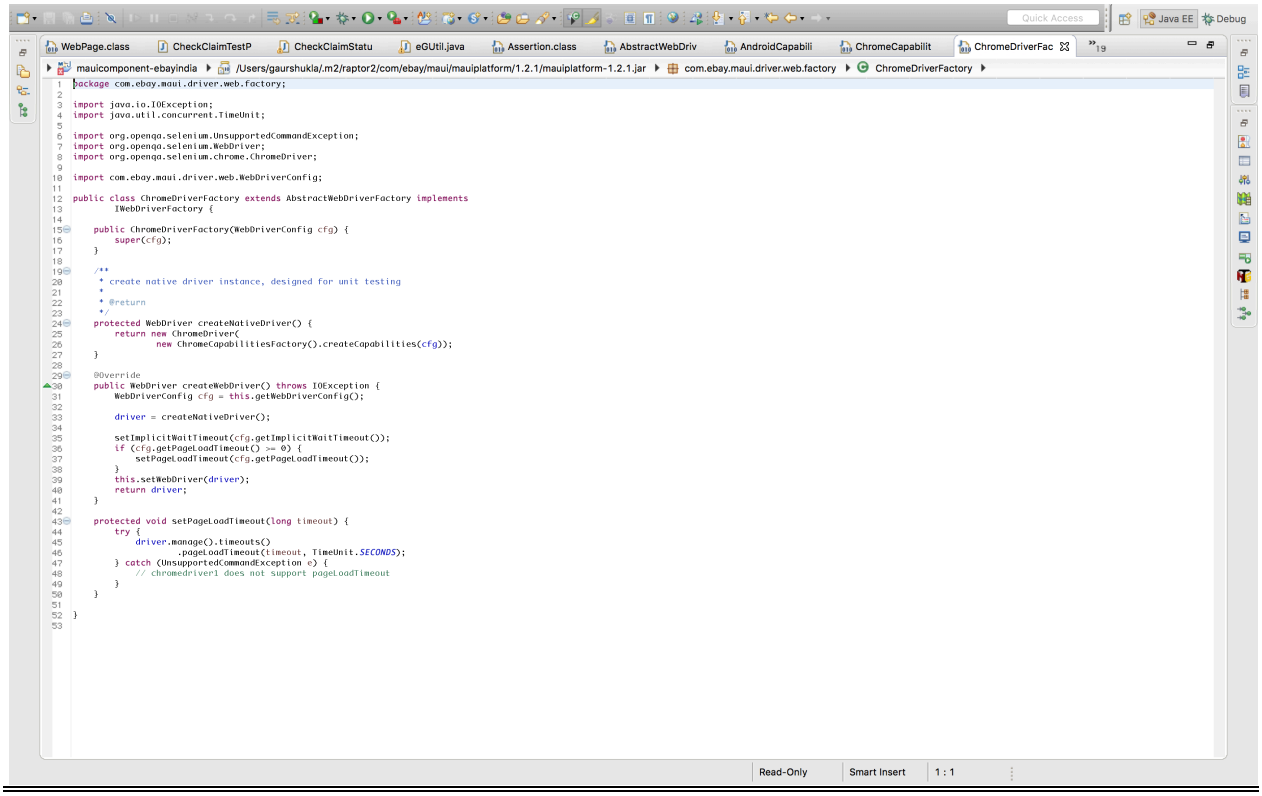
```
import org.openqa.selenium.Proxy;
import org.openqa.selenium.remote.CapabilityType;
import org.openqa.selenium.remote.DesiredCapabilities;

import com.ebay.maui.controller.ContextManager;
import com.ebay.maui.driver.web.WebDriverConfig;
import com.ebay.qi.autobot.listeners.BeDashListener;

public class AndroidCapabilitiesFactory implements ICapabilitiesFactory {

    public DesiredCapabilities createCapabilities(WebDriverConfig cfg) {
        DesiredCapabilities capability = DesiredCapabilities.android();

//      if (cfg.getAndroidTarget() != null) {
//          capability = SelendroidCapabilities
//                  .android(getDeviceTargetPlatform(cfg.getAndroidTarget()));
//      } else {
//          capability = SelendroidCapabilities.android();
//      }

        capability.setCapability("rotatable", true);
        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.getBrowserVersion() != null) {
            capability.setVersion(cfg.getBrowserVersion());
        }

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

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

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

        if (ContextManager.getThreadContext().isBeDashListenerEnabled()) {
            BeDashListener.setCapabilityMetrics(capability);
        }

        return capability;
    }

/*  private DeviceTargetPlatform getDeviceTargetPlatform(String deviceTarget) {
        if (deviceTarget != null) {
            for (DeviceTargetPlatform b : DeviceTargetPlatform.values()) {
                if (b.name().equalsIgnoreCase(deviceTarget)) {
                    return b;
                }
            }
        }
        return DeviceTargetPlatform.ANDROID17;
    }*/

}
```



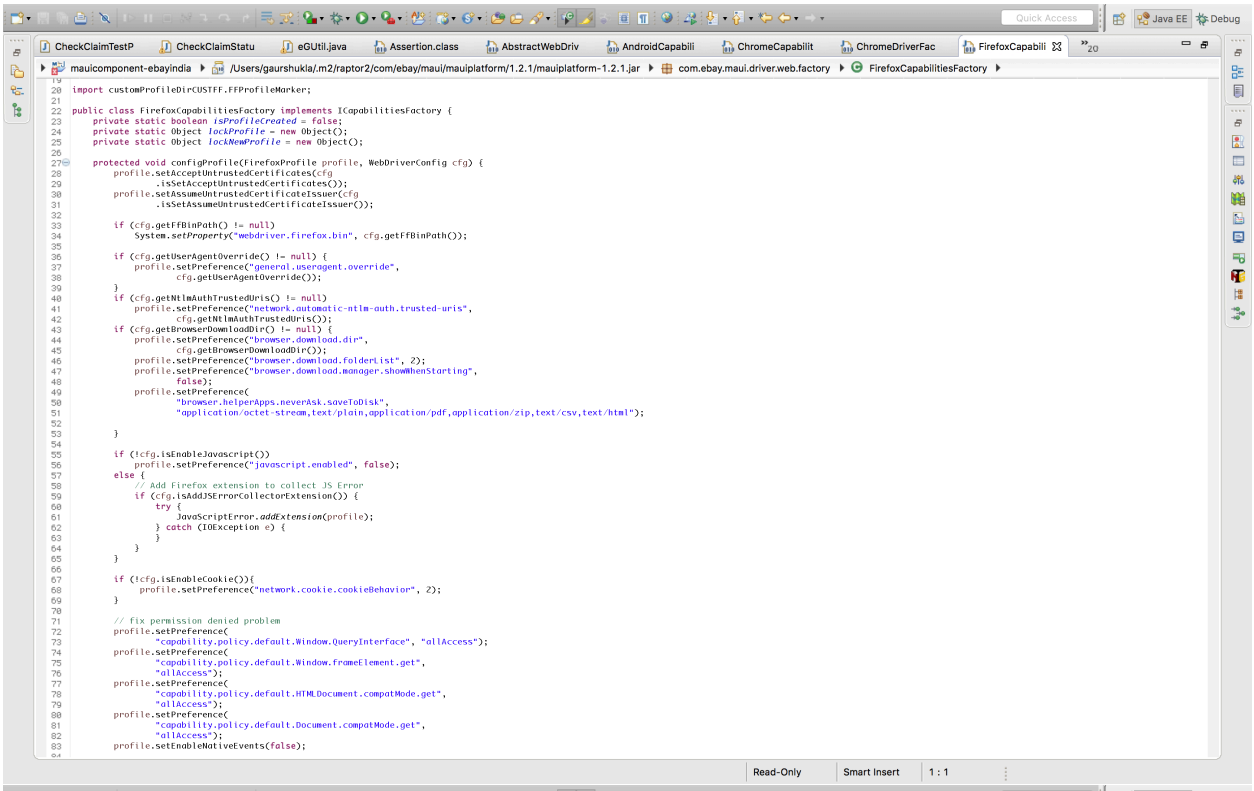
```
public class ChromeCapabilitiesFactory implements ICapabilitiesFactory {

    public DesiredCapabilities createCapabilities(WebDriverConfig cfg) {
        DesiredCapabilities capability = DesiredCapabilities.chrome();

        ChromeOptions options = new ChromeOptions();
        if (cfg.getUserAgentOverride() != null) {
            options.addArguments("--user-agent=" + cfg.getUserAgentOverride());
            /*
             * capability .setCapability( "chrome.switches",
             * Arrays.asList("--user-agent=" + cfg.getUserAgentOverride()));
             */
        }

        if (cfg.getChromeExtensionPath() != null) {
            options.addExtensions(new File(cfg.getChromeExtensionPath()));
        }

        if (!cfg.isEnableCookie()) {
            Map<String, Integer> prefs = new HashMap<String, Integer>();
            prefs.put("profile.default_content_settings.cookies", 2);
            options.setExperimentalOption("prefs", prefs);
        }

        capability.setCapability(ChromeOptions.CAPABILITY, options);

        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.getBrowserVersion() != null) {
            capability.setVersion(cfg.getBrowserVersion());
        }

        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 (cfg.getChromeBinPath() != null) {
            capability.setCapability("chrome.binary", cfg.getChromeBinPath());
        }

        // Set ChromeDriver for local mode
        if (cfg.getMode() == WebDriverMode.LocallyOnRC) {
            String chromeDriverPath = cfg.getChromeDriverPath();
            if (chromeDriverPath == null) {
                try {
                    if (System.getenv("webdriver.chrome.driver") != null) {
                        System.out.println("Chrome driver get from property:"
                                + System.getenv("webdriver.chrome.driver"));
                        System.setProperty("webdriver.chrome.driver",
                                System.getenv("webdriver.chrome.driver"));
```



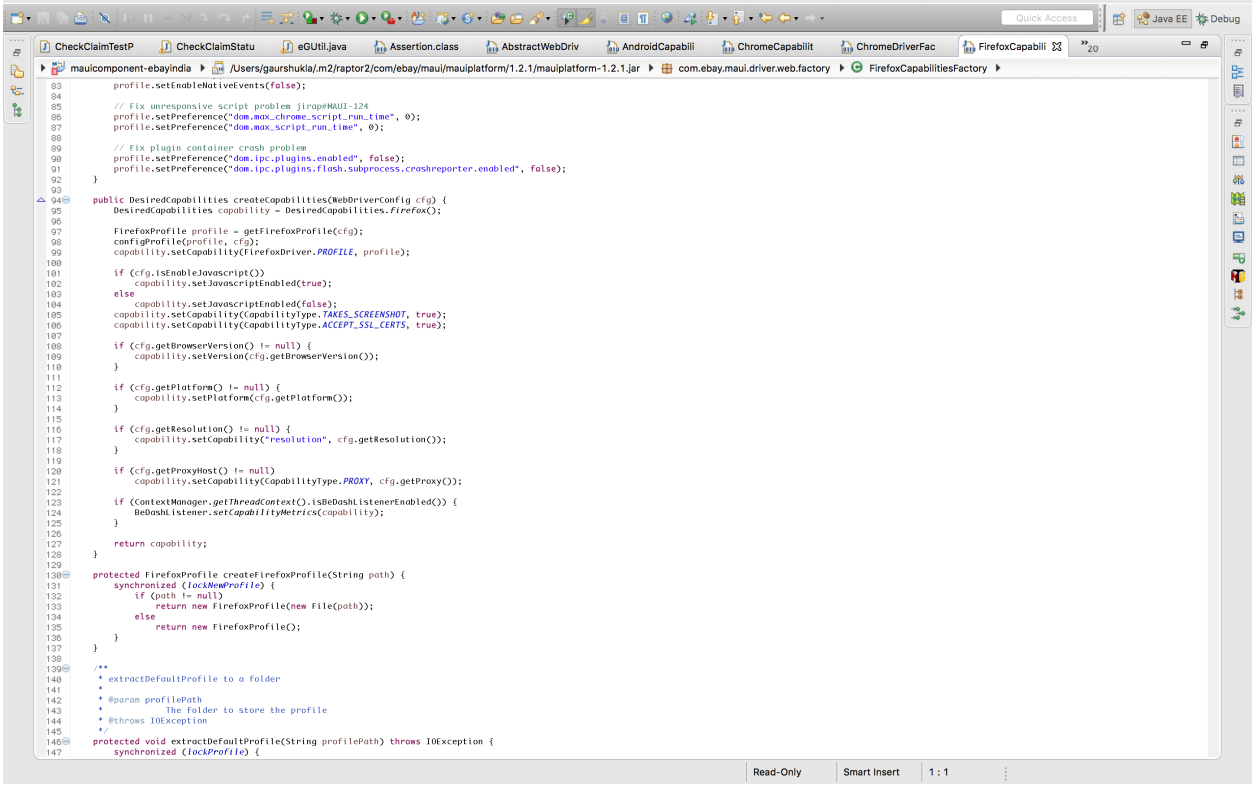
```
        }

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

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

        if (cfg.getChromeBinPath() != null) {
            capability.setCapability("chrome.binary", cfg.getChromeBinPath());
        }

        // Set ChromeDriver for local mode
        if (cfg.getMode() == WebDriverMode.LocallyOnRC) {
            String chromeDriverPath = cfg.getChromeDriverPath();
            if (chromeDriverPath == null) {
                try {
                    if (System.getenv("webdriver.chrome.driver") != null) {
                        System.out.println("Chrome driver get from property:"
                                + System.getenv("webdriver.chrome.driver"));
                        System.setProperty("webdriver.chrome.driver",
                                System.getenv("webdriver.chrome.driver"));
                    } else
                        handleExtractResources();
                } catch (IOException ex) {
                    ex.printStackTrace();
                }
            } else
                System.setProperty("webdriver.chrome.driver", chromeDriverPath);
        }

        if (ContextManager.getThreadContext().isBeDashlistenerEnabled()) {
            BeDashListener.setCapabilityMetrics(capability);
        }

        return capability;
    }

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

        if (!new File(dir).exists()) {
            System.out.println("handling chrome resources in " + dir);
            FileHelper.extractJar(dir, WebDriverExternalResources.class);
        }
        if (!new File(dir + OSHelper.getSlash() + "chromedriver.exe").exists()) {
            FileHelper.extractJar(dir, WebDriverExternalResources.class);
        }

        if (OSHelper.isWindows())
            System.setProperty("webdriver.chrome.driver", dir
                    + "\\chromedriver.exe");
        else {
            System.setProperty("webdriver.chrome.driver", dir + "/chromedriver");
            new File(dir + "/chromedriver").setExecutable(true);
        }
    }

}
```



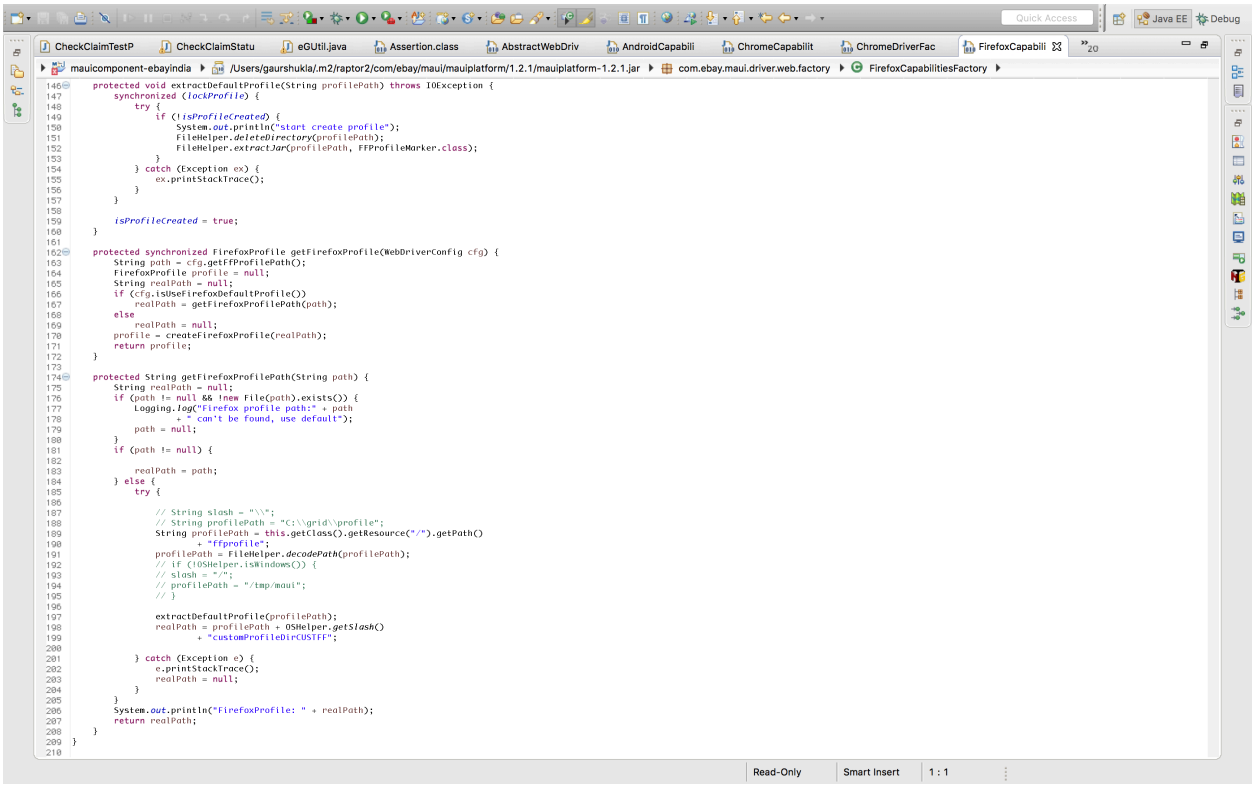
```
package com.ebay.maui.driver.web.factory;

import java.io.IOException;
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.UnsupportedCommandException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

import com.ebay.maui.driver.web.WebDriverConfig;

public class ChromeDriverFactory extends AbstractWebDriverFactory implements
        IWebDriverFactory {

    public ChromeDriverFactory(WebDriverConfig cfg) {
        super(cfg);
    }

    /**
     * create native driver instance, designed for unit testing
     *
     * @return
     */
    protected WebDriver createNativeDriver() {
        return new ChromeDriver(
                new ChromeCapabilitiesFactory().createCapabilities(cfg));
    }

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

        driver = createNativeDriver();

        setImplicitWaitTimeout(cfg.getImplicitWaitTimeout());
        if (cfg.getPageLoadTimeout() >= 0) {
            setPageLoadTimeout(cfg.getPageLoadTimeout());
        }
        this.setWebDriver(driver);
        return driver;
    }

    protected void setPageLoadTimeout(long timeout) {
        try {
            driver.manage().timeouts()
                    .pageLoadTimeout(timeout, TimeUnit.SECONDS);
        } catch (UnsupportedCommandException e) {
            // chromedriver1 does not support pageLoadTimeout
        }
    }

}
```



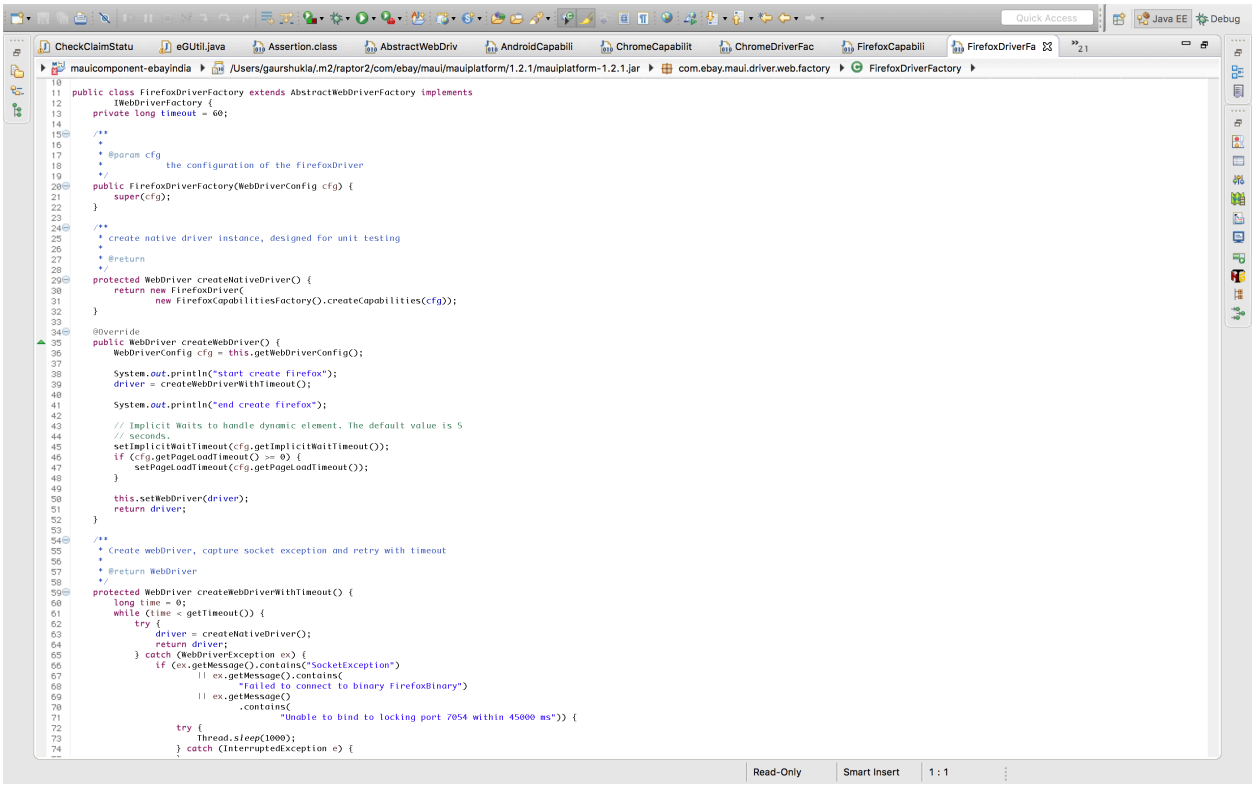
```
import customProfileDirCUSTFF.FFProfileMarker;

public class FirefoxCapabilitiesFactory implements ICapabilitiesFactory {
    private static boolean isProfileCreated = false;
    private static Object lockProfile = new Object();
    private static Object lockNewProfile = new Object();

    protected void configProfile(FirefoxProfile profile, WebDriverConfig cfg) {
        profile.setAcceptUntrustedCertificates(cfg
                .isSetAcceptUntrustedCertificates());
        profile.setAssumeUntrustedCertificateIssuer(cfg
                .isSetAssumeUntrustedCertificateIssuer());

        if (cfg.getFfBinPath() != null)
            System.setProperty("webdriver.firefox.bin", cfg.getFfBinPath());

        if (cfg.getUserAgentOverride() != null) {
            profile.setPreference("general.useragent.override",
                    cfg.getUserAgentOverride());
        }
        if (cfg.getNtlmAuthTrustedUris() != null)
            profile.setPreference("network.automatic-ntlm-auth.trusted-uris",
                    cfg.getNtlmAuthTrustedUris());
        if (cfg.getBrowserDownloadDir() != null) {
            profile.setPreference("browser.download.dir",
                    cfg.getBrowserDownloadDir());
            profile.setPreference("browser.download.folderList", 2);
            profile.setPreference("browser.download.manager.showWhenStarting",
                    false);
            profile.setPreference(
                    "browser.helperApps.neverAsk.saveToDisk",
                    "application/octet-stream,text/plain,application/pdf,application/zip,text/csv,text/html");

        }

        if (!cfg.isEnableJavascript())
            profile.setPreference("javascript.enabled", false);
        else {
            // Add Firefox extension to collect JS Error
            if (cfg.isAddJSErrorCollectorExtension()) {
                try {
                    JavaScriptError.addExtension(profile);
                } catch (IOException e) {
                }
            }
        }

        if (!cfg.isEnableCookie()){
            profile.setPreference("network.cookie.cookieBehavior", 2);
        }

        // fix permission denied problem
        profile.setPreference(
                "capability.policy.default.Window.QueryInterface", "allAccess");
        profile.setPreference(
                "capability.policy.default.Window.frameElement.get",
                "allAccess");
        profile.setPreference(
                "capability.policy.default.HTMLDocument.compatMode.get",
                "allAccess");
        profile.setPreference(
                "capability.policy.default.Document.compatMode.get",
                "allAccess");
        profile.setEnableNativeEvents(false);
```



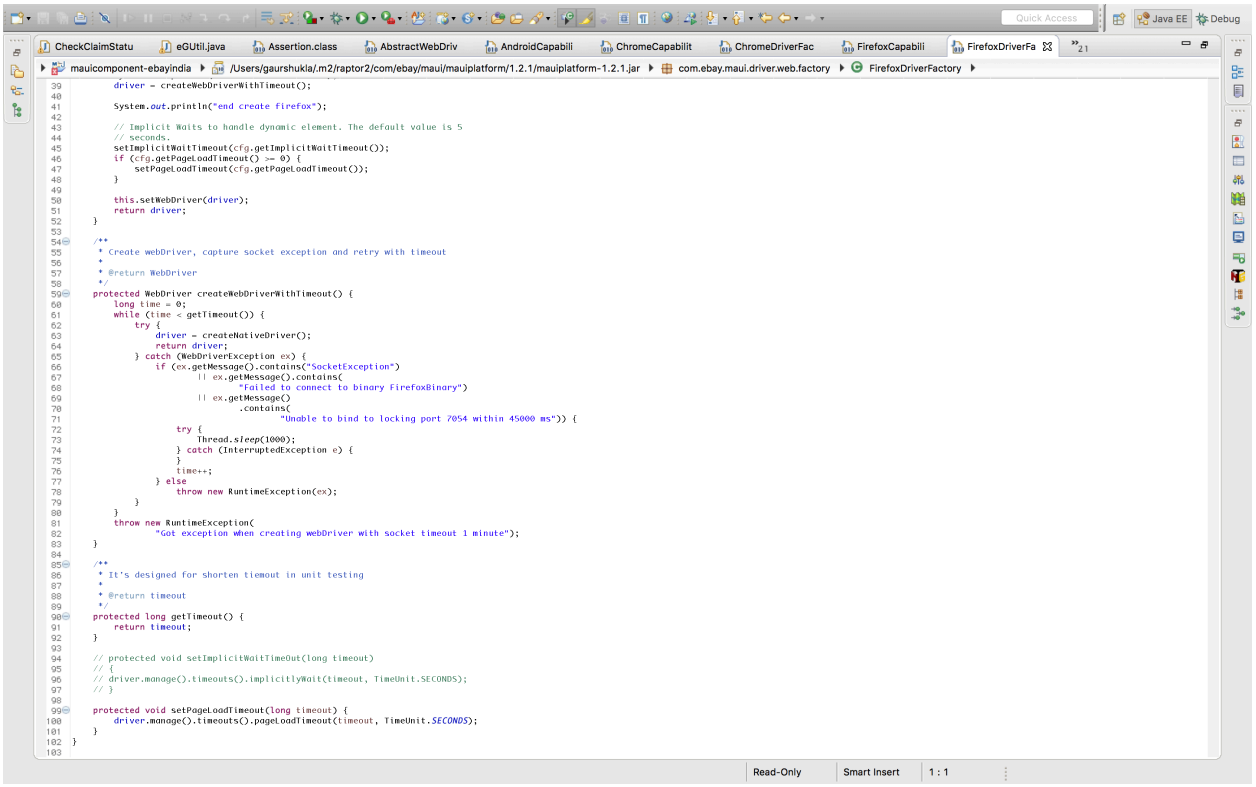
```
        profile.setEnableNativeEvents(false);

        // Fix unresponsive script problem jira#MAUI-124
        profile.setPreference("dom.max_chrome_script_run_time", 0);
        profile.setPreference("dom.max_script_run_time", 0);

        // Fix plugin container crash problem
        profile.setPreference("dom.ipc.plugins.enabled", false);
        profile.setPreference("dom.ipc.plugins.flash.subprocess.crashreporter.enabled", false);
    }

    public DesiredCapabilities createCapabilities(WebDriverConfig cfg) {
        DesiredCapabilities capability = DesiredCapabilities.firefox();

        FirefoxProfile profile = getFirefoxProfile(cfg);
        configProfile(profile, cfg);
        capability.setCapability(FirefoxDriver.PROFILE, profile);

        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.getBrowserVersion() != null) {
            capability.setVersion(cfg.getBrowserVersion());
        }

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

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

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

        if (ContextManager.getThreadContext().isBeDashListenerEnabled()) {
            BeDashListener.setCapabilityMetrics(capability);
        }

        return capability;
    }

    protected FirefoxProfile createFirefoxProfile(String path) {
        synchronized (lockNewProfile) {
            if (path != null)
                return new FirefoxProfile(new File(path));
            else
                return new FirefoxProfile();
        }
    }

    /**
     * extractDefaultProfile to a folder
     *
     * @param profilePath
     *            The folder to store the profile
     * @throws IOException
     */
    protected void extractDefaultProfile(String profilePath) throws IOException {
        synchronized (lockProfile) {
```



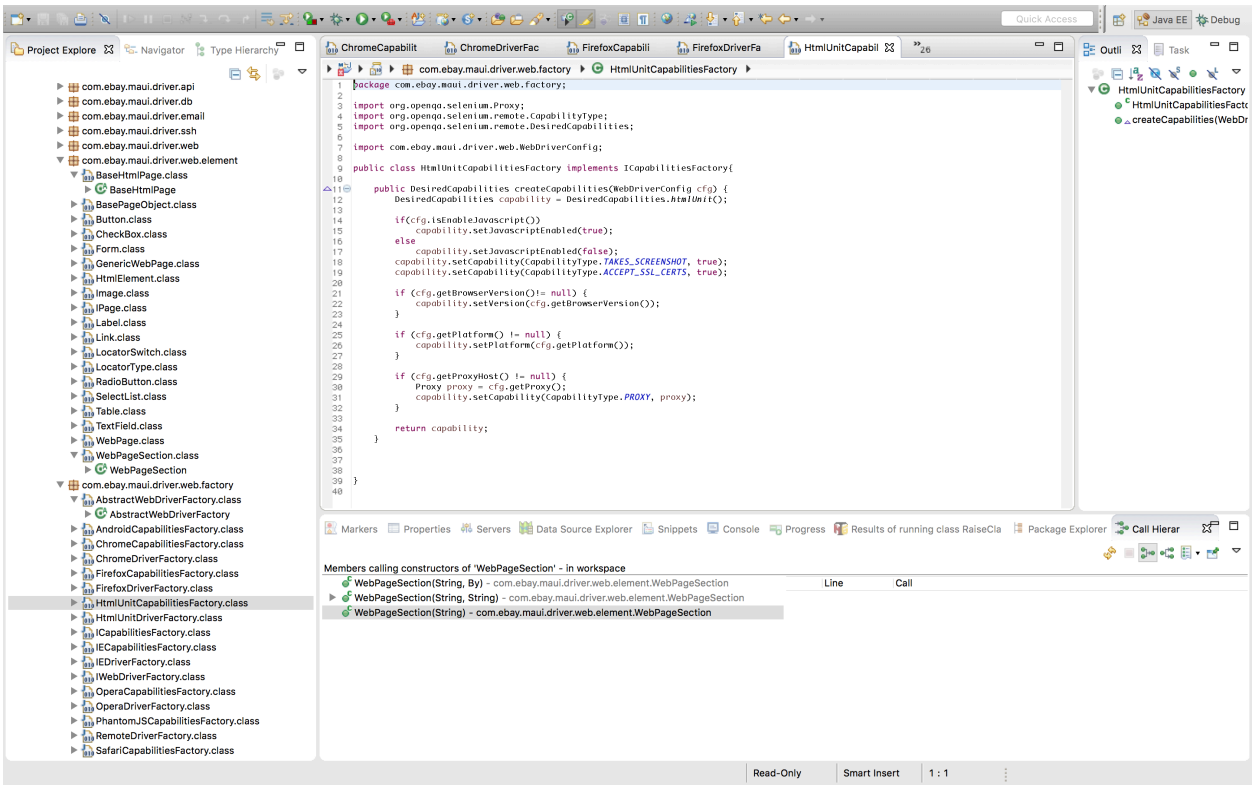
```
protected void extractDefaultProfile(String profilePath) throws IOException {
    synchronized (lockProfile) {
        try {
            if (!isProfileCreated) {
                System.out.println("start create profile");
                FileHelper.deleteDirectory(profilePath);
                FileHelper.extractJar(profilePath, FFProfileMarker.class);
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    isProfileCreated = true;
}

protected synchronized FirefoxProfile getFirefoxProfile(WebDriverConfig cfg) {
    String path = cfg.getFFProfilePath();
    FirefoxProfile profile = null;
    String realPath = null;
    if (cfg.isUseFirefoxDefaultProfile())
        realPath = getFirefoxProfilePath(path);
    else
        realPath = null;
    profile = createFirefoxProfile(realPath);
    return profile;
}

protected String getFirefoxProfilePath(String path) {
    String realPath = null;
    if (path != null && !new File(path).exists()) {
        Logging.log("Firefox profile path:" + path
                + " can't be found, use default");
        path = null;
    }
    if (path != null) {

        realPath = path;
    } else {
        try {

            // String slash = "\\";
            // String profilePath = "C:\\grid\\profile";
            String profilePath = this.getClass().getResource("/").getPath()
                    + "ffprofile";
            profilePath = FileHelper.decodePath(profilePath);
            // if (!OSHelper.isWindows()) {
            // slash = "/";
            // profilePath = "/tmp/maui";
            // }

            extractDefaultProfile(profilePath);
            realPath = profilePath + OSHelper.getSlash()
                    + "customProfileDirCUSTFF";

        } catch (Exception e) {
            e.printStackTrace();
            realPath = null;
        }
    }
    System.out.println("FirefoxProfile: " + realPath);
    return realPath;
}
}
```



```
public class FirefoxDriverFactory extends AbstractWebDriverFactory implements
        IWebDriverFactory {
    private long timeout = 60;

    /**
     *
     * @param cfg
     *            the configuration of the FirefoxDriver
     */
    public FirefoxDriverFactory(WebDriverConfig cfg) {
        super(cfg);
    }

    /**
     * create native driver instance, designed for unit testing
     *
     * @return
     */
    protected WebDriver createNativeDriver() {
        return new FirefoxDriver(
                new FirefoxCapabilitiesFactory().createCapabilities(cfg));
    }

    @Override
    public WebDriver createWebDriver() {
        WebDriverConfig cfg = this.getWebDriverConfig();

        System.out.println("start create firefox");
        driver = createWebDriverWithTimeout();

        System.out.println("end create firefox");

        // Implicit Waits to handle dynamic element. The default value is 5
        // seconds.
        setImplicitWaitTimeout(cfg.getImplicitWaitTimeout());
        if (cfg.getPageLoadTimeout() >= 0) {
            setPageLoadTimeout(cfg.getPageLoadTimeout());
        }

        this.setWebDriver(driver);
        return driver;
    }

    /**
     * Create webDriver, capture socket exception and retry with timeout
     *
     * @return WebDriver
     */
    protected WebDriver createWebDriverWithTimeout() {
        long time = 0;
        while (time < getTimeout()) {
            try {
                driver = createNativeDriver();
                return driver;
            } catch (WebDriverException ex) {
                if (ex.getMessage().contains("SocketException")
                        || ex.getMessage().contains(
                                "Failed to connect to binary FirefoxBinary")
                        || ex.getMessage()
                                .contains(
                                        "Unable to bind to locking port 7054 within 45000 ms")) {
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
```

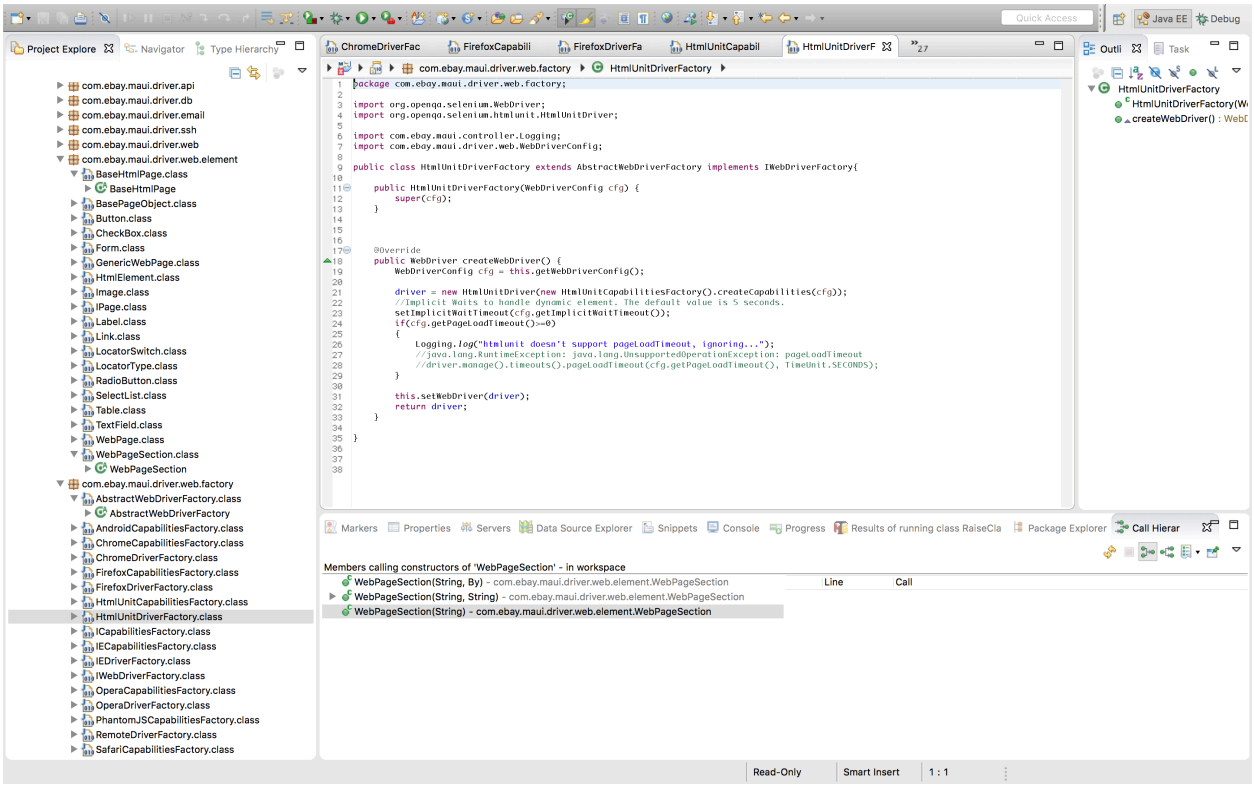


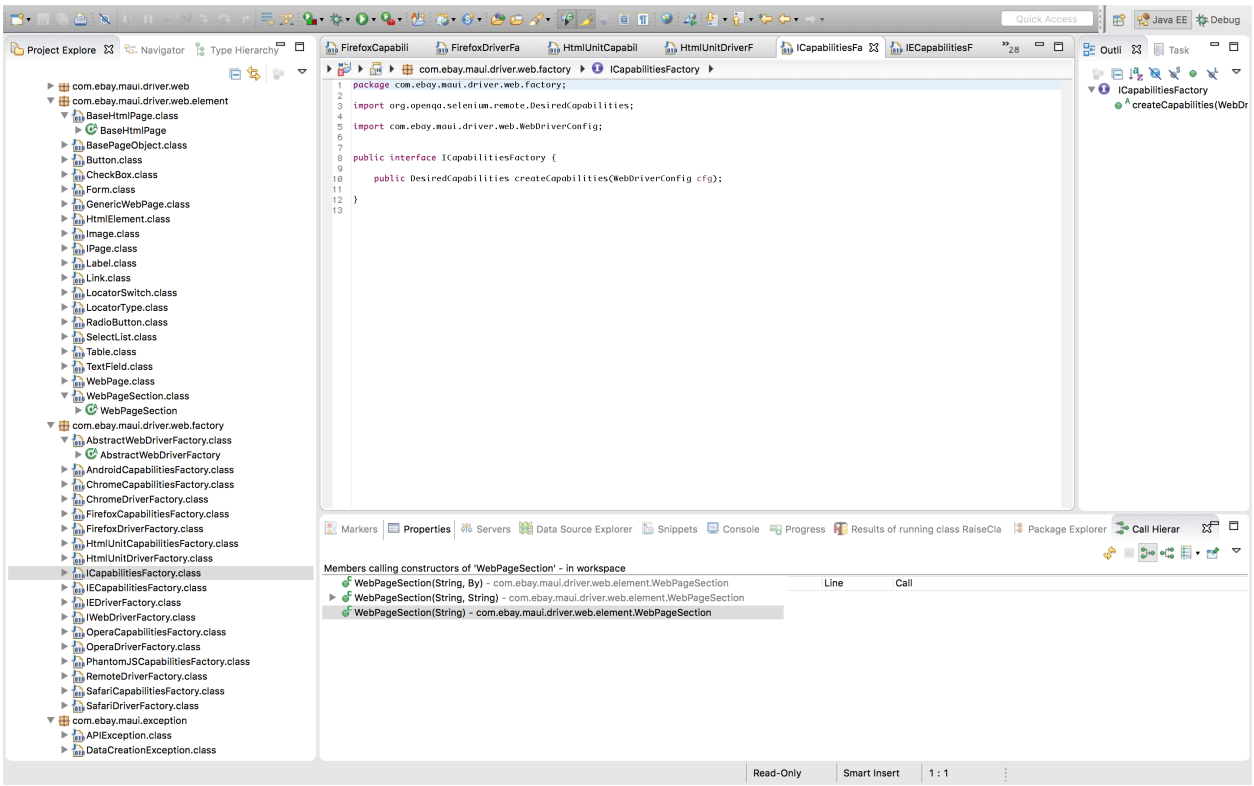
```
        driver = createWebDriverWithTimeout();

        System.out.println("end create firefox");

        // Implicit Waits to handle dynamic element. The default value is 5
        // seconds.
        setImplicitWaitTimeout(cfg.getImplicitWaitTimeout());
        if (cfg.getPageLoadTimeout() >= 0) {
            setPageLoadTimeout(cfg.getPageLoadTimeout());
        }

        this.setWebDriver(driver);
        return driver;
    }

    /**
     * Create WebDriver, capture socket exception and retry with timeout
     *
     * @return WebDriver
     */
    protected WebDriver createWebDriverWithTimeout() {
        long time = 0;
        while (time < getTimeout()) {
            try {
                driver = createNativeDriver();
                return driver;
            } catch (WebDriverException ex) {
                if (ex.getMessage().contains("SocketException")
                        || ex.getMessage().contains(
                                "Failed to connect to binary FirefoxBinary")
                        || ex.getMessage()
                                .contains(
                                        "Unable to bind to locking port 7054 within 45000 ms")) {

                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                    }
                    time++;
                } else
                    throw new RuntimeException(ex);
            }
        }
        throw new RuntimeException(
                "Got exception when creating webDriver with socket timeout 1 minute");
    }

    /**
     * It's designed for shorten tiemout in unit testing
     *
     * @return timeout
     */
    protected long getTimeout() {
        return timeout;
    }

    // protected void setImplicitWaitTimeOut(long timeout)
    // {
    // driver.manage().timeouts().implicitlyWait(timeout, TimeUnit.SECONDS);
    // }

    protected void setPageLoadTimeout(long timeout) {
        driver.manage().timeouts().pageLoadTimeout(timeout, TimeUnit.SECONDS);
    }
}
```

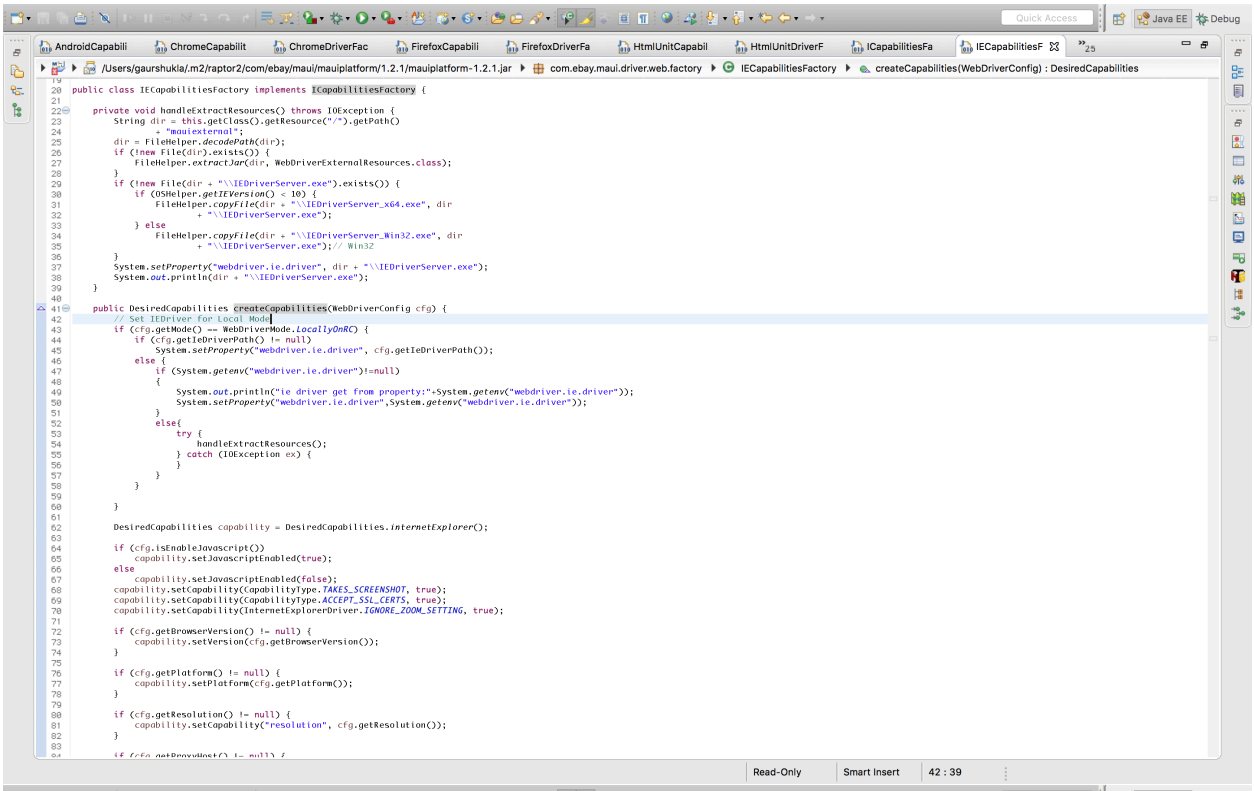


```
package com.ebay.maui.driver.web.factory;

import org.openqa.selenium.Proxy;
import org.openqa.selenium.remote.CapabilityType;
import org.openqa.selenium.remote.DesiredCapabilities;

import com.ebay.maui.driver.web.WebDriverConfig;

public class HtmlUnitCapabilitiesFactory implements ICapabilitiesFactory{

    public DesiredCapabilities createCapabilities(WebDriverConfig cfg) {
        DesiredCapabilities capability = DesiredCapabilities.htmlUnit();

        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.getBrowserVersion()!= null) {
            capability.setVersion(cfg.getBrowserVersion());
        }

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

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

        return capability;
    }


}
```

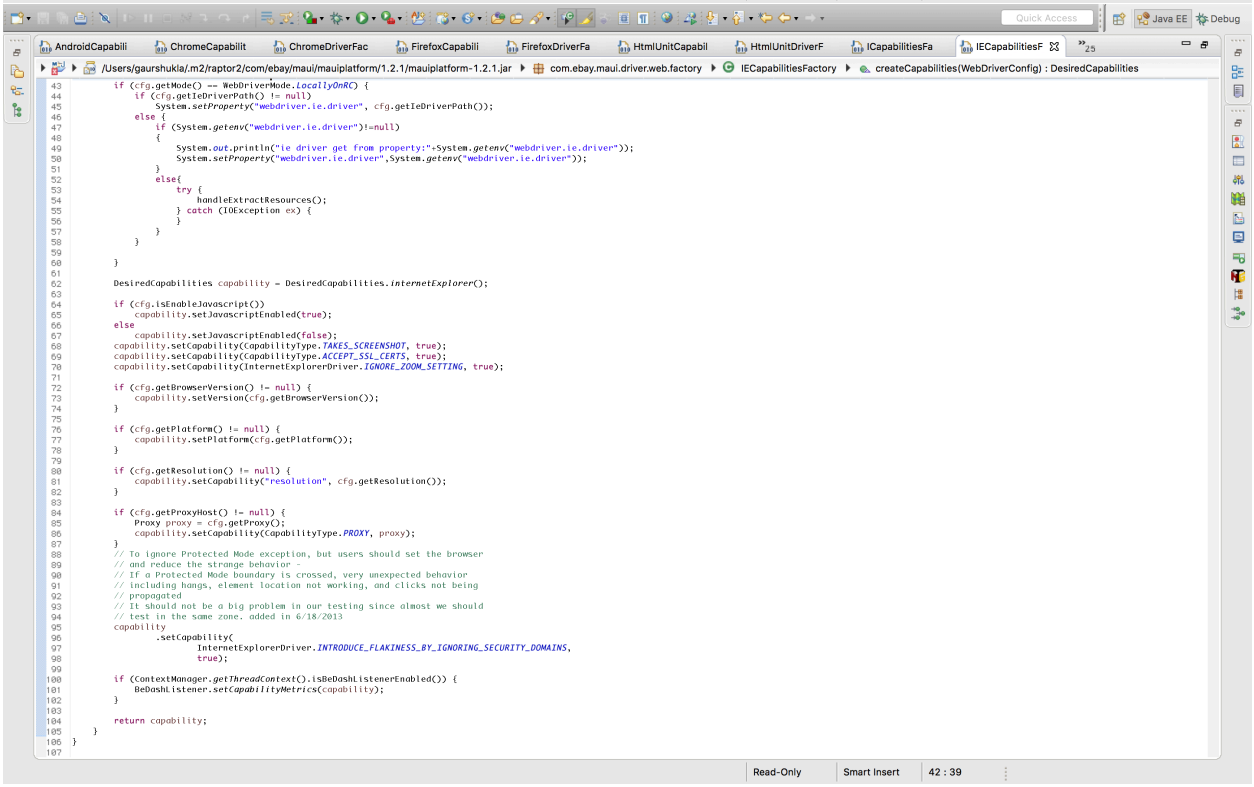
```
Members calling constructors of 'WebPageSection' - in workspace
WebPageSection(String, By) - com.ebay.maui.driver.web.element.WebPageSection
WebPageSection(String, String) - com.ebay.maui.driver.web.element.WebPageSection
WebPageSection(String) - com.ebay.maui.driver.web.element.WebPageSection
```


Project Explore
Navigator
Type Hierarchy
com.ebay.maui.driver.api
com.ebay.maui.driver.db
com.ebay.maui.driver.email
com.ebay.maui.driver.ssh
com.ebay.maui.driver.web
com.ebay.maui.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.maui.driver.web.factory
AbstractWebDriverFactory.class
AbstractWebDriverFactory
AndroidCapabilitiesFactory.class
ChromeCapabilitiesFactory.class
ChromeDriverFactory.class
FirefoxCapabilitiesFactory.class
FirefoxDriverFactory.class
HtmlUnitCapabilitiesFactory.class
HtmlUnitDriverFactory.class
ICapabilitiesFactory.class
IECapabilitiesFactory.class
IEDriverFactory.class
IWebDriverFactory.class
OperaCapabilitiesFactory.class
OperaDriverFactory.class
PhantomJSCapabilitiesFactory.class
RemoteDriverFactory.class
SafariCapabilitiesFactory.class
ChromeDriverFac
FirefoxCapabili
FirefoxDriverFa
HtmlUnitCapabil
HtmlUnitDriverF
com.ebay.maui.driver.web.factory
HtmlUnitDriverFactory
package com.ebay.maui.driver.web.factory;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.htmlunit.HtmlUnitDriver;
import com.ebay.maui.controller.Logging;
import com.ebay.maui.driver.web.WebDriverConfig;
public class HtmlUnitDriverFactory extends AbstractWebDriverFactory implements IWebDriverFactory{
public HtmlUnitDriverFactory(WebDriverConfig cfg) {
super(cfg);
}
@Override
public WebDriver createWebDriver() {
WebDriverConfig cfg = this.getWebDriverConfig();
driver = new HtmlUnitDriver(new HtmlUnitCapabilitiesFactory().createCapabilities(cfg));
//Implicit Waits to handle dynamic element. The default value is 5 seconds.
setImplicitWaitTimeout(cfg.getImplicitWaitTimeout());
if(cfg.getPageLoadTimeout()>=0)
{
Logging.log("htmlunit doesn't support pageLoadTimeout, ignoring...");
//java.lang.RuntimeException: java.lang.UnsupportedOperationException: pageLoadTimeout
//driver.manage().timeouts().pageLoadTimeout(cfg.getPageLoadTimeout(), TimeUnit.SECONDS);
}
this.setWebDriver(driver);
return driver;
}
}
Outli
Task
HtmlUnitDriverFactory
HtmlUnitDriverFactory(W
createWebDriver() : WebD
Markers
Properties
Servers
Data Source Explorer
Snippets
Console
Progress
Results of running class RaiseCla
Package Explorer
Call Hierar
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
Line
Call
Read-Only
Smart Insert
1 : 1

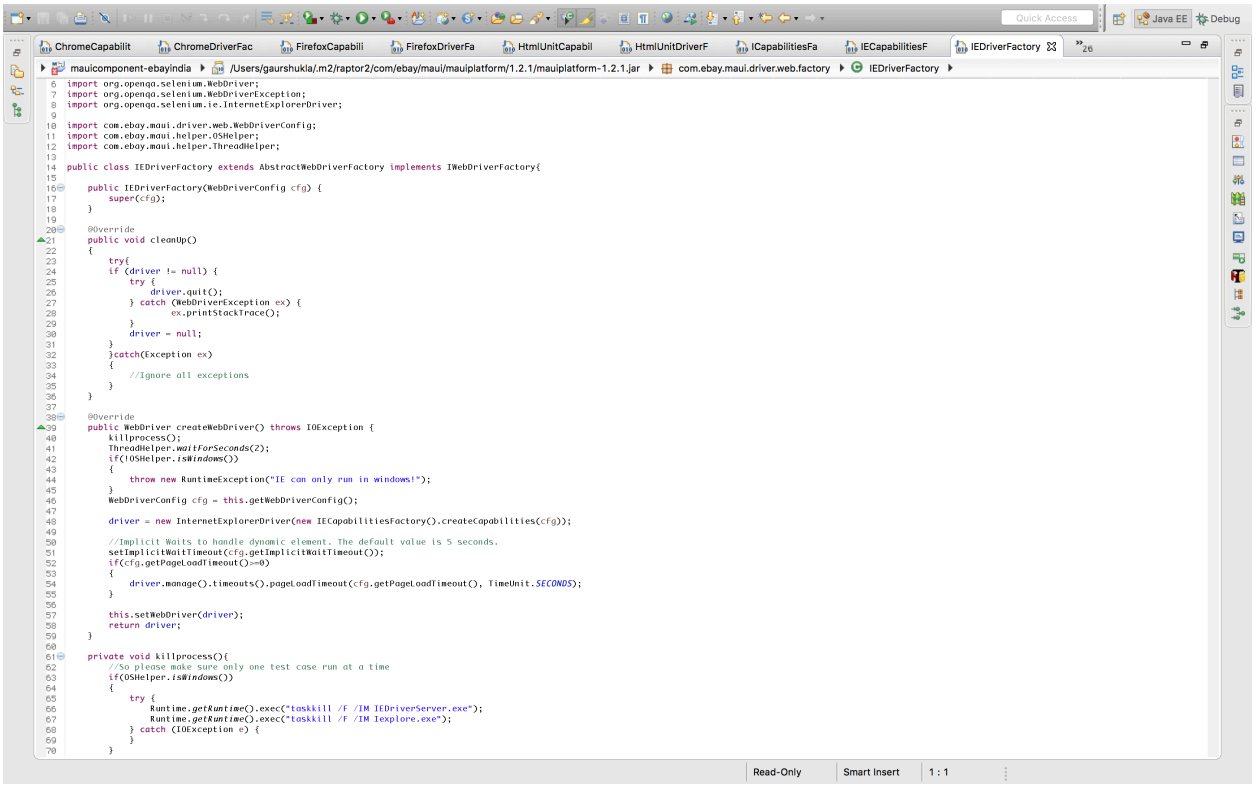

Project Explore
Navigator
Type Hierarchy
com.ebay.maui.driver.web
com.ebay.maui.driver.web.element
BaseHtmlPage.class
BaseHtmlPage
BasePageObject.class
Button.class
CheckBox.class
Form.class
GenericWebPage.class
HtmlElement.class
Image.class
IPage.class
Label.class
Link.class
LocatorSwitch.class
LocatorType.class
RadioButton.class
SelectList.class
Table.class
TextField.class
WebPage.class
WebPageSection.class
WebPageSection
com.ebay.maui.driver.web.factory
AbstractWebDriverFactory.class
AbstractWebDriverFactory
AndroidCapabilitiesFactory.class
ChromeCapabilitiesFactory.class
ChromeDriverFactory.class
FirefoxCapabilitiesFactory.class
FirefoxDriverFactory.class
HtmlUnitCapabilitiesFactory.class
HtmlUnitDriverFactory.class
ICapabilitiesFactory.class
IECapabilitiesFactory.class
IEDriverFactory.class
IWebDriverFactory.class
OperaCapabilitiesFactory.class
OperaDriverFactory.class
PhantomJSCapabilitiesFactory.class
RemoteDriverFactory.class
SafariCapabilitiesFactory.class
SafariDriverFactory.class
com.ebay.maui.exception
APIException.class
DataCreationException.class
FirefoxCapabili
FirefoxDriverFa
HtmlUnitCapabil
HtmlUnitDriverF
ICapabilitiesFa
IECapabilitiesF
com.ebay.maui.driver.web.factory
ICapabilitiesFactory
package com.ebay.maui.driver.web.factory;
import org.openqa.selenium.remote.DesiredCapabilities;
import com.ebay.maui.driver.web.WebDriverConfig;
public interface ICapabilitiesFactory {
public DesiredCapabilities createCapabilities(WebDriverConfig cfg);
}
Outli
Task
ICapabilitiesFactory
createCapabilities(WebDr
Markers
Properties
Servers
Data Source Explorer
Snippets
Console
Progress
Results of running class RaiseCla
Package Explorer
Call Hierar
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
Line
Call
Read-Only
Smart Insert
1 : 1



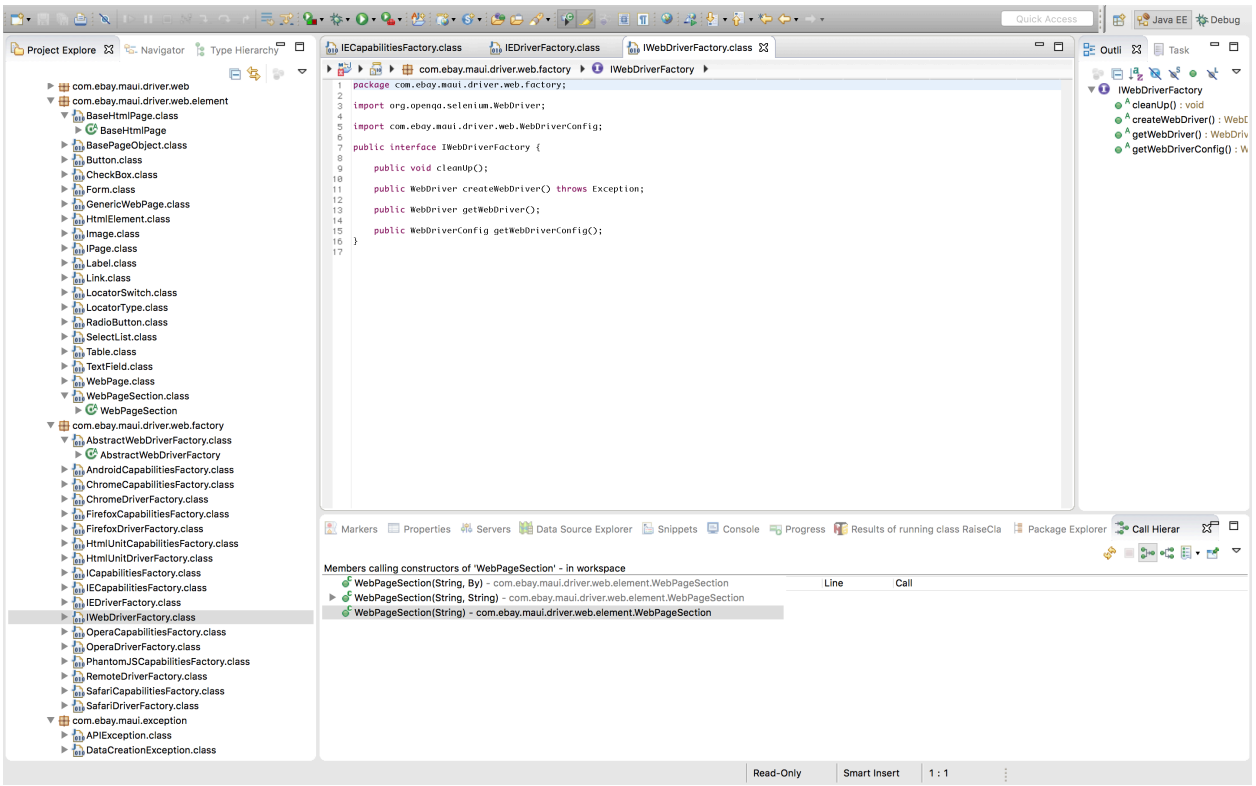
```
public class IECapabilitiesFactory implements ICapabilitiesFactory {

    private void handleExtractResources() throws IOException {
        String dir = this.getClass().getResource("/").getPath()
                + "mauiexternal";
        dir = FileHelper.decodePath(dir);
        if (!new File(dir).exists()) {
            FileHelper.extractJar(dir, WebDriverExternalResources.class);
        }
        if (!new File(dir + "\\IEDriverServer.exe").exists()) {
            if (OSHelper.getIEVersion() < 10) {
                FileHelper.copyFile(dir + "\\IEDriverServer_x64.exe", dir
                        + "\\IEDriverServer.exe");
            } else
                FileHelper.copyFile(dir + "\\IEDriverServer_Win32.exe", dir
                        + "\\IEDriverServer.exe");// Win32
        }
        System.setProperty("webdriver.ie.driver", dir + "\\IEDriverServer.exe");
        System.out.println(dir + "\\IEDriverServer.exe");
    }

    public DesiredCapabilities createCapabilities(WebDriverConfig cfg) {
        // Set IEDriver for Local Mode
        if (cfg.getMode() == WebDriverMode.LocallyOnRC) {
            if (cfg.getIeDriverPath() != null)
                System.setProperty("webdriver.ie.driver", cfg.getIeDriverPath());
            else {
                if (System.getenv("webdriver.ie.driver")!=null)
                {
                    System.out.println("ie driver get from property:"+System.getenv("webdriver.ie.driver"));
                    System.setProperty("webdriver.ie.driver",System.getenv("webdriver.ie.driver"));
                }
                else{
                    try {
                        handleExtractResources();
                    } catch (IOException ex) {
                    }
                }
            }

        }

        DesiredCapabilities capability = DesiredCapabilities.internetExplorer();

        if (cfg.isEnableJavascript())
            capability.setJavascriptEnabled(true);
        else
            capability.setJavascriptEnabled(false);
        capability.setCapability(CapabilityType.TAKES_SCREENSHOT, true);
        capability.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);
        capability.setCapability(InternetExplorerDriver.IGNORE_ZOOM_SETTING, true);

        if (cfg.getBrowserVersion() != null) {
            capability.setVersion(cfg.getBrowserVersion());
        }

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

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

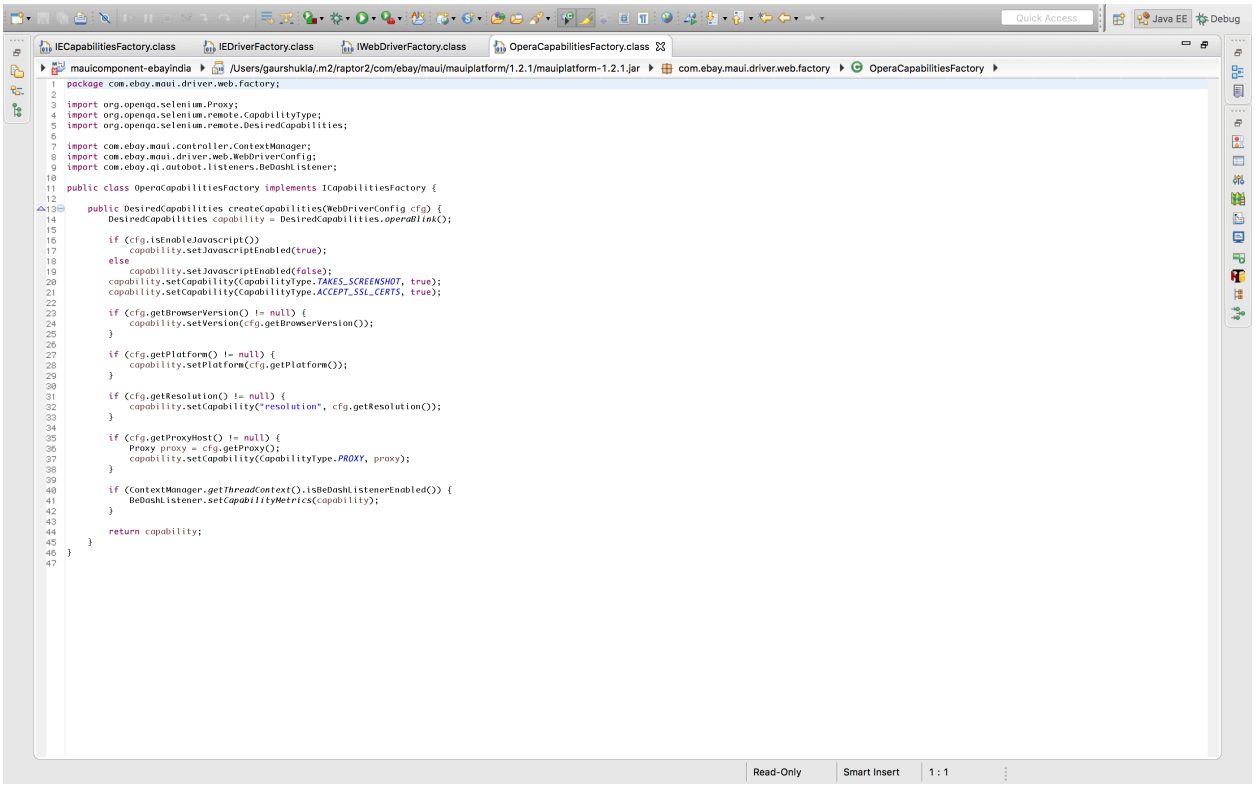
```
        if (cfg.getMode() == WebDriverMode.LocallyOnRC) {
            if (cfg.getIeDriverPath() != null)
                System.setProperty("webdriver.ie.driver", cfg.getIeDriverPath());
            else {
                if (System.getenv("webdriver.ie.driver")!=null)
                {
                    System.out.println("ie driver get from property:"+System.getenv("webdriver.ie.driver"));
                    System.setProperty("webdriver.ie.driver",System.getenv("webdriver.ie.driver"));
                }
                else{
                    try {
                        handleExtractResources();
                    } catch (IOException ex) {
                    }
                }
            }

        }

        DesiredCapabilities capability = DesiredCapabilities.internetExplorer();

        if (cfg.isEnableJavascript())
            capability.setJavascriptEnabled(true);
        else
            capability.setJavascriptEnabled(false);
        capability.setCapability(CapabilityType.TAKES_SCREENSHOT, true);
        capability.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);
        capability.setCapability(InternetExplorerDriver.IGNORE_ZOOM_SETTING, true);

        if (cfg.getBrowserVersion() != null) {
            capability.setVersion(cfg.getBrowserVersion());
        }

        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);
        }
        // To ignore Protected Mode exception, but users should set the browser
        // and reduce the strange behavior --
        // If a Protected Mode boundary is crossed, very unexpected behavior
        // including hangs, element location not working, and clicks not being
        // propagated
        // It should not be a big problem in our testing since almost we should
        // test in the same zone. added in 6/18/2013
        capability
                .setCapability(
                        InternetExplorerDriver.INTRODUCE_FLAKINESS_BY_IGNORING_SECURITY_DOMAINS,
                        true);

        if (ContextManager.getThreadContext().isBeDashListenerEnabled()) {
            BeDashListener.setCapabilityMetrics(capability);
        }

        return capability;
    }
}
```

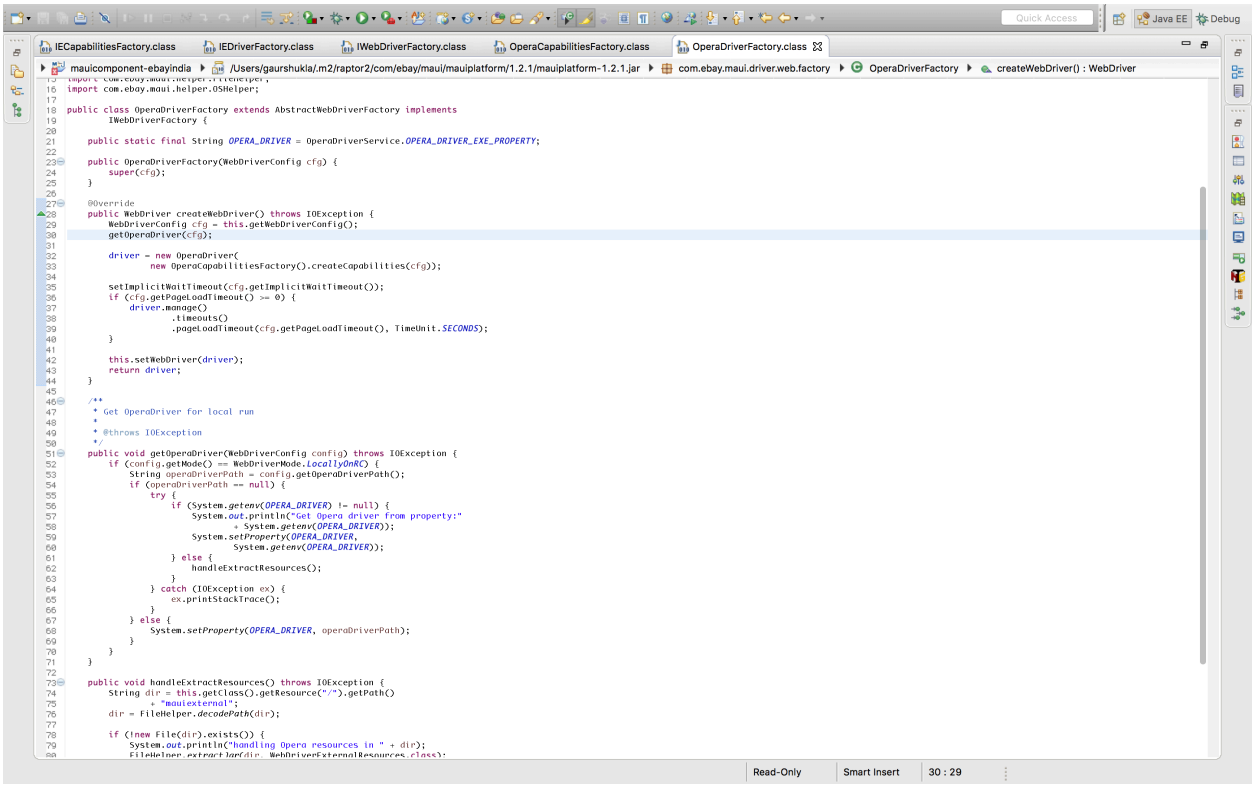


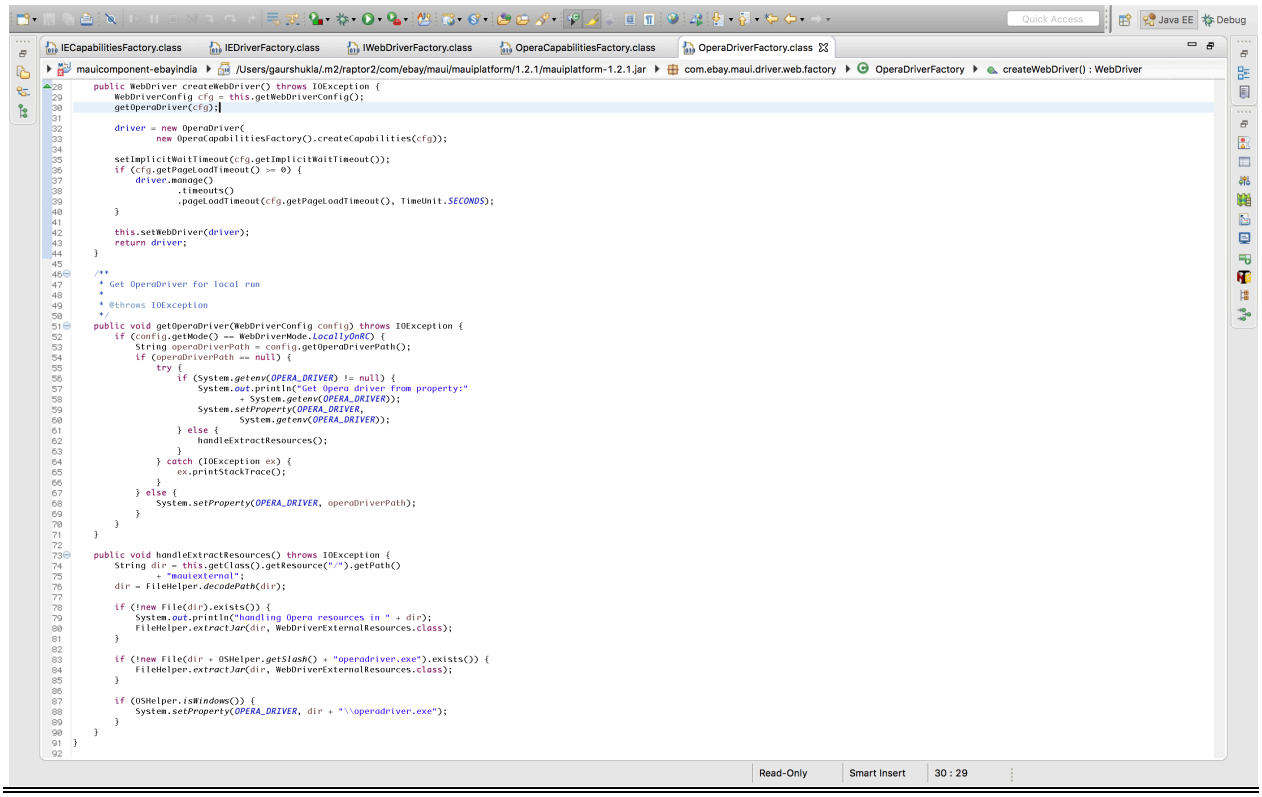
```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebDriverException;
import org.openqa.selenium.ie.InternetExplorerDriver;

import com.ebay.maui.driver.web.WebDriverConfig;
import com.ebay.maui.helper.OSHelper;
import com.ebay.maui.helper.ThreadHelper;

public class IEDriverFactory extends AbstractWebDriverFactory implements IWebDriverFactory{

    public IEDriverFactory(WebDriverConfig cfg) {
        super(cfg);
    }

    @Override
    public void cleanUp()
    {
        try{
        if (driver != null) {
            try {
                driver.quit();
            } catch (WebDriverException ex) {
                    ex.printStackTrace();
            }
            driver = null;
        }
        }catch(Exception ex)
        {
            //Ignore all exceptions
        }
    }

    @Override
    public WebDriver createWebDriver() throws IOException {
        killprocess();
        ThreadHelper.waitForSeconds(2);
        if(!OSHelper.isWindows())
        {
            throw new RuntimeException("IE can only run in windows!");
        }
        WebDriverConfig cfg = this.getWebDriverConfig();

        driver = new InternetExplorerDriver(new IECapabilitiesFactory().createCapabilities(cfg));

        //Implicit Waits to handle dynamic element. The default value is 5 seconds.
        setImplicitWaitTimeout(cfg.getImplicitWaitTimeout());
        if(cfg.getPageLoadTimeout()>=0)
        {
            driver.manage().timeouts().pageLoadTimeout(cfg.getPageLoadTimeout(), TimeUnit.SECONDS);
        }

        this.setWebDriver(driver);
        return driver;
    }

    private void killprocess(){
        //So please make sure only one test case run at a time
        if(OSHelper.isWindows())
        {
            try {
                Runtime.getRuntime().exec("taskkill /F /IM IEDriverServer.exe");
                Runtime.getRuntime().exec("taskkill /F /IM iexplore.exe");
            } catch (IOException e) {
            }
        }
```

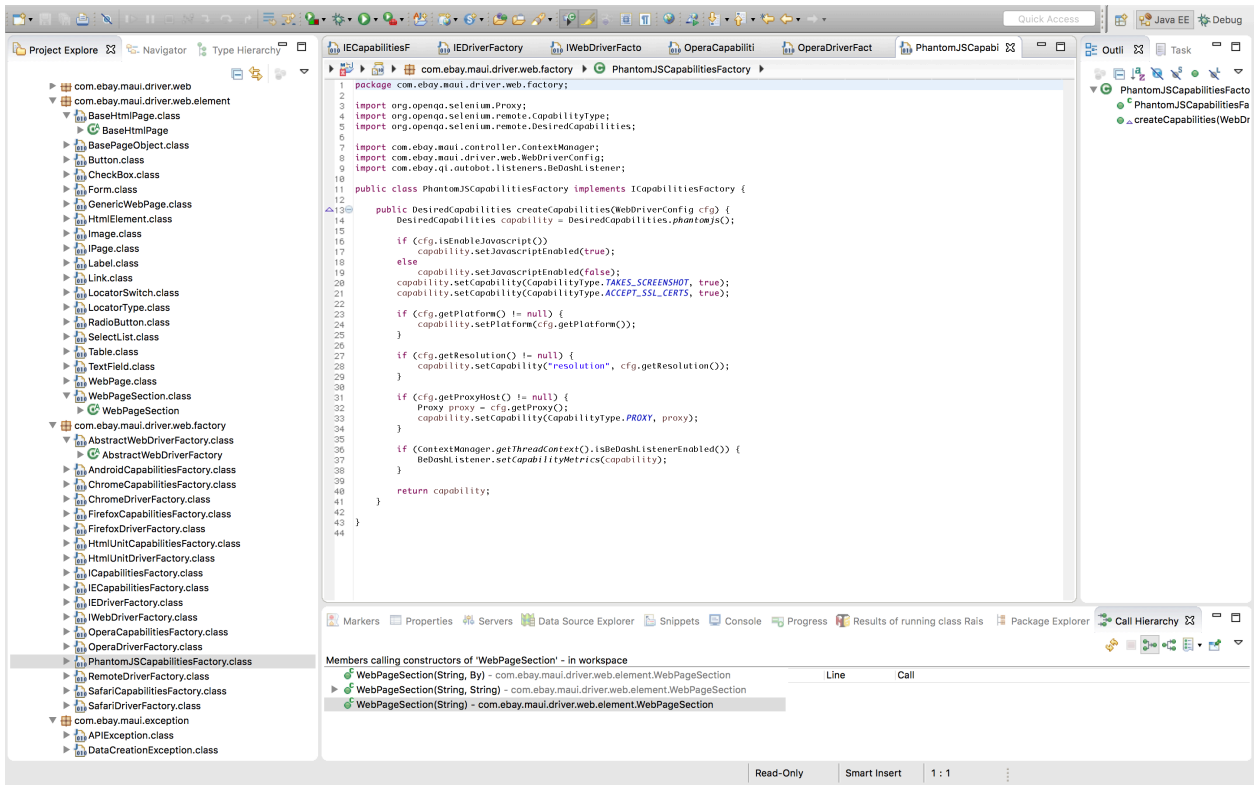


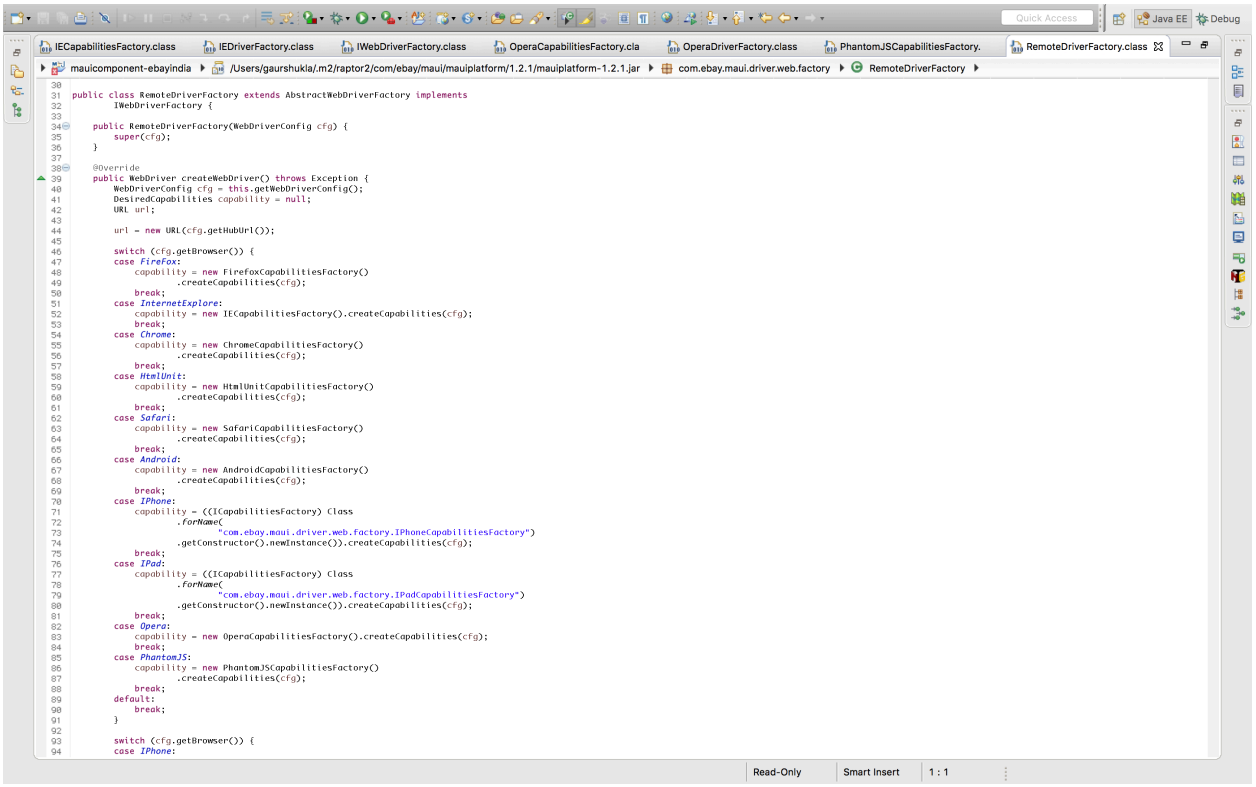
```
package com.ebay.maui.driver.web.factory;

import org.openqa.selenium.WebDriver;

import com.ebay.maui.driver.web.WebDriverConfig;

public interface IWebDriverFactory {

    public void cleanUp();

    public WebDriver createWebDriver() throws Exception;

    public WebDriver getWebDriver();

    public WebDriverConfig getWebDriverConfig();
}
```


IECapabilitiesFactory.class
IEDriverFactory.class
IWebDriverFactory.class
OperaCapabilitiesFactory.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web.factory
OperaCapabilitiesFactory
Read-Only
Smart Insert
1 : 1

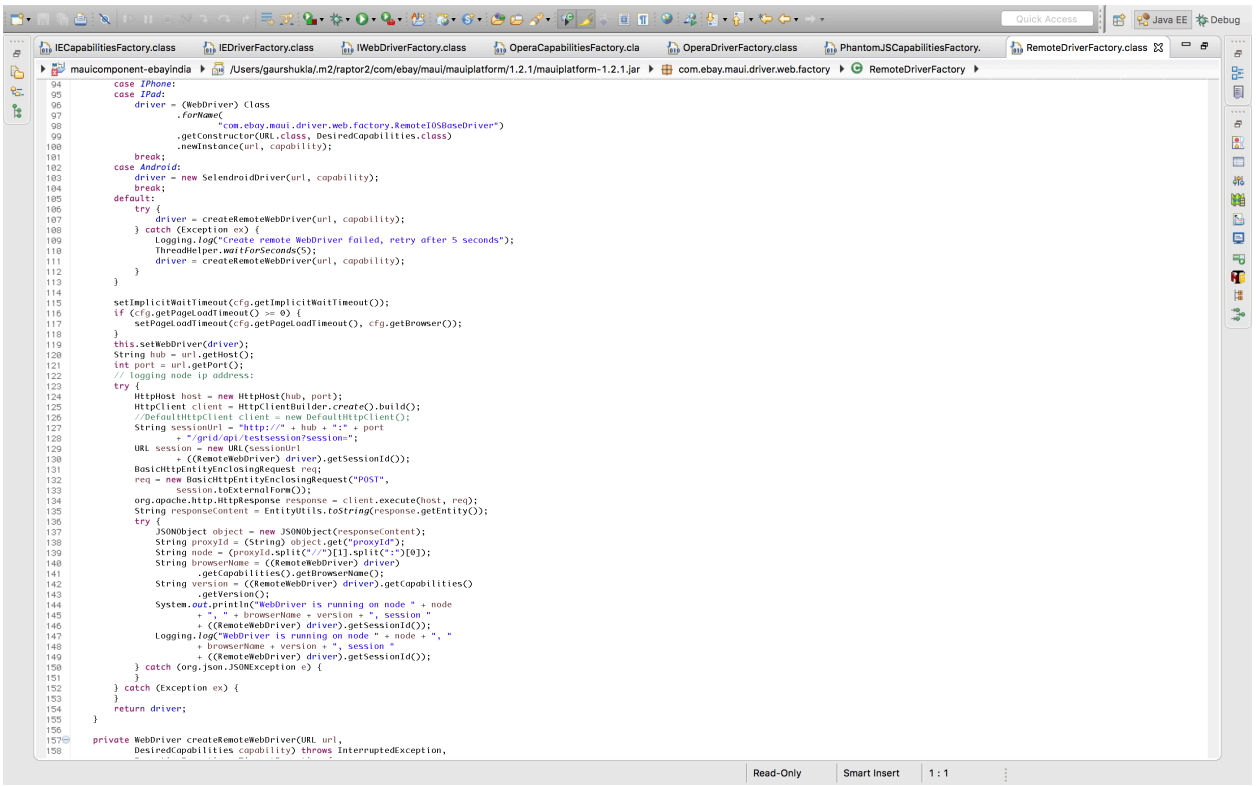

IECapabilitiesFactory.class
IEDriverFactory.class
IWebDriverFactory.class
OperaCapabilitiesFactory.class
OperaDriverFactory.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web.factory
OperaDriverFactory
createWebDriver() : WebDriver
Read-Only
Smart Insert
30 : 29


IECapabilitiesFactory.class
IEDriverFactory.class
IWebDriverFactory.class
OperaCapabilitiesFactory.class
OperaDriverFactory.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.driver.web.factory
OperaDriverFactory
createWebDriver() : WebDriver
Read-Only
Smart Insert
30 : 29

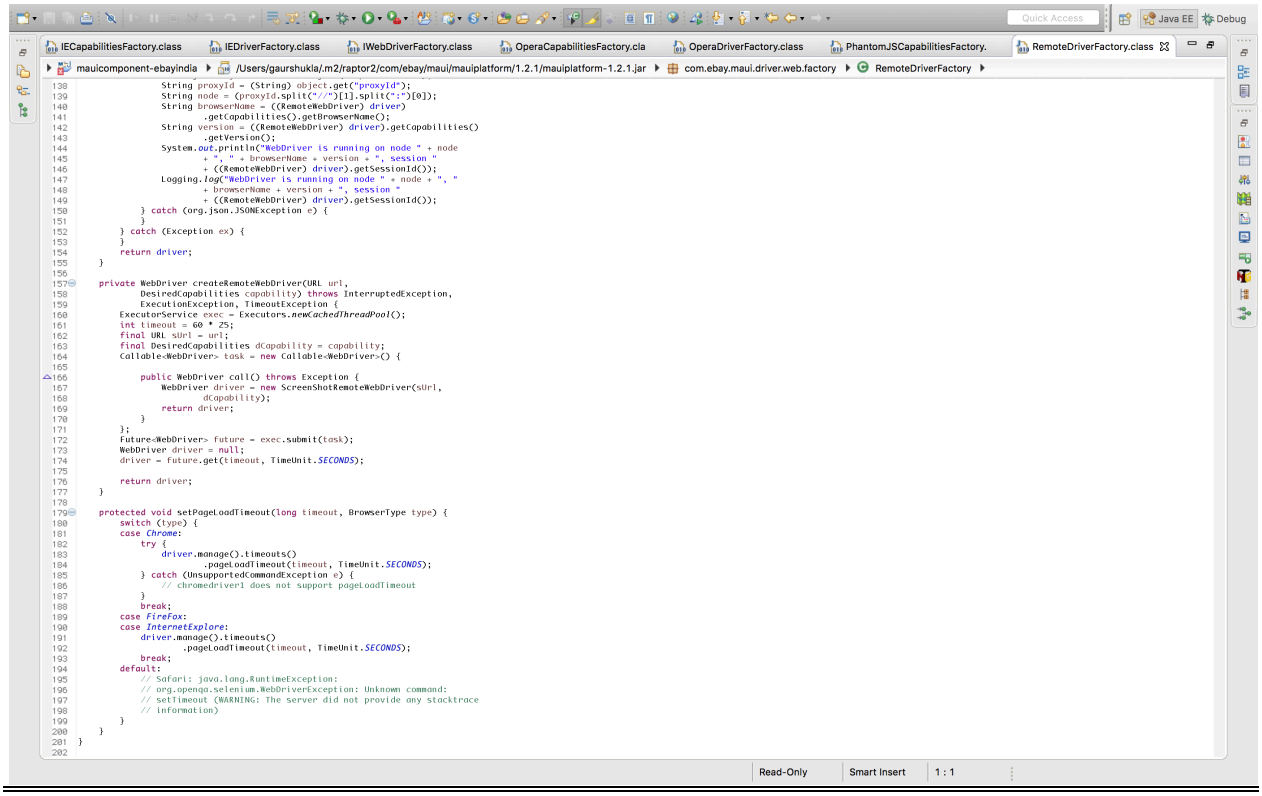

Project Explorer
Navigator
Type Hierarchy
com.ebay.maui.driver.web
com.ebay.maui.driver.web.element
com.ebay.maui.driver.web.factory
com.ebay.maui.exception
PhantomJSCapabilitiesFactory.class
com.ebay.maui.driver.web.factory
PhantomJSCapabilitiesFactory
Markers
Properties
Servers
Data Source Explorer
Snippets
Console
Progress
Results of running class Rais
Package Explorer
Call Hierarchy
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
Line
Call
Read-Only
Smart Insert
1 : 1



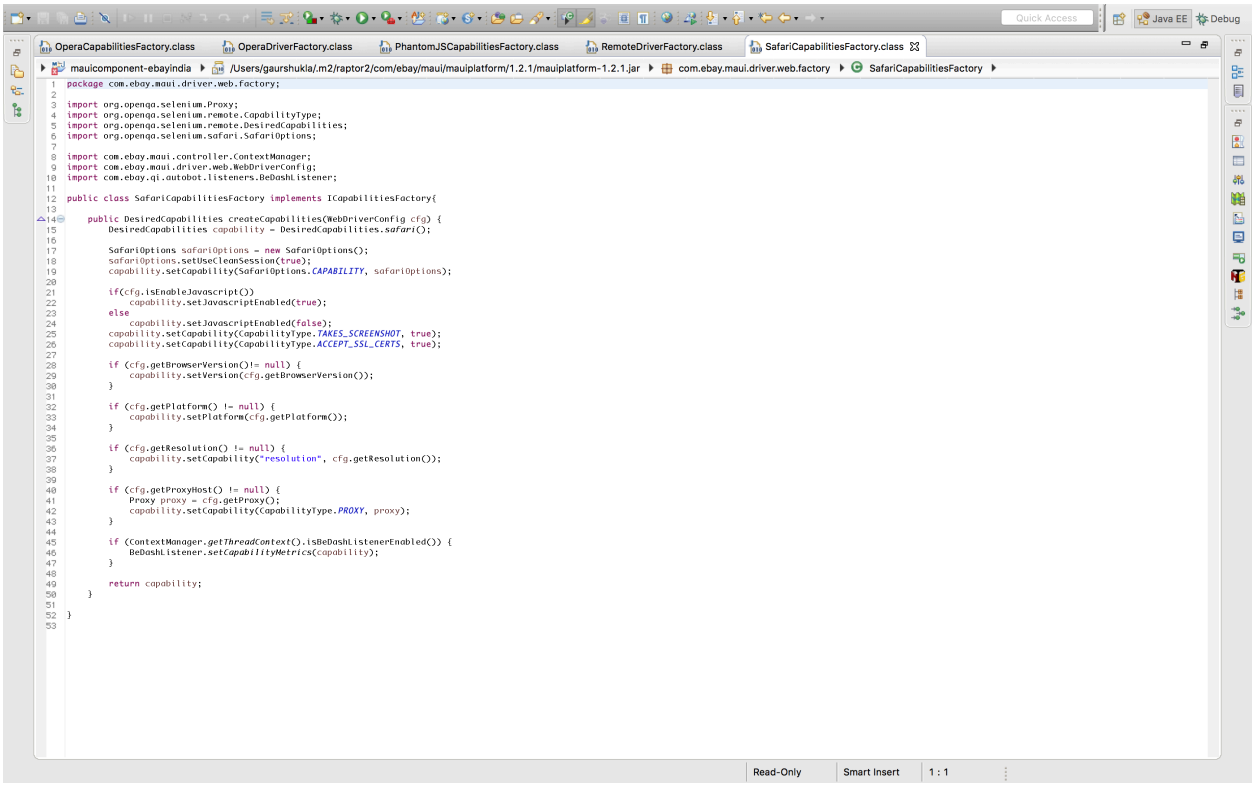
```
30
31  public class RemoteDriverFactory extends AbstractWebDriverFactory implements
32          IWebDriverFactory {
33
34      public RemoteDriverFactory(WebDriverConfig cfg) {
35          super(cfg);
36      }
37
38      @Override
39      public WebDriver createWebDriver() throws Exception {
40          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              break;
51          case InternetExplore:
52              capability = new IECapabilitiesFactory().createCapabilities(cfg);
53              break;
54          case Chrome:
55              capability = new ChromeCapabilitiesFactory()
56                      .createCapabilities(cfg);
57              break;
58          case HtmlUnit:
59              capability = new HtmlUnitCapabilitiesFactory()
60                      .createCapabilities(cfg);
61              break;
62          case Safari:
63              capability = new SafariCapabilitiesFactory()
64                      .createCapabilities(cfg);
65              break;
66          case Android:
67              capability = new AndroidCapabilitiesFactory()
68                      .createCapabilities(cfg);
69              break;
70          case IPhone:
71              capability = ((ICapabilitiesFactory) Class
72                      .forName(
73                              "com.ebay.maui.driver.web.factory.IPhoneCapabilitiesFactory")
74                      .getConstructor().newInstance()).createCapabilities(cfg);
75              break;
76          case IPad:
77              capability = ((ICapabilitiesFactory) Class
78                      .forName(
79                              "com.ebay.maui.driver.web.factory.IPadCapabilitiesFactory")
80                      .getConstructor().newInstance()).createCapabilities(cfg);
81              break;
82          case Opera:
83              capability = new OperaCapabilitiesFactory().createCapabilities(cfg);
84              break;
85          case PhantomJS:
86              capability = new PhantomJSCapabilitiesFactory()
87                      .createCapabilities(cfg);
88              break;
89          default:
90              break;
91          }
92
93          switch (cfg.getBrowser()) {
94          case IPhone:
```



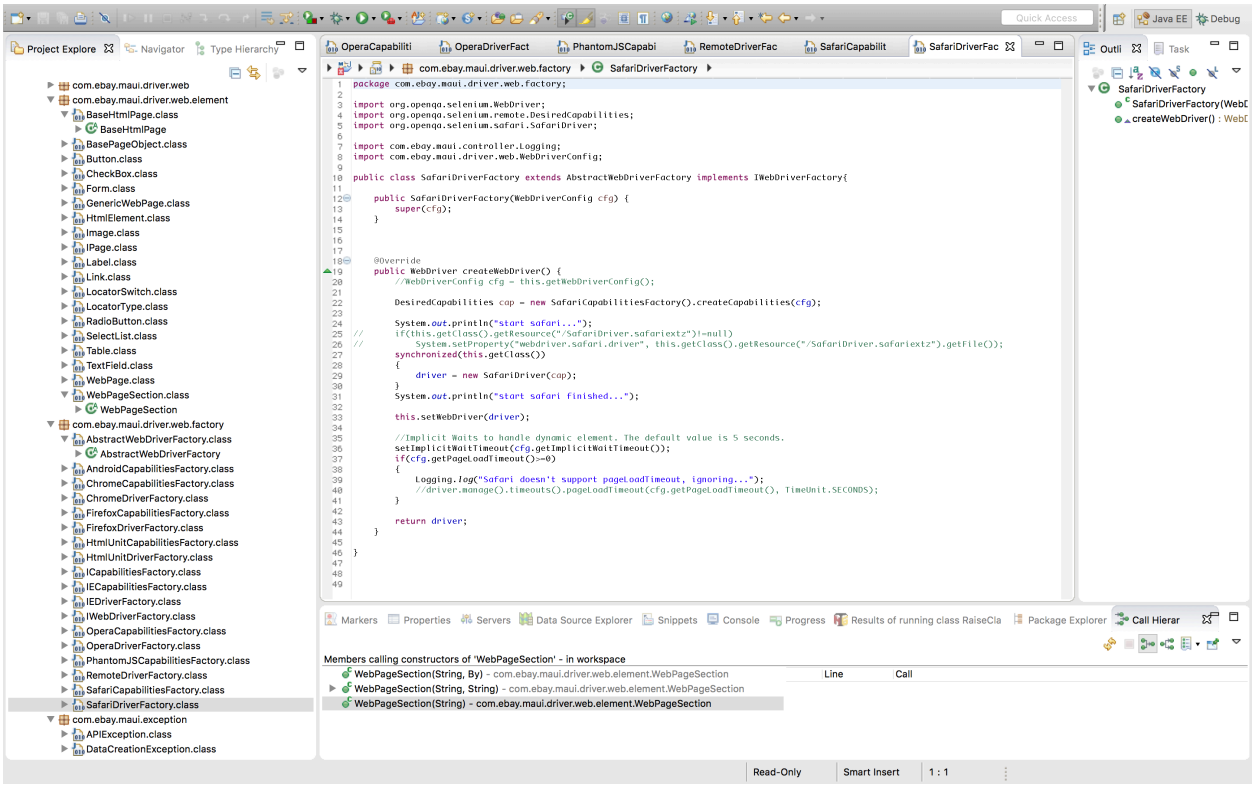
```
94          case IPhone:
95          case IPad:
96              driver = (WebDriver) Class
97                      .forName(
98                              "com.ebay.maui.driver.web.factory.RemoteIOSBaseDriver")
99                      .getConstructor(URL.class, DesiredCapabilities.class)
100                     .newInstance(url, capability);
101             break;
102         case Android:
103             driver = new SelendroidDriver(url, capability);
104             break;
105         default:
106             try {
107                 driver = createRemoteWebDriver(url, capability);
108             } catch (Exception ex) {
109                 Logging.log("Create remote WebDriver failed, retry after 5 seconds");
110                 ThreadHelper.waitForSeconds(5);
111                 driver = createRemoteWebDriver(url, capability);
112             }
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 node ip 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?session=";
129             URL session = new URL(sessionUrl
130                     + ((RemoteWebDriver) driver).getSessionId());
131             BasicHttpEntityEnclosingRequest req;
132             req = new BasicHttpEntityEnclosingRequest("POST",
133                     session.toExternalForm());
134             org.apache.http.HttpResponse response = client.execute(host, req);
135             String responseContent = EntityUtils.toString(response.getEntity());
136             try {
137                 JSONObject object = new JSONObject(responseContent);
138                 String proxyId = (String) object.get("proxyId");
139                 String node = (proxyId.split("//")[1].split(":")[0]);
140                 String browserName = ((RemoteWebDriver) driver)
141                         .getCapabilities().getBrowserName();
142                 String version = ((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         return driver;
155     }
156
157     private WebDriver createRemoteWebDriver(URL url,
158             DesiredCapabilities capability) throws InterruptedException,
```



```
                String proxyId = (String) object.get("proxyId");
                String node = (proxyId.split("//")[1].split(":")[0]);
                String browserName = ((RemoteWebDriver) driver)
                        .getCapabilities().getBrowserName();
                String version = ((RemoteWebDriver) driver).getCapabilities()
                        .getVersion();
                System.out.println("WebDriver is running on node " + node
                        + ", " + browserName + version + ", session "
                        + ((RemoteWebDriver) driver).getSessionId());
                Logging.log("WebDriver is running on node " + node + ", "
                        + browserName + version + ", session "
                        + ((RemoteWebDriver) driver).getSessionId());
            } catch (org.json.JSONException e) {
            }
        } catch (Exception ex) {
        }
        return driver;
    }

    private WebDriver createRemoteWebDriver(URL url,
            DesiredCapabilities capability) throws InterruptedException,
            ExecutionException, TimeoutException {
        ExecutorService exec = Executors.newCachedThreadPool();
        int timeout = 60 * 25;
        final URL sUrl = url;
        final DesiredCapabilities dCapability = capability;
        Callable<WebDriver> task = new Callable<WebDriver>() {

            public WebDriver call() throws Exception {
                WebDriver driver = new ScreenShotRemoteWebDriver(sUrl,
                        dCapability);
                return driver;
            }
        };
        Future<WebDriver> future = exec.submit(task);
        WebDriver driver = null;
        driver = future.get(timeout, TimeUnit.SECONDS);

        return driver;
    }

    protected void setPageLoadTimeout(long timeout, BrowserType type) {
        switch (type) {
        case Chrome:
            try {
                driver.manage().timeouts()
                        .pageLoadTimeout(timeout, TimeUnit.SECONDS);
            } catch (UnsupportedCommandException e) {
                // chromedriver1 does not support pageLoadTimeout
            }
            break;
        case FireFox:
        case InternetExplore:
            driver.manage().timeouts()
                    .pageLoadTimeout(timeout, TimeUnit.SECONDS);
            break;
        default:
            // Safari: java.lang.RuntimeException:
            // org.openqa.selenium.WebDriverException: Unknown command:
            // setTimeout (WARNING: The server did not provide any stacktrace
            // information)
        }
    }
}
```



```
package com.ebay.maui.driver.web.factory;

import org.openqa.selenium.Proxy;
import org.openqa.selenium.remote.CapabilityType;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.safari.SafariOptions;

import com.ebay.maui.controller.ContextManager;
import com.ebay.maui.driver.web.WebDriverConfig;
import com.ebay.qi.autobot.listeners.BeDashListener;

public class SafariCapabilitiesFactory implements ICapabilitiesFactory{

    public DesiredCapabilities createCapabilities(WebDriverConfig cfg) {
        DesiredCapabilities capability = DesiredCapabilities.safari();

        SafariOptions safariOptions = new SafariOptions();
        safariOptions.setUseCleanSession(true);
        capability.setCapability(SafariOptions.CAPABILITY, safariOptions);

        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.getBrowserVersion()!= null) {
            capability.setVersion(cfg.getBrowserVersion());
        }

        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().isBeDashListenerEnabled()) {
            BeDashListener.setCapabilityMetrics(capability);
        }

        return capability;
    }

}
```


Project Explore
Navigator
Type Hierarchy
com.ebay.maui.driver.web
com.ebay.maui.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.maui.driver.web.factory
AbstractWebDriverFactory.class
AbstractWebDriverFactory
AndroidCapabilitiesFactory.class
ChromeCapabilitiesFactory.class
ChromeDriverFactory.class
FirefoxCapabilitiesFactory.class
FirefoxDriverFactory.class
HtmlUnitCapabilitiesFactory.class
HtmlUnitDriverFactory.class
ICapabilitiesFactory.class
IECapabilitiesFactory.class
IEDriverFactory.class
IWebDriverFactory.class
OperaCapabilitiesFactory.class
OperaDriverFactory.class
PhantomJSCapabilitiesFactory.class
RemoteDriverFactory.class
SafariCapabilitiesFactory.class
SafariDriverFactory.class
com.ebay.maui.exception
APIException.class
DataCreationException.class
OperaCapabiliti
OperaDriverFact
PhantomJSCapabi
RemoteDriverFac
SafariCapabilit
SafariDriverFac
com.ebay.maui.driver.web.factory
SafariDriverFactory
package com.ebay.maui.driver.web.factory;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.safari.SafariDriver;
import com.ebay.maui.controller.Logging;
import com.ebay.maui.driver.web.WebDriverConfig;
public class SafariDriverFactory extends AbstractWebDriverFactory implements IWebDriverFactory{
public SafariDriverFactory(WebDriverConfig cfg) {
super(cfg);
}
@Override
public WebDriver createWebDriver() {
//WebDriverConfig cfg = this.getWebDriverConfig();
DesiredCapabilities cap = new SafariCapabilitiesFactory().createCapabilities(cfg);
System.out.println("start safari...");
// if(this.getClass().getResource("/SafariDriver.safariextz")!=null)
// System.setProperty("webdriver.safari.driver", this.getClass().getResource("/SafariDriver.safariextz").getFile());
synchronized(this.getClass())
{
driver = new SafariDriver(cap);
}
System.out.println("start safari finished...");
this.setWebDriver(driver);
//Implicit Waits to handle dynamic element. The default value is 5 seconds.
setImplicitWaitTimeout(cfg.getImplicitWaitTimeout());
if(cfg.getPageLoadTimeout()>=0)
{
Logging.log("Safari doesn't support pageLoadTimeout, ignoring...");
//driver.manage().timeouts().pageLoadTimeout(cfg.getPageLoadTimeout(), TimeUnit.SECONDS);
}
return driver;
}
}
Markers
Properties
Servers
Data Source Explorer
Snippets
Console
Progress
Results of running class RaiseCla
Package Explorer
Call Hierar
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
Line
Call
Read-Only
Smart Insert
1 : 1