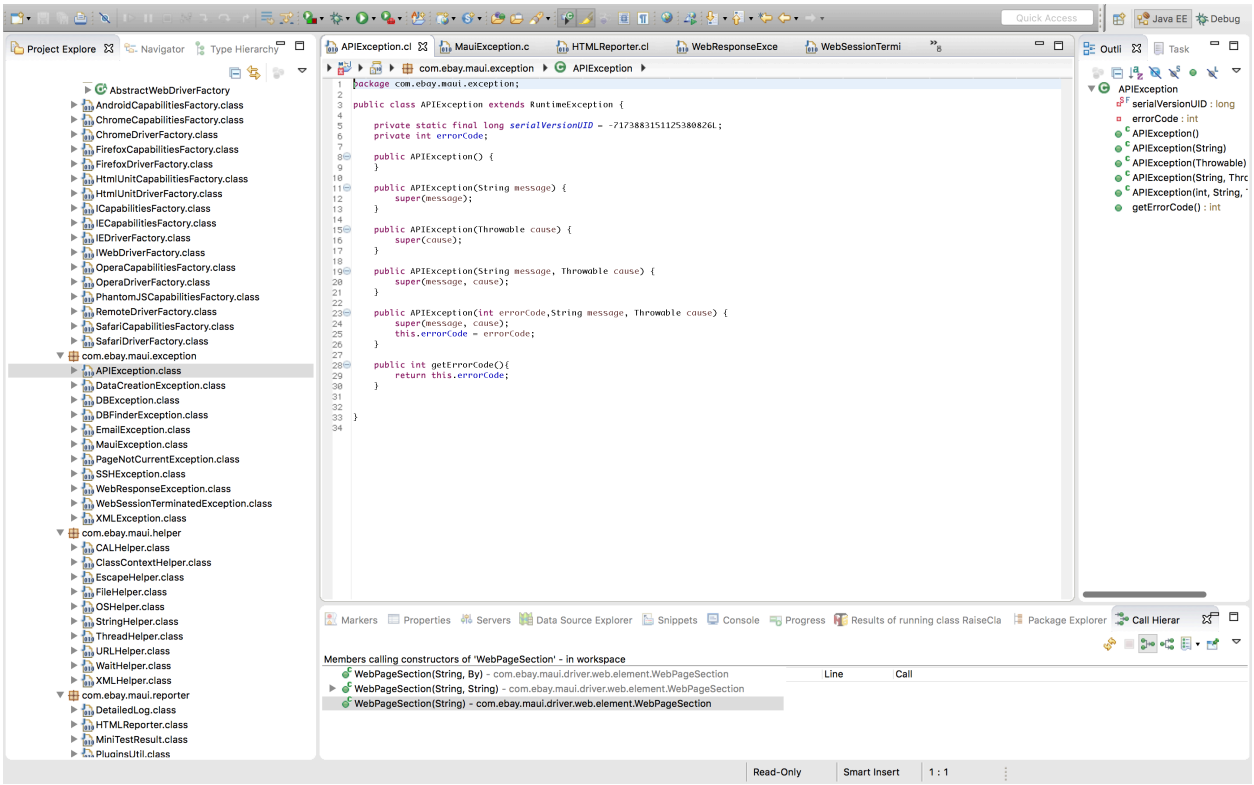
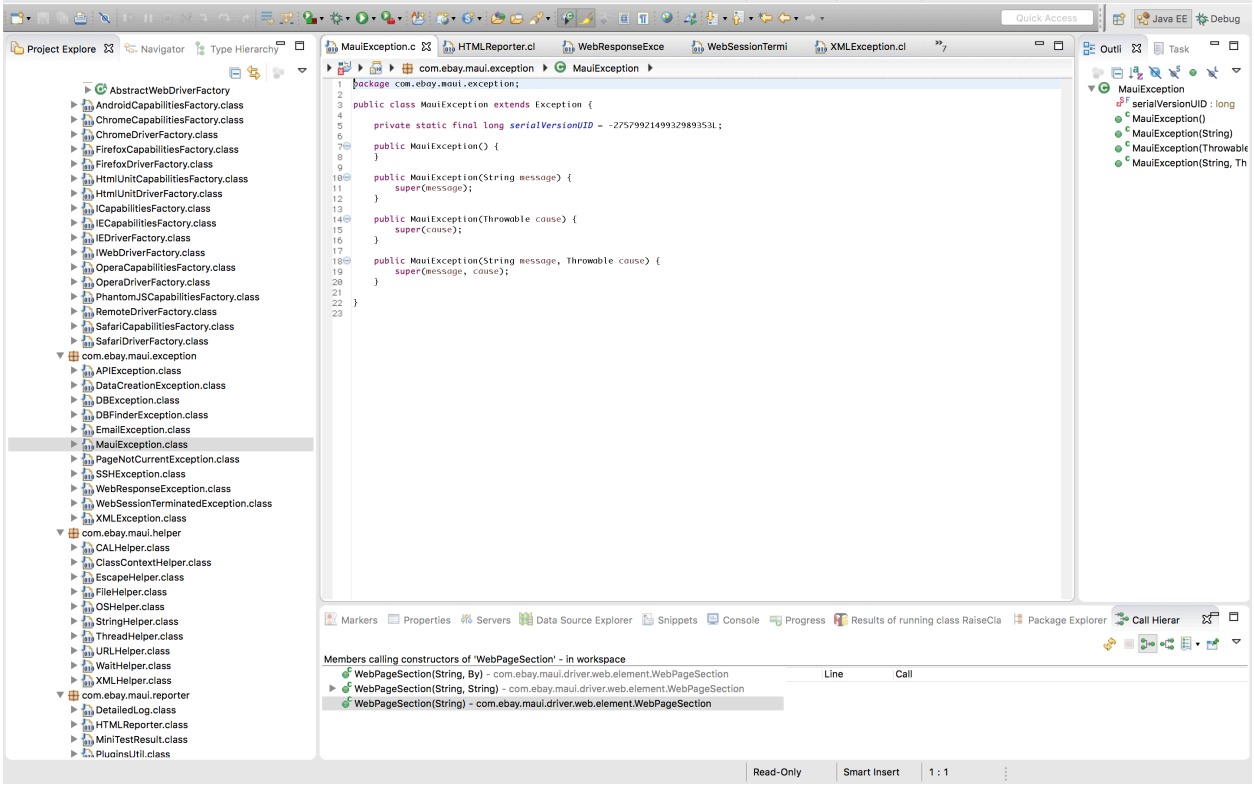
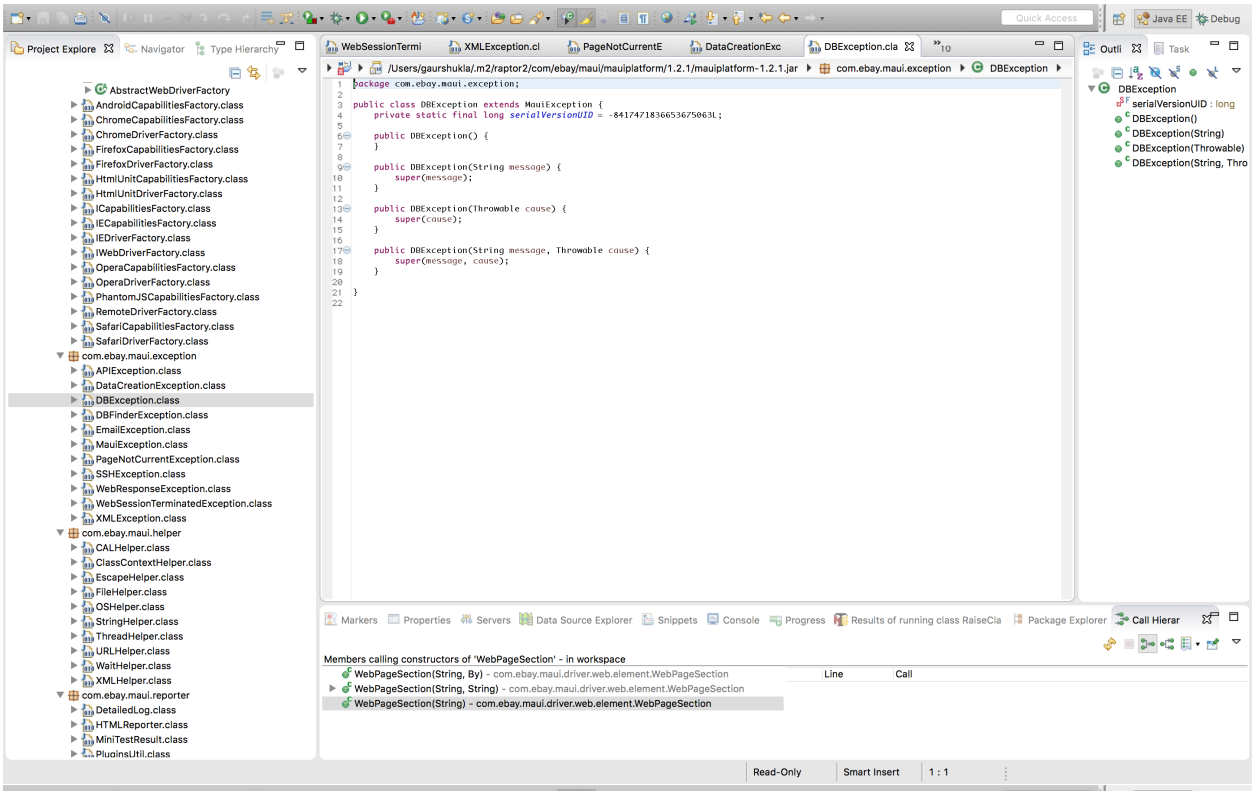
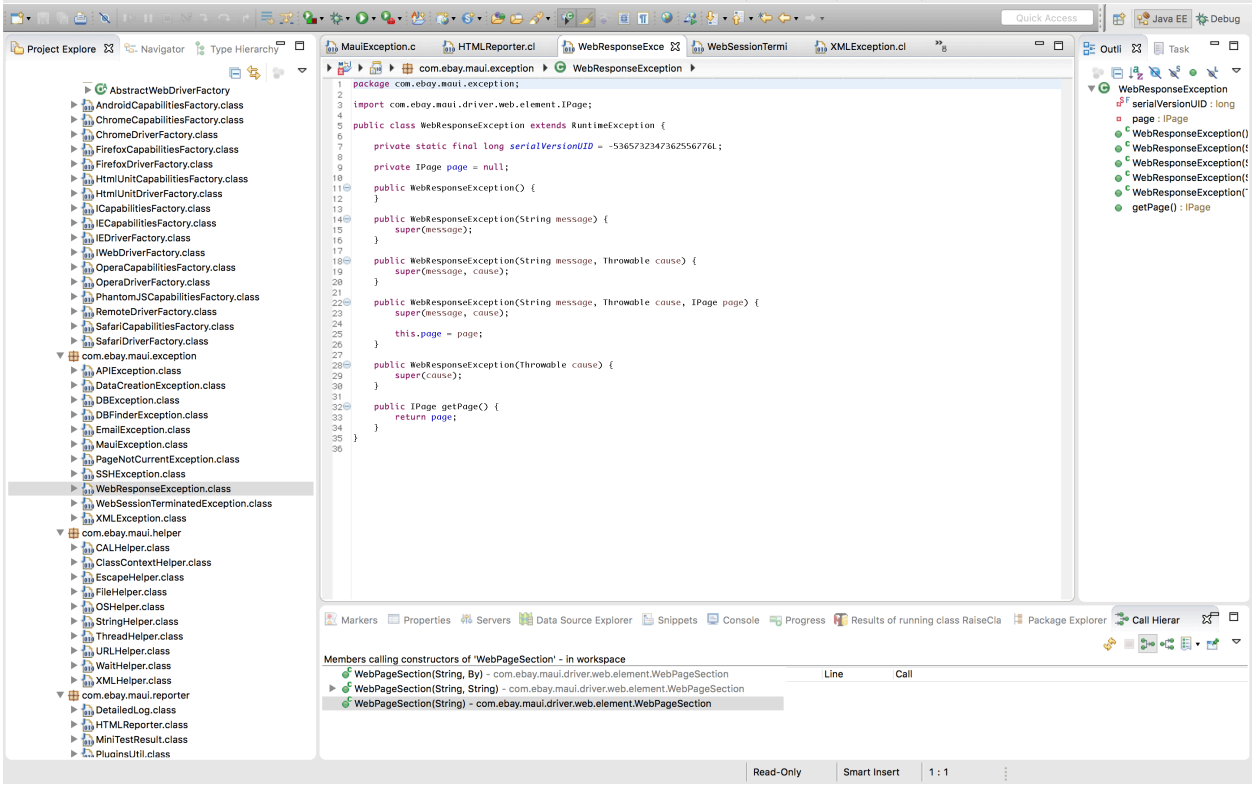
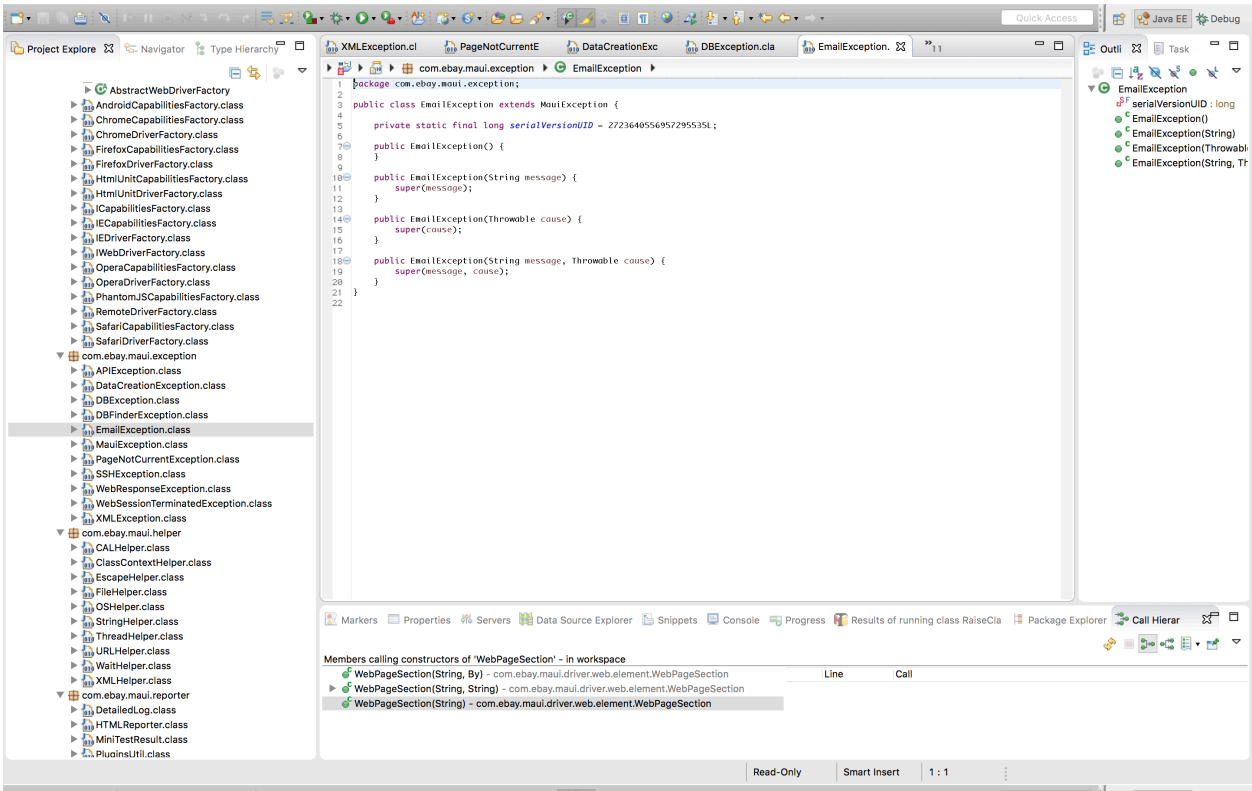
Exception:

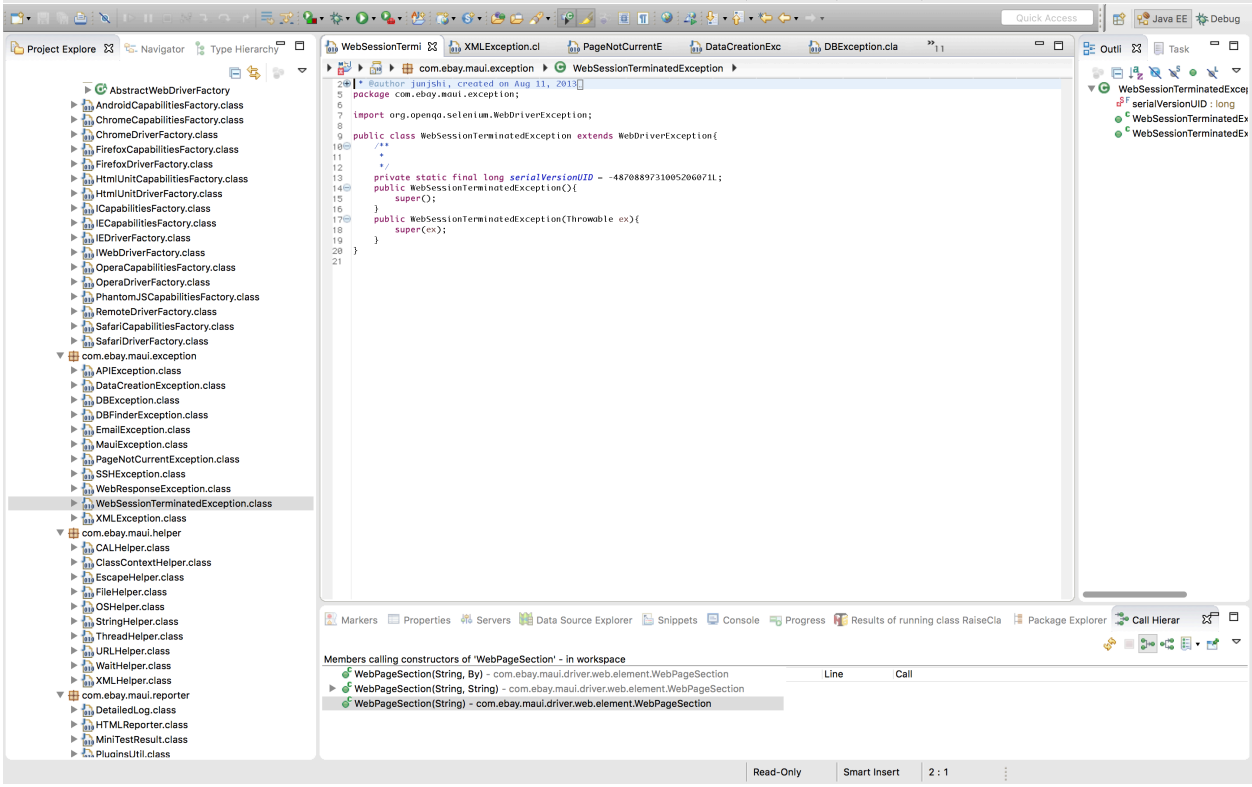

Project Explore
Navigator
Type Hierarchy
APIException.cl
MauiException.c
HTMLReporter.cl
WebResponseExce
WebSessionTermi
com.ebay.maui.exception
APIException
package com.ebay.maui.exception;
public class APIException extends RuntimeException {
private static final long serialVersionUID = -7173883151125380826L;
private int errorCode;
public APIException() {
}
public APIException(String message) {
super(message);
}
public APIException(Throwable cause) {
super(cause);
}
public APIException(String message, Throwable cause) {
super(message, cause);
}
public APIException(int errorCode,String message, Throwable cause) {
super(message, cause);
this.errorCode = errorCode;
}
public int getErrorCode(){
return this.errorCode;
}
}
Outli
Task
APIException
serialVersionUID : long
errorCode : int
APIException()
APIException(String)
APIException(Throwable)
APIException(String, Thro
APIException(int, String,
getErrorCode() : int
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
Line
Call
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
Read-Only
Smart Insert
1 : 1

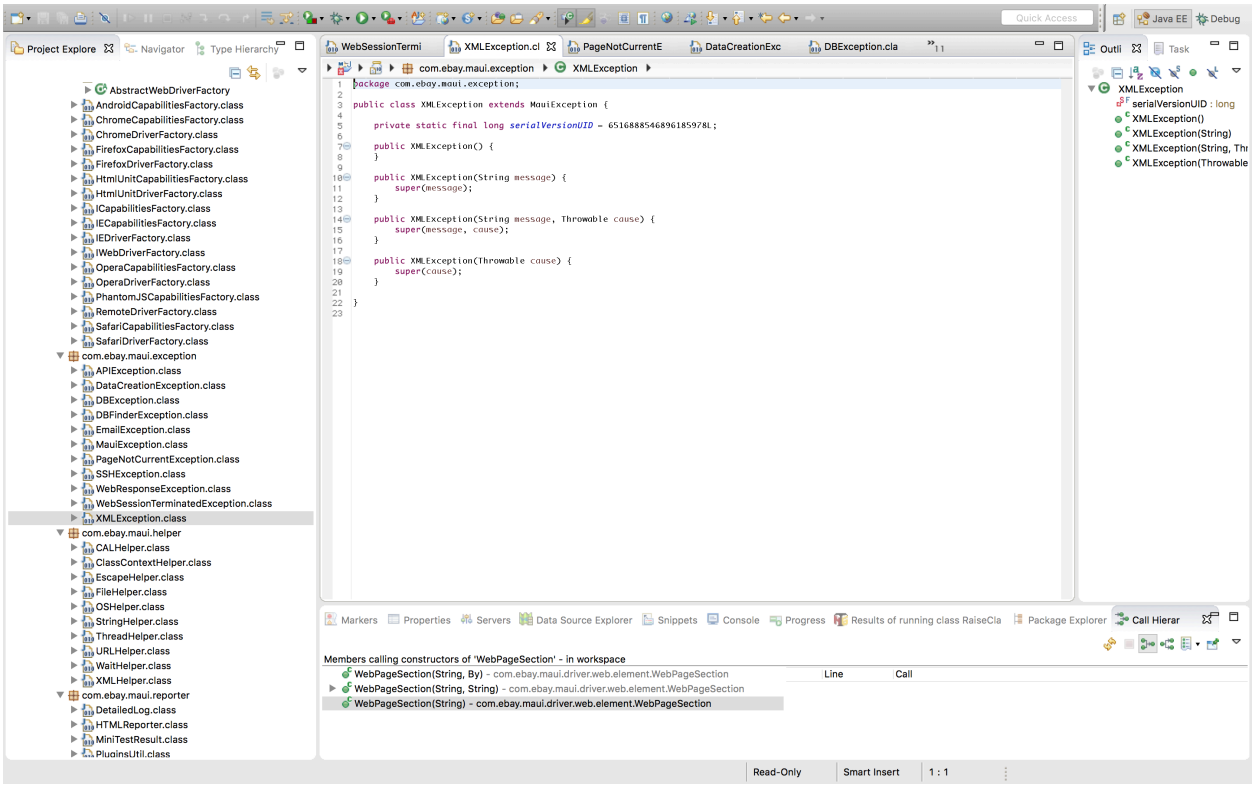

Project Explore
Navigator
Type Hierarchy
MauiException.c
HTMLReporter.cl
WebResponseExce
WebSessionTermi
XMLException.cl
com.ebay.maui.exception
MauiException
package com.ebay.maui.exception;
public class MauiException extends Exception {
private static final long serialVersionUID = -2757992149932989353L;
public MauiException() {
}
public MauiException(String message) {
super(message);
}
public MauiException(Throwable cause) {
super(cause);
}
public MauiException(String message, Throwable cause) {
super(message, cause);
}
}
Outli
Task
MauiException
serialVersionUID : long
MauiException()
MauiException(String)
MauiException(Throwable
MauiException(String, Th
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
Line
Call
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
Read-Only
Smart Insert
1 : 1


Project Explore
Navigator
Type Hierarchy
WebSessionTermi
XMLException.cl
PageNotCurrentE
DataCreationExc
DBException.cla
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.exception
DBException
package com.ebay.maui.exception;
public class DBException extends MauiException {
private static final long serialVersionUID = -8417471836653675063L;
public DBException() {
}
public DBException(String message) {
super(message);
}
public DBException(Throwable cause) {
super(cause);
}
public DBException(String message, Throwable cause) {
super(message, cause);
}
}
Outli
Task
DBException
serialVersionUID : long
DBException()
DBException(String)
DBException(Throwable)
DBException(String, Thro
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
Line
Call
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
Read-Only
Smart Insert
1 : 1

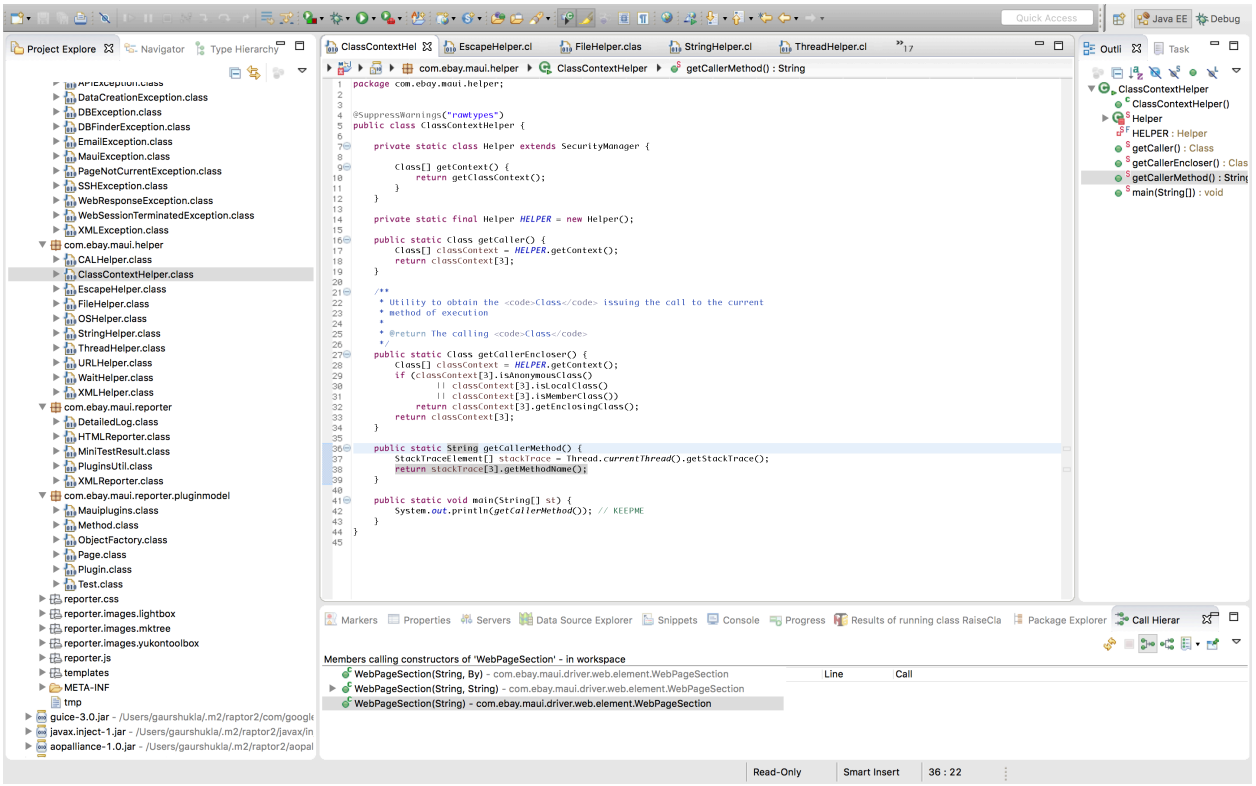

Project Explore
Navigator
Type Hierarchy
MauiException.c
HTMLReporter.cl
WebResponseExce
WebSessionTermi
XMLException.cl
com.ebay.maui.exception
WebResponseException
package com.ebay.maui.exception;
import com.ebay.maui.driver.web.element.IPage;
public class WebResponseException extends RuntimeException {
private static final long serialVersionUID = -5365732347362556776L;
private IPage page = null;
public WebResponseException() {
}
public WebResponseException(String message) {
super(message);
}
public WebResponseException(String message, Throwable cause) {
super(message, cause);
}
public WebResponseException(String message, Throwable cause, IPage page) {
super(message, cause);
this.page = page;
}
public WebResponseException(Throwable cause) {
super(cause);
}
public IPage getPage() {
return page;
}
}
Outli
Task
WebResponseException
serialVersionUID : long
page : IPage
WebResponseException()
getPage() : IPage
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
Line
Call
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
Read-Only
Smart Insert
1 : 1

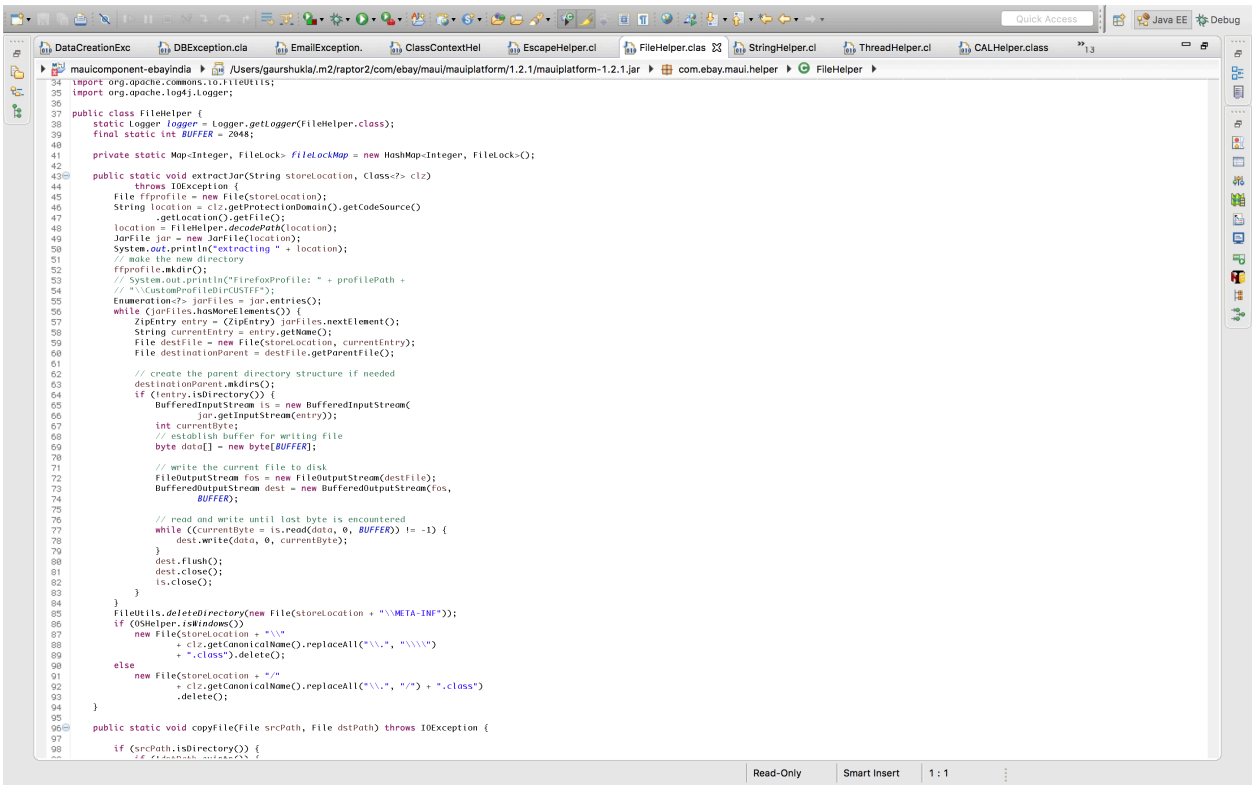

Project Explore
Navigator
Type Hierarchy
AbstractWebDriverFactory
AndroidCapabilitiesFactory.class
ChromeCapabilitiesFactory.class
ChromeDriverFactory.class
FirefoxCapabilitiesFactory.class
FirefoxDriverFactory.class
HtmlUnitCapabilitiesFactory.class
HtmlUnitDriverFactory.class
ICapabilitiesFactory.class
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
DBException.class
DBFinderException.class
EmailException.class
MauiException.class
PageNotCurrentException.class
SSHException.class
WebResponseException.class
WebSessionTerminatedException.class
XMLException.class
com.ebay.maui.helper
CALHelper.class
ClassContextHelper.class
EscapeHelper.class
FileHelper.class
OSHelper.class
StringHelper.class
ThreadHelper.class
URLHelper.class
WaitHelper.class
XMLHelper.class
com.ebay.maui.reporter
DetailedLog.class
HTMLReporter.class
MiniTestResult.class
PluginsUtil.class
XMLException.cl
PageNotCurrentE
DataCreationExc
DBException.cla
EmailException.
com.ebay.maui.exception
EmailException
package com.ebay.maui.exception;
public class EmailException extends MauiException {
private static final long serialVersionUID = 2723640556957955351;
public EmailException() {
}
public EmailException(String message) {
super(message);
}
public EmailException(Throwable cause) {
super(cause);
}
public EmailException(String message, Throwable cause) {
super(message, cause);
}
}
Quick Access
Java EE
Debug
Outli
Task
EmailException
serialVersionUID : long
EmailException()
EmailException(String)
EmailException(Throwabl
EmailException(String, Th
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
Line
Call
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
Read-Only
Smart Insert
1 : 1


Project Explore
Navigator
Type Hierarchy
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
DBException.class
DBFinderException.class
EmailException.class
MauiException.class
PageNotCurrentException.class
SSHException.class
WebResponseException.class
WebSessionTerminatedException.class
XMLException.class
com.ebay.maui.helper
CALHelper.class
ClassContextHelper.class
EscapeHelper.class
FileHelper.class
OSHelper.class
StringHelper.class
ThreadHelper.class
URLHelper.class
WaitHelper.class
XMLHelper.class
com.ebay.maui.reporter
DetailedLog.class
HTMLReporter.class
MiniTestResult.class
PluginsUtil.class
WebSessionTermi
XMLException.cl
PageNotCurrentE
DataCreationExc
DBException.cla
com.ebay.maui.exception
WebSessionTerminatedException
* @author junjshi, created on Aug 11, 2013
package com.ebay.maui.exception;
import org.openqa.selenium.WebDriverException;
public class WebSessionTerminatedException extends WebDriverException{
/**
*
*/
private static final long serialVersionUID = -4870889731005206071L;
public WebSessionTerminatedException(){
super();
}
public WebSessionTerminatedException(Throwable ex){
super(ex);
}
}
Quick Access
Java EE
Debug
Outli
Task
WebSessionTerminatedExce
serialVersionUID : long
WebSessionTerminatedEx
WebSessionTerminatedEx
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
Line
Call
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
Read-Only
Smart Insert
2 : 1

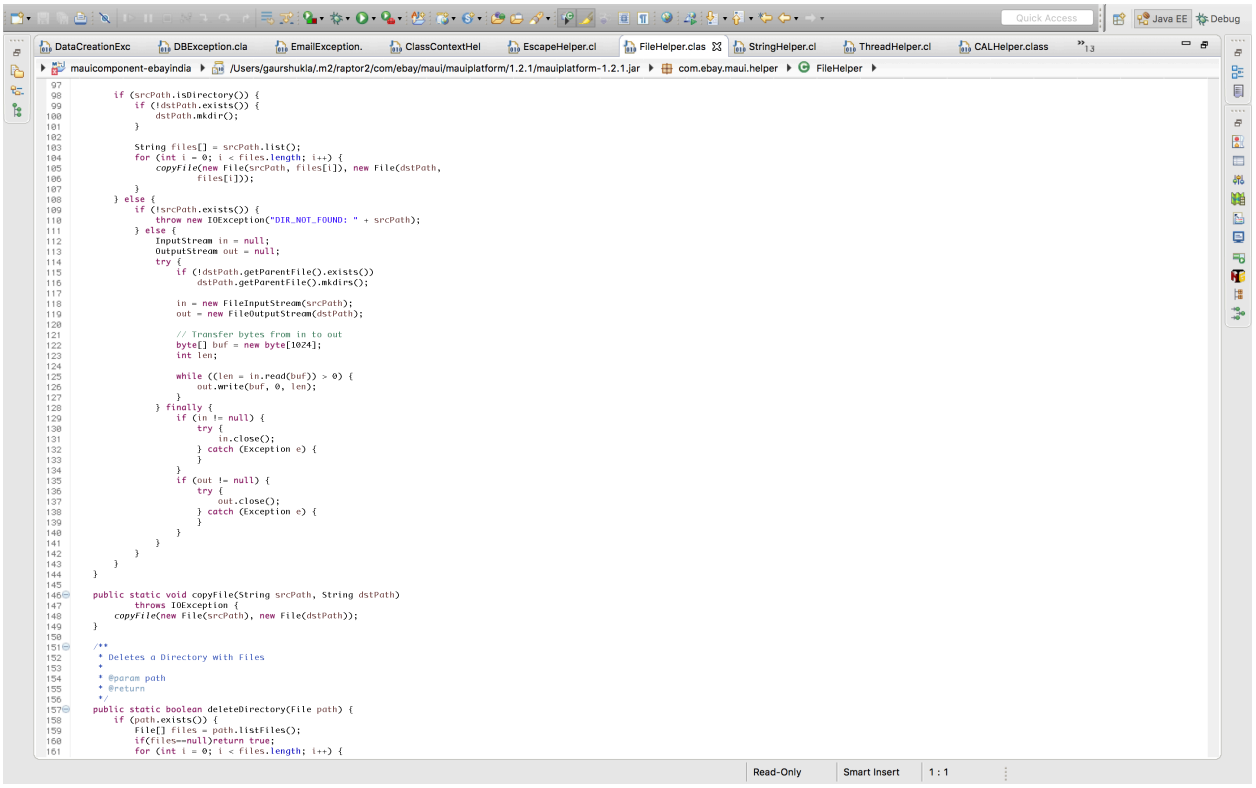

Project Explore
Navigator
Type Hierarchy
WebSessionTermi
XMLException.cl
PageNotCurrentE
DataCreationExc
DBException.cla
com.ebay.maui.exception
XMLException
package com.ebay.maui.exception;
public class XMLException extends MauiException {
private static final long serialVersionUID = 6516888546896185978L;
public XMLException() {
}
public XMLException(String message) {
super(message);
}
public XMLException(String message, Throwable cause) {
super(message, cause);
}
public XMLException(Throwable cause) {
super(cause);
}
}
Outli
Task
XMLException
serialVersionUID : long
XMLException()
XMLException(String)
XMLException(String, Thr
XMLException(Throwable
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

Helper

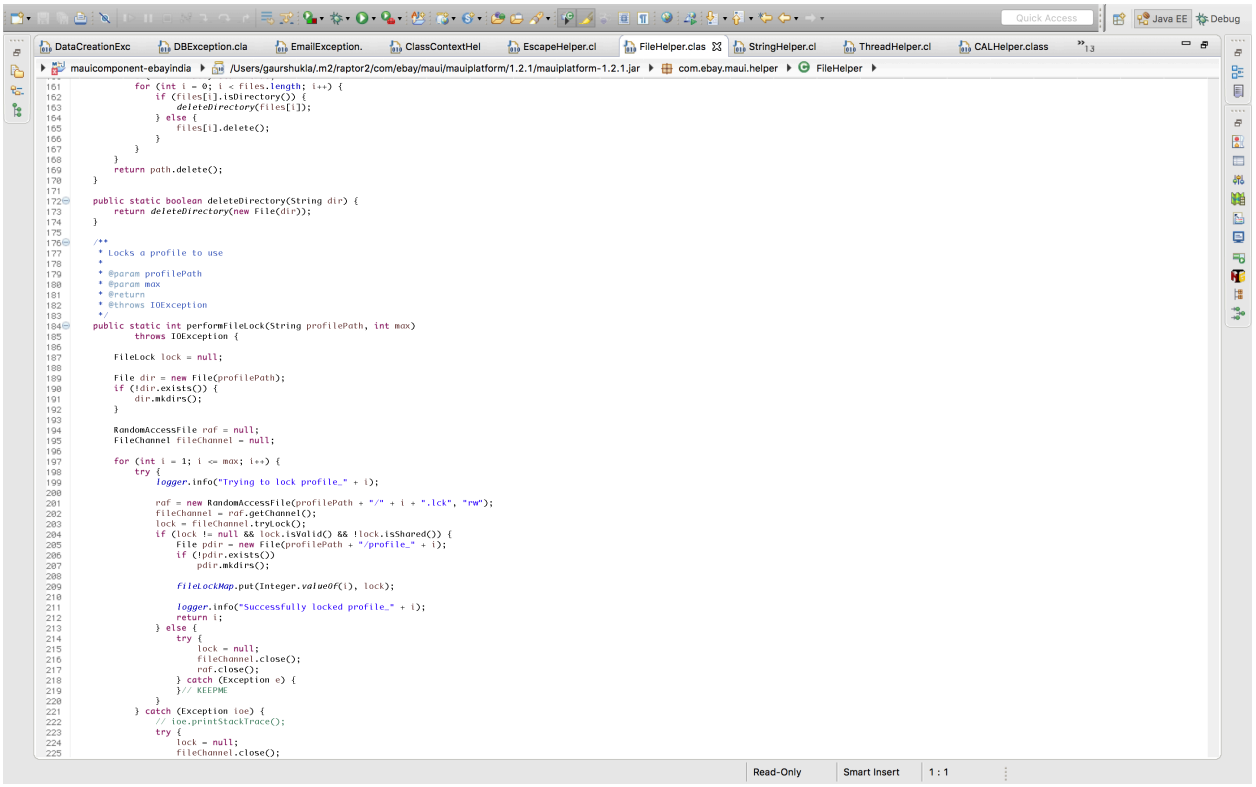

Project Explore
Navigator
Type Hierarchy
ClassContextHel
EscapeHelper.cl
FileHelper.clas
StringHelper.cl
ThreadHelper.cl
com.ebay.maui.helper
ClassContextHelper
getCallerMethod() : String
package com.ebay.maui.helper;
@SuppressWarnings("rawtypes")
public class ClassContextHelper {
private static class Helper extends SecurityManager {
Class[] getContext() {
return getClassContext();
}
}
private static final Helper HELPER = new Helper();
public static Class getCaller() {
Class[] classContext = HELPER.getContext();
return classContext[3];
}
/**
* Utility to obtain the <code>Class</code> issuing the call to the current
* method of execution
*
* @return The calling <code>Class</code>
*/
public static Class getCallerEncloser() {
Class[] classContext = HELPER.getContext();
if (classContext[3].isAnonymousClass()
|| classContext[3].isLocalClass()
|| classContext[3].isMemberClass())
return classContext[3].getEnclosingClass();
return classContext[3];
}
public static String getCallerMethod() {
StackTraceElement[] stackTrace = Thread.currentThread().getStackTrace();
return stackTrace[3].getMethodName();
}
public static void main(String[] st) {
System.out.println(getCallerMethod()); // KEEPME
}
}
Outli
Task
ClassContextHelper
ClassContextHelper()
Helper
HELPER : Helper
getCaller() : Class
getCallerEncloser() : Clas
getCallerMethod() : Strin
main(String[]) : void
DataCreationException.class
DBException.class
DBFinderException.class
EmailException.class
MauiException.class
PageNotCurrentException.class
SSHException.class
WebResponseException.class
WebSessionTerminatedException.class
XMLException.class
CALHelper.class
ClassContextHelper.class
EscapeHelper.class
FileHelper.class
OSHelper.class
StringHelper.class
ThreadHelper.class
URLHelper.class
WaitHelper.class
XMLHelper.class
com.ebay.maui.reporter
DetailedLog.class
HTMLReporter.class
MiniTestResult.class
PluginsUtil.class
XMLReporter.class
com.ebay.maui.reporter.pluginmodel
Mauiplugins.class
Method.class
ObjectFactory.class
Page.class
Plugin.class
Test.class
reporter.css
reporter.images.lightbox
reporter.images.mktree
reporter.images.yukontoolbox
reporter.js
templates
META-INF
tmp
guice-3.0.jar - /Users/gaurshukla/.m2/raptor2/com/google
javax.inject-1.jar - /Users/gaurshukla/.m2/raptor2/javax/in
aopalliance-1.0.jar - /Users/gaurshukla/.m2/raptor2/aopal
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
36 : 22



```
import org.apache.commons.io.FileUtils;
import org.apache.log4j.Logger;

public class FileHelper {
    static Logger logger = Logger.getLogger(FileHelper.class);
    final static int BUFFER = 2048;

    private static Map<Integer, FileLock> fileLockMap = new HashMap<Integer, FileLock>();

    public static void extractJar(String storeLocation, Class<?> clz)
            throws IOException {
        File ffprofile = new File(storeLocation);
        String location = clz.getProtectionDomain().getCodeSource()
                .getLocation().getFile();
        location = FileHelper.decodePath(location);
        JarFile jar = new JarFile(location);
        System.out.println("extracting " + location);
        // make the new directory
        ffprofile.mkdir();
        // System.out.println("FirefoxProfile: " + profilePath +
        // "\\CustomProfileDirCUSTFF");
        Enumeration<?> jarFiles = jar.entries();
        while (jarFiles.hasMoreElements()) {
            ZipEntry entry = (ZipEntry) jarFiles.nextElement();
            String currentEntry = entry.getName();
            File destFile = new File(storeLocation, currentEntry);
            File destinationParent = destFile.getParentFile();

            // create the parent directory structure if needed
            destinationParent.mkdirs();
            if (!entry.isDirectory()) {
                BufferedInputStream is = new BufferedInputStream(
                        jar.getInputStream(entry));
                int currentByte;
                // establish buffer for writing file
                byte data[] = new byte[BUFFER];

                // write the current file to disk
                FileOutputStream fos = new FileOutputStream(destFile);
                BufferedOutputStream dest = new BufferedOutputStream(fos,
                        BUFFER);

                // read and write until last byte is encountered
                while ((currentByte = is.read(data, 0, BUFFER)) != -1) {
                    dest.write(data, 0, currentByte);
                }
                dest.flush();
                dest.close();
                is.close();
            }
        }
        FileUtils.deleteDirectory(new File(storeLocation + "\\META-INF"));
        if (OSHelper.isWindows())
            new File(storeLocation + "\\"
                    + clz.getCanonicalName().replaceAll("\\.", "\\\\")
                    + ".class").delete();
        else
            new File(storeLocation + "/"
                    + clz.getCanonicalName().replaceAll("\\.", "/") + ".class")
                    .delete();
    }

    public static void copyFile(File srcPath, File dstPath) throws IOException {

        if (srcPath.isDirectory()) {
```

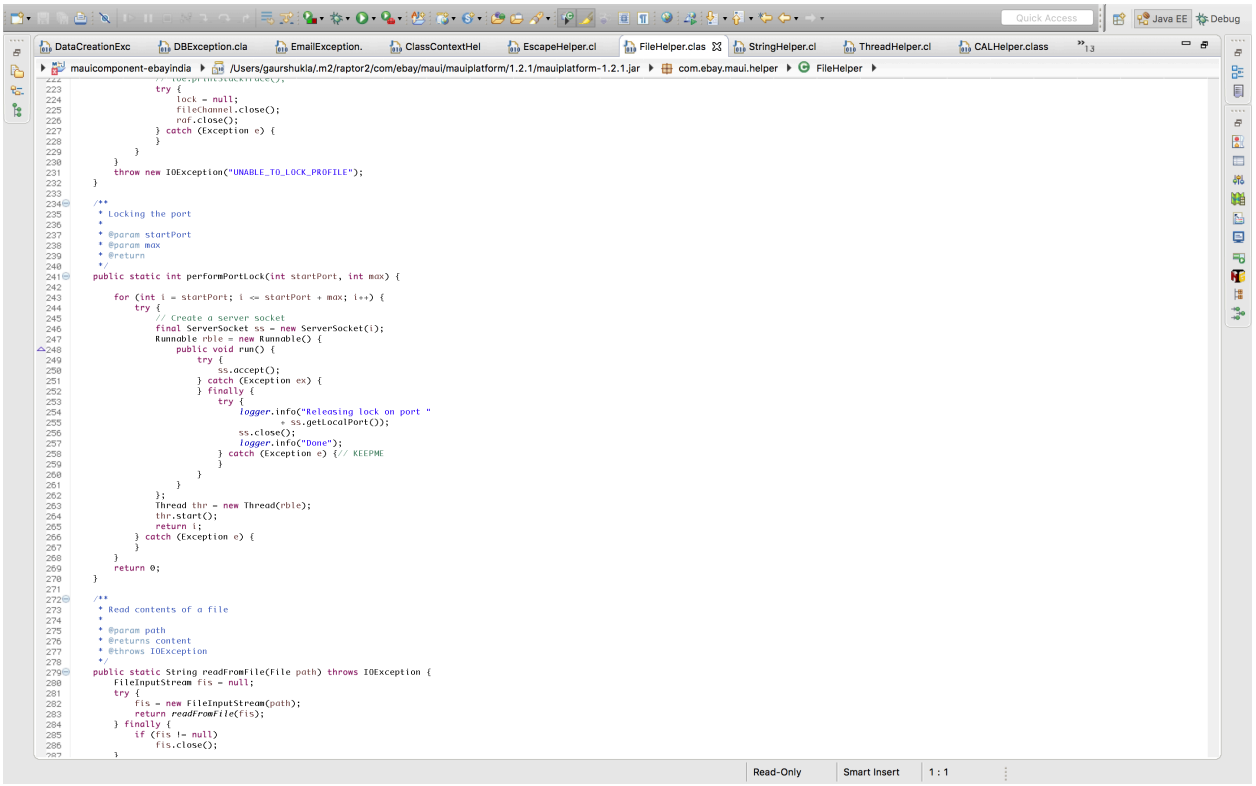


```
        if (srcPath.isDirectory()) {
            if (!dstPath.exists()) {
                dstPath.mkdir();
            }

            String files[] = srcPath.list();
            for (int i = 0; i < files.length; i++) {
                copyFile(new File(srcPath, files[i]), new File(dstPath,
                        files[i]));
            }
        } else {
            if (!srcPath.exists()) {
                throw new IOException("DIR_NOT_FOUND: " + srcPath);
            } else {
                InputStream in = null;
                OutputStream out = null;
                try {
                    if (!dstPath.getParentFile().exists())
                        dstPath.getParentFile().mkdirs();

                    in = new FileInputStream(srcPath);
                    out = new FileOutputStream(dstPath);

                    // Transfer bytes from in to out
                    byte[] buf = new byte[1024];
                    int len;

                    while ((len = in.read(buf)) > 0) {
                        out.write(buf, 0, len);
                    }
                } finally {
                    if (in != null) {
                        try {
                            in.close();
                        } catch (Exception e) {
                        }
                    }
                    if (out != null) {
                        try {
                            out.close();
                        } catch (Exception e) {
                        }
                    }
                }
            }
        }
    }

    public static void copyFile(String srcPath, String dstPath)
            throws IOException {
        copyFile(new File(srcPath), new File(dstPath));
    }

    /**
     * Deletes a Directory with Files
     *
     * @param path
     * @return
     */
    public static boolean deleteDirectory(File path) {
        if (path.exists()) {
            File[] files = path.listFiles();
            if(files==null)return true;
            for (int i = 0; i < files.length; i++) {
```



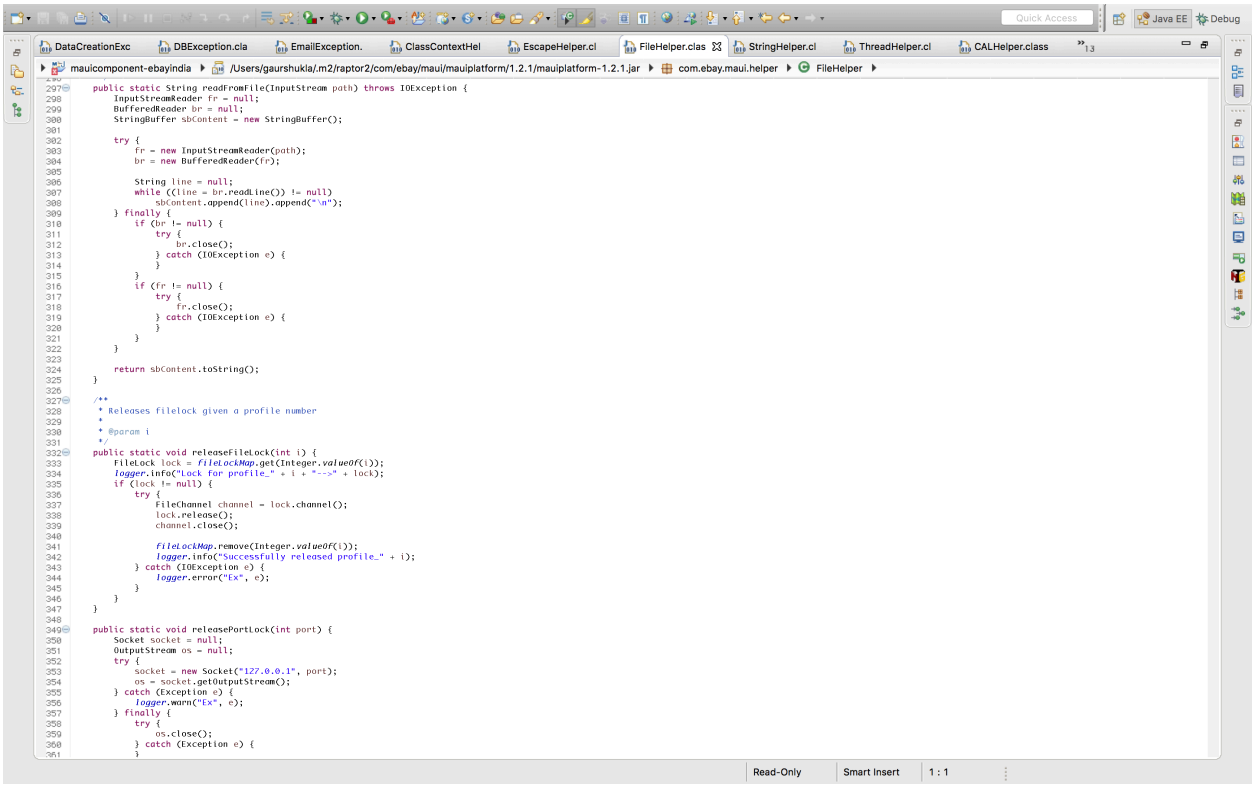
```
            for (int i = 0; i < files.length; i++) {
                if (files[i].isDirectory()) {
                    deleteDirectory(files[i]);
                } else {
                    files[i].delete();
                }
            }
        }
        return path.delete();
    }

    public static boolean deleteDirectory(String dir) {
        return deleteDirectory(new File(dir));
    }

    /**
     * Locks a profile to use
     *
     * @param profilePath
     * @param max
     * @return
     * @throws IOException
     */
    public static int performFileLock(String profilePath, int max)
            throws IOException {

        FileLock lock = null;

        File dir = new File(profilePath);
        if (!dir.exists()) {
            dir.mkdirs();
        }

        RandomAccessFile raf = null;
        FileChannel fileChannel = null;

        for (int i = 1; i <= max; i++) {
            try {
                logger.info("Trying to lock profile_" + i);

                raf = new RandomAccessFile(profilePath + "/" + i + ".lck", "rw");
                fileChannel = raf.getChannel();
                lock = fileChannel.tryLock();
                if (lock != null && lock.isValid() && !lock.isShared()) {
                    File pdir = new File(profilePath + "/profile_" + i);
                    if (!pdir.exists())
                        pdir.mkdirs();

                    fileLockMap.put(Integer.valueOf(i), lock);

                    logger.info("Successfully locked profile_" + i);
                    return i;
                } else {
                    try {
                        lock = null;
                        fileChannel.close();
                        raf.close();
                    } catch (Exception e) {
                    }// KEEPME
                }
            } catch (Exception ioe) {
                // ioe.printStackTrace();
                try {
                    lock = null;
                    fileChannel.close();
```

Read-Only | Smart Insert | 1 : 1

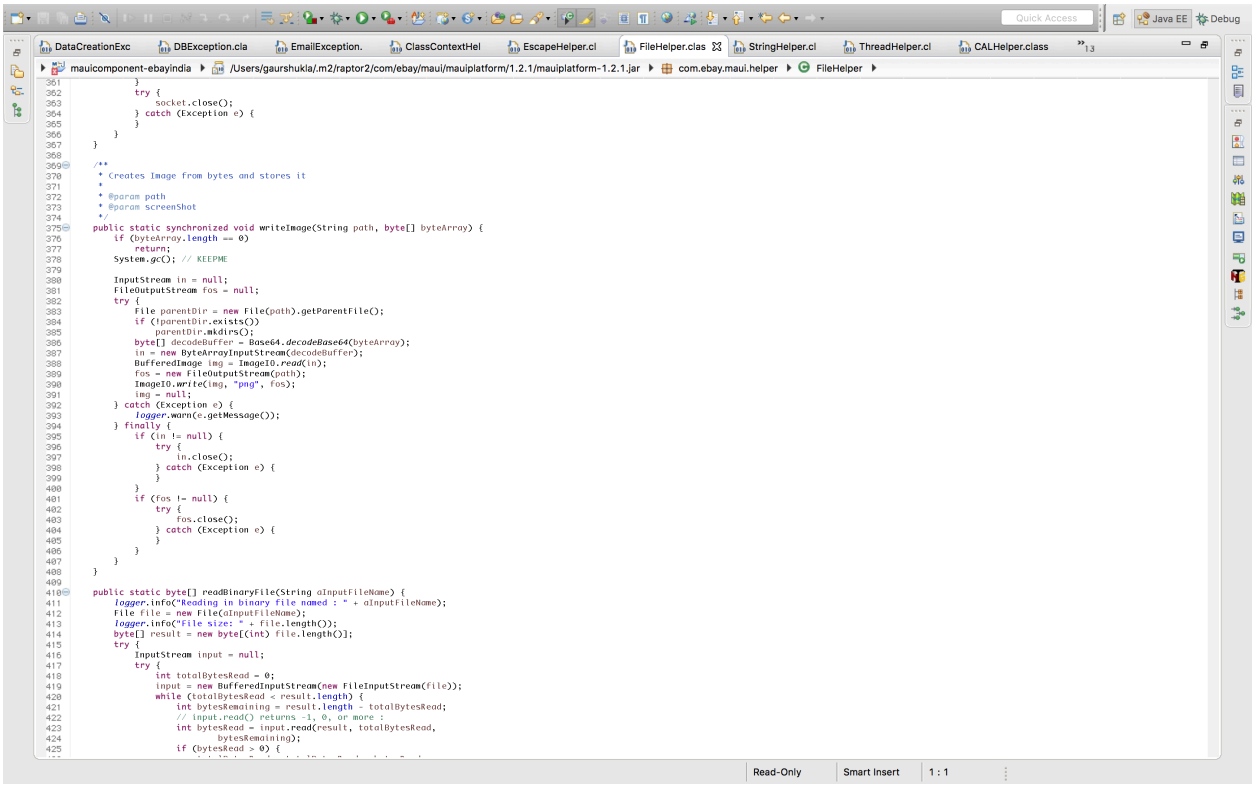


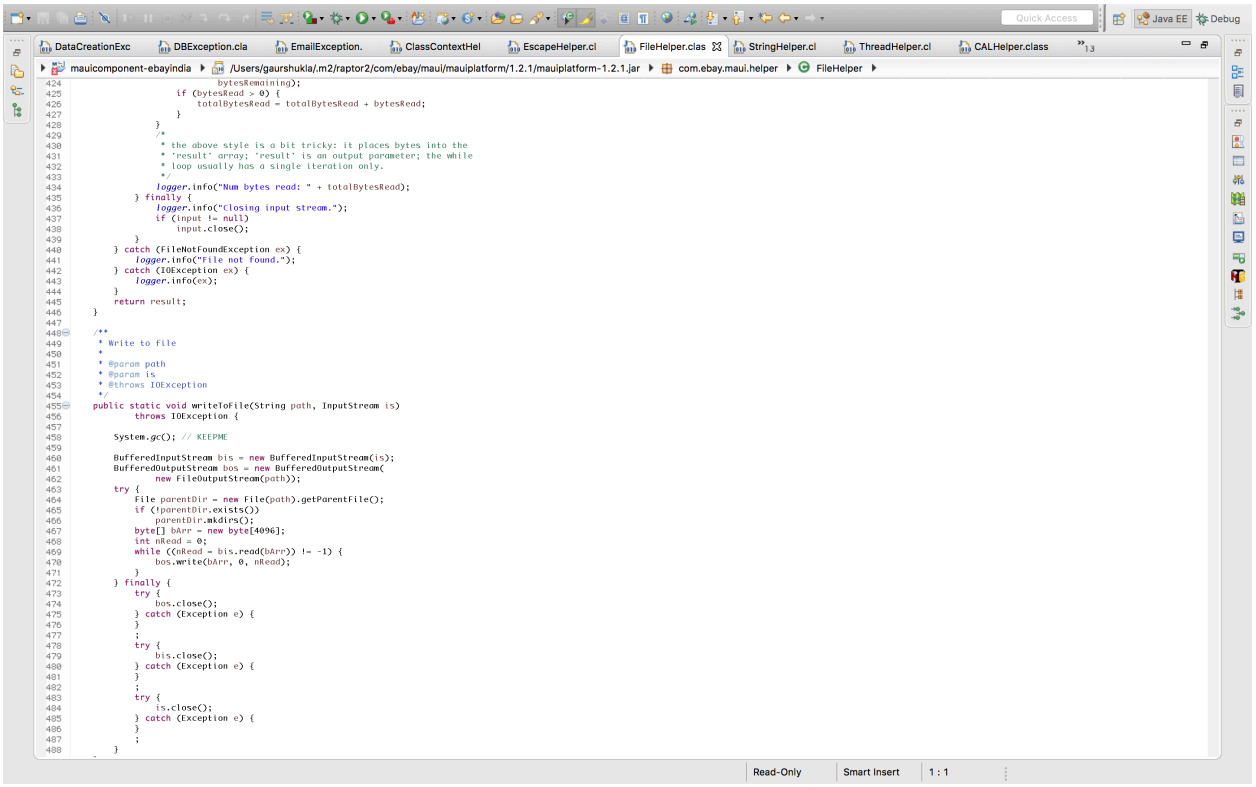
```
                    lock = null;
                    fileChannel.close();
                    raf.close();
                } catch (Exception e) {
                }
            }
        }
        throw new IOException("UNABLE_TO_LOCK_PROFILE");
    }

    /**
     * Locking the port
     *
     * @param startPort
     * @param max
     * @return
     */
    public static int performPortLock(int startPort, int max) {

        for (int i = startPort; i <= startPort + max; i++) {
            try {
                // Create a server socket
                final ServerSocket ss = new ServerSocket(i);
                Runnable rble = new Runnable() {
                    public void run() {
                        try {
                            ss.accept();
                        } catch (Exception ex) {
                        } finally {
                            try {
                                logger.info("Releasing lock on port "
                                        + ss.getLocalPort());
                                ss.close();
                                logger.info("Done");
                            } catch (Exception e) {// KEEPME
                            }
                        }
                    }
                };
                Thread thr = new Thread(rble);
                thr.start();
                return i;
            } catch (Exception e) {
            }
        }
        return 0;
    }

    /**
     * Read contents of a file
     *
     * @param path
     * @returns content
     * @throws IOException
     */
    public static String readFromFile(File path) throws IOException {
        FileInputStream fis = null;
        try {
            fis = new FileInputStream(path);
            return readFromFile(fis);
        } finally {
            if (fis != null)
                fis.close();
```

Read-Only | Smart Insert | 1 : 1

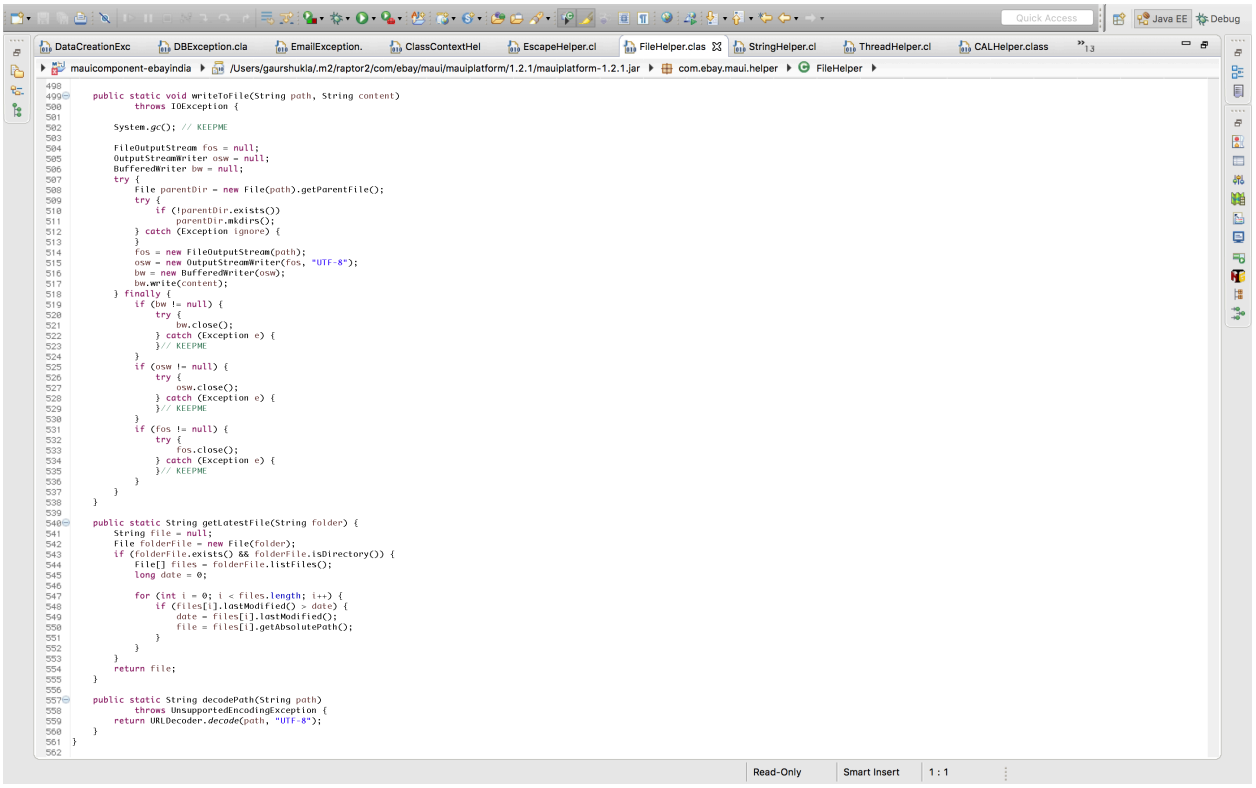


```
    public static String readFromFile(InputStream path) throws IOException {
        InputStreamReader fr = null;
        BufferedReader br = null;
        StringBuffer sbContent = new StringBuffer();

        try {
            fr = new InputStreamReader(path);
            br = new BufferedReader(fr);

            String line = null;
            while ((line = br.readLine()) != null)
                sbContent.append(line).append("\n");
        } finally {
            if (br != null) {
                try {
                    br.close();
                } catch (IOException e) {
                }
            }
            if (fr != null) {
                try {
                    fr.close();
                } catch (IOException e) {
                }
            }
        }

        return sbContent.toString();
    }

    /**
     * Releases filelock given a profile number
     *
     * @param i
     */
    public static void releaseFileLock(int i) {
        FileLock lock = fileLockMap.get(Integer.valueOf(i));
        logger.info("Lock for profile_" + i + "-->" + lock);
        if (lock != null) {
            try {
                FileChannel channel = lock.channel();
                lock.release();
                channel.close();

                fileLockMap.remove(Integer.valueOf(i));
                logger.info("Successfully released profile_" + i);
            } catch (IOException e) {
                logger.error("Ex", e);
            }
        }
    }

    public static void releasePortLock(int port) {
        Socket socket = null;
        OutputStream os = null;
        try {
            socket = new Socket("127.0.0.1", port);
            os = socket.getOutputStream();
        } catch (Exception e) {
            logger.warn("Ex", e);
        } finally {
            try {
                os.close();
            } catch (Exception e) {
            }
            try {
                socket.close();
            } catch (Exception e) {
            }
        }
    }

    /**
     * Creates Image from bytes and stores it
     *
     * @param path
     * @param screenShot
     */
    public static synchronized void writeImage(String path, byte[] byteArray) {
        if (byteArray.length == 0)
            return;
        System.gc(); // KEEPME

        InputStream in = null;
        FileOutputStream fos = null;
        try {
            File parentDir = new File(path).getParentFile();
            if (!parentDir.exists())
                parentDir.mkdirs();
            byte[] decodeBuffer = Base64.decodeBase64(byteArray);
            in = new ByteArrayInputStream(decodeBuffer);
            BufferedImage img = ImageIO.read(in);
            fos = new FileOutputStream(path);
            ImageIO.write(img, "png", fos);
            img = null;
        } catch (Exception e) {
            logger.warn(e.getMessage());
        } finally {
            if (in != null) {
                try {
                    in.close();
                } catch (Exception e) {
                }
            }
            if (fos != null) {
                try {
                    fos.close();
                } catch (Exception e) {
                }
            }
        }
    }

    public static byte[] readBinaryFile(String aInputFileName) {
        logger.info("Reading in binary file named : " + aInputFileName);
        File file = new File(aInputFileName);
        logger.info("File size: " + file.length());
        byte[] result = new byte[(int) file.length()];
        try {
            InputStream input = null;
            try {
                int totalBytesRead = 0;
                input = new BufferedInputStream(new FileInputStream(file));
                while (totalBytesRead < result.length) {
                    int bytesRemaining = result.length - totalBytesRead;
                    // input.read() returns -1, 0, or more :
                    int bytesRead = input.read(result, totalBytesRead,
                            bytesRemaining);
                    if (bytesRead > 0) {
```

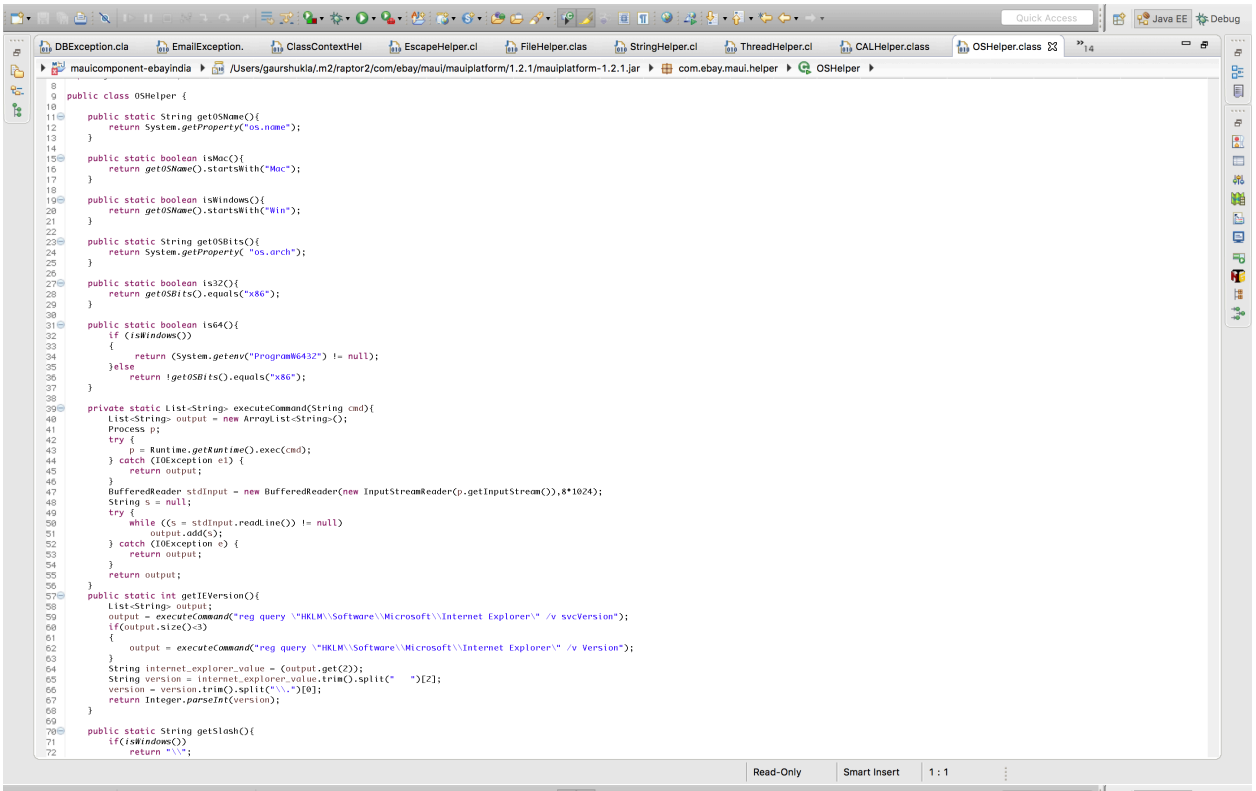




```
                        bytesRemaining);
                if (bytesRead > 0) {
                    totalBytesRead = totalBytesRead + bytesRead;
                }
            }
            /*
             * the above style is a bit tricky: it places bytes into the
             * 'result' array; 'result' is an output parameter; the while
             * loop usually has a single iteration only.
             */
            logger.info("Num bytes read: " + totalBytesRead);
        } finally {
            logger.info("Closing input stream.");
            if (input != null)
                input.close();
        }
    } catch (FileNotFoundException ex) {
        logger.info("File not found.");
    } catch (IOException ex) {
        logger.info(ex);
    }
    return result;
}

/**
 * Write to file
 *
 * @param path
 * @param is
 * @throws IOException
 */
public static void writeToFile(String path, InputStream is)
        throws IOException {

    System.gc(); // KEEPME

    BufferedInputStream bis = new BufferedInputStream(is);
    BufferedOutputStream bos = new BufferedOutputStream(
            new FileOutputStream(path));
    try {
        File parentDir = new File(path).getParentFile();
        if (!parentDir.exists())
            parentDir.mkdirs();
        byte[] bArr = new byte[4096];
        int nRead = 0;
        while ((nRead = bis.read(bArr)) != -1) {
            bos.write(bArr, 0, nRead);
        }
    } finally {
        try {
            bos.close();
        } catch (Exception e) {
        }
        ;
        try {
            bis.close();
        } catch (Exception e) {
        }
        ;
        try {
            is.close();
        } catch (Exception e) {
        }
        ;
    }
```

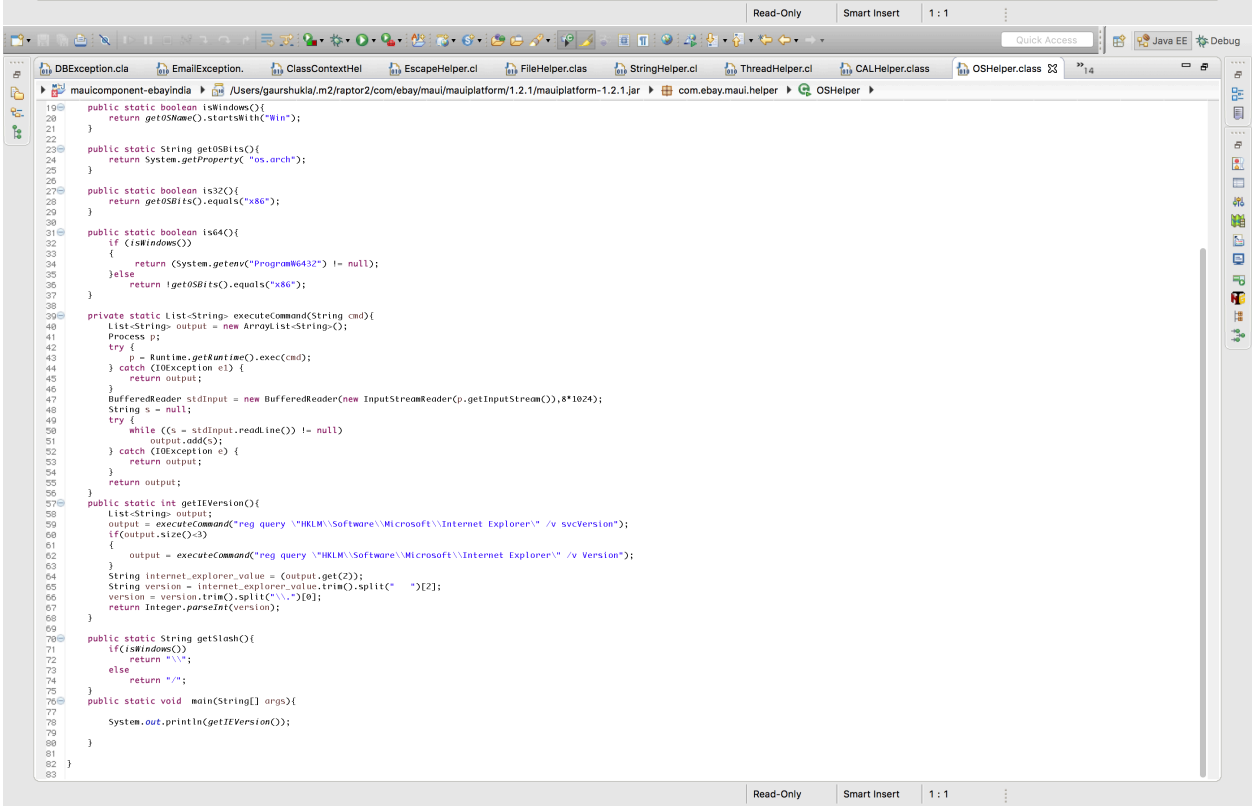


```
public static void writeToFile(String path, String content)
        throws IOException {

    System.gc(); // KEEPME

    FileOutputStream fos = null;
    OutputStreamWriter osw = null;
    BufferedWriter bw = null;
    try {
        File parentDir = new File(path).getParentFile();
        try {
            if (!parentDir.exists())
                parentDir.mkdirs();
        } catch (Exception ignore) {
        }
        fos = new FileOutputStream(path);
        osw = new OutputStreamWriter(fos, "UTF-8");
        bw = new BufferedWriter(osw);
        bw.write(content);
    } finally {
        if (bw != null) {
            try {
                bw.close();
            } catch (Exception e) {
            }// KEEPME
        }
        if (osw != null) {
            try {
                osw.close();
            } catch (Exception e) {
            }// KEEPME
        }
        if (fos != null) {
            try {
                fos.close();
            } catch (Exception e) {
            }// KEEPME
        }
    }
}

public static String getLatestFile(String folder) {
    String file = null;
    File folderFile = new File(folder);
    if (folderFile.exists() && folderFile.isDirectory()) {
        File[] files = folderFile.listFiles();
        long date = 0;

        for (int i = 0; i < files.length; i++) {
            if (files[i].lastModified() > date) {
                date = files[i].lastModified();
                file = files[i].getAbsolutePath();
            }
        }
    }
    return file;
}

public static String decodePath(String path)
        throws UnsupportedEncodingException {
    return URLDecoder.decode(path, "UTF-8");
}
}
```



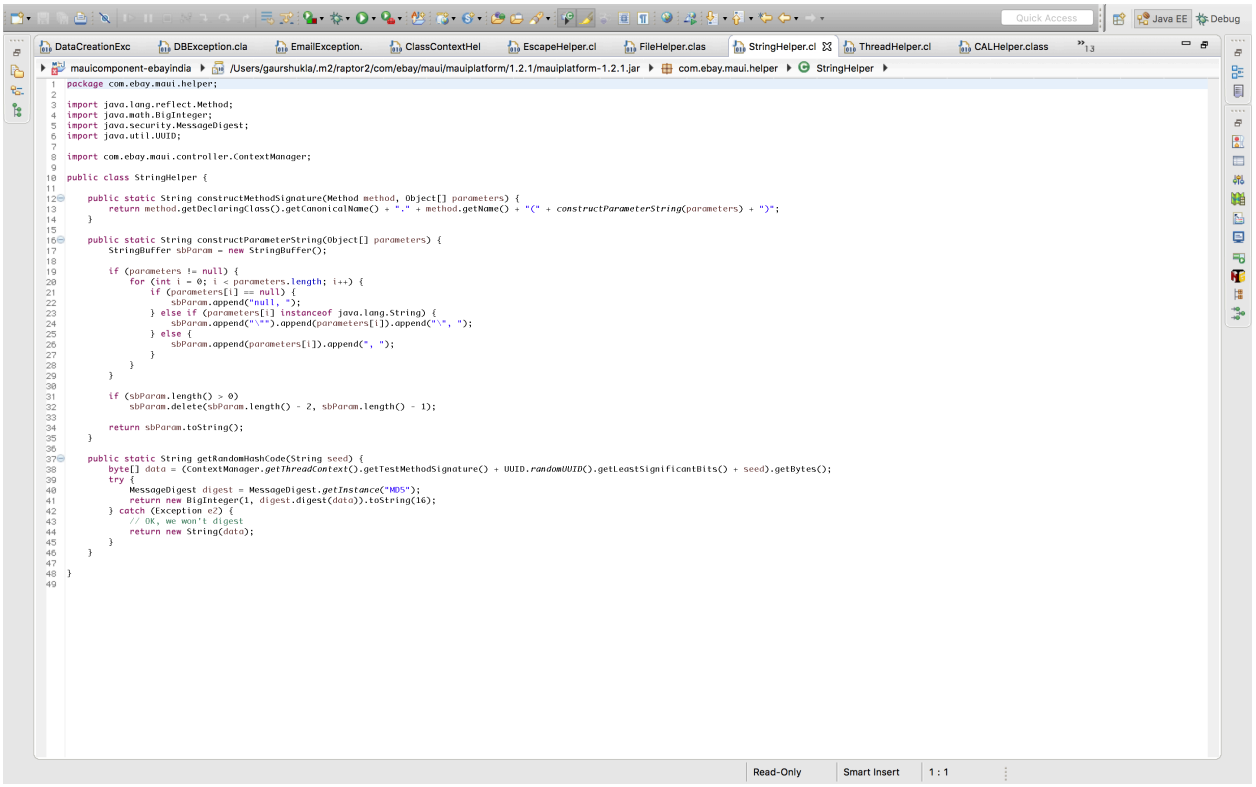
```
public class OSHelper {

    public static String getOSName(){
        return System.getProperty("os.name");
    }

    public static boolean isMac(){
        return getOSName().startsWith("Mac");
    }

    public static boolean isWindows(){
        return getOSName().startsWith("Win");
    }

    public static String getOSBits(){
        return System.getProperty( "os.arch");
    }

    public static boolean is32(){
        return getOSBits().equals("x86");
    }

    public static boolean is64(){
        if (isWindows())
        {
            return (System.getenv("ProgramW6432") != null);
        }else
            return !getOSBits().equals("x86");
    }

    private static List<String> executeCommand(String cmd){
        List<String> output = new ArrayList<String>();
        Process p;
        try {
            p = Runtime.getRuntime().exec(cmd);
        } catch (IOException e1) {
            return output;
        }
        BufferedReader stdInput = new BufferedReader(new InputStreamReader(p.getInputStream()),8*1024);
        String s = null;
        try {
            while ((s = stdInput.readLine()) != null)
                output.add(s);
        } catch (IOException e) {
            return output;
        }
        return output;
    }
    public static int getIEVersion(){
        List<String> output;
        output = executeCommand("reg query \"HKLM\\Software\\Microsoft\\Internet Explorer\" /v svcVersion");
        if(output.size()<3)
        {
            output = executeCommand("reg query \"HKLM\\Software\\Microsoft\\Internet Explorer\" /v Version");
        }
        String internet_explorer_value = (output.get(2));
        String version = internet_explorer_value.trim().split("    ")[2];
        version = version.trim().split("\\.")[0];
        return Integer.parseInt(version);
    }

    public static String getSlash(){
        if(isWindows())
            return "\\";
```

Read-Only | Smart Insert | 1 : 1

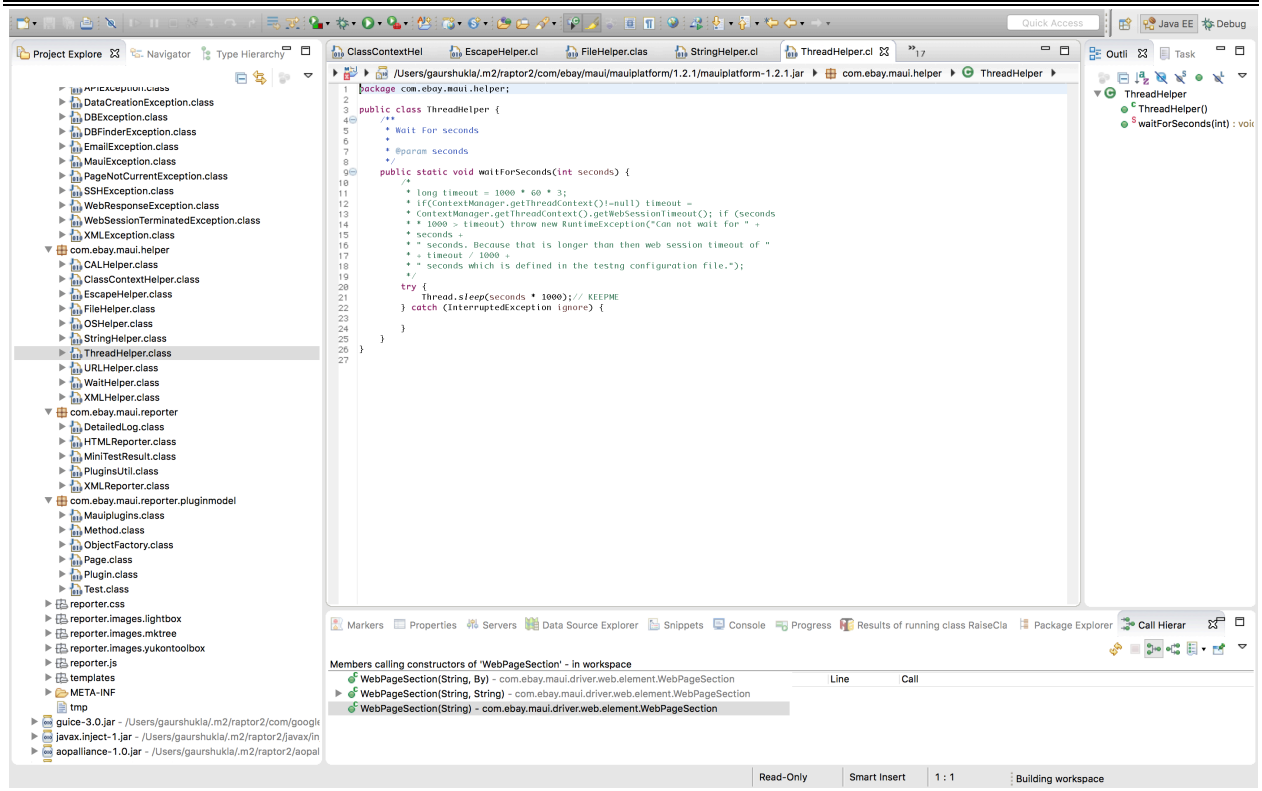


```
    public static boolean isWindows(){
        return getOSName().startsWith("Win");
    }

    public static String getOSBits(){
        return System.getProperty( "os.arch");
    }

    public static boolean is32(){
        return getOSBits().equals("x86");
    }

    public static boolean is64(){
        if (isWindows())
        {
            return (System.getenv("ProgramW6432") != null);
        }else
            return !getOSBits().equals("x86");
    }

    private static List<String> executeCommand(String cmd){
        List<String> output = new ArrayList<String>();
        Process p;
        try {
            p = Runtime.getRuntime().exec(cmd);
        } catch (IOException e1) {
            return output;
        }
        BufferedReader stdInput = new BufferedReader(new InputStreamReader(p.getInputStream()),8*1024);
        String s = null;
        try {
            while ((s = stdInput.readLine()) != null)
                output.add(s);
        } catch (IOException e) {
            return output;
        }
        return output;
    }
    public static int getIEVersion(){
        List<String> output;
        output = executeCommand("reg query \"HKLM\\Software\\Microsoft\\Internet Explorer\" /v svcVersion");
        if(output.size()<3)
        {
            output = executeCommand("reg query \"HKLM\\Software\\Microsoft\\Internet Explorer\" /v Version");
        }
        String internet_explorer_value = (output.get(2));
        String version = internet_explorer_value.trim().split("    ")[2];
        version = version.trim().split("\\.")[0];
        return Integer.parseInt(version);
    }

    public static String getSlash(){
        if(isWindows())
            return "\\";
        else
            return "/";
    }
    public static void  main(String[] args){

        System.out.println(getIEVersion());

    }
}
```

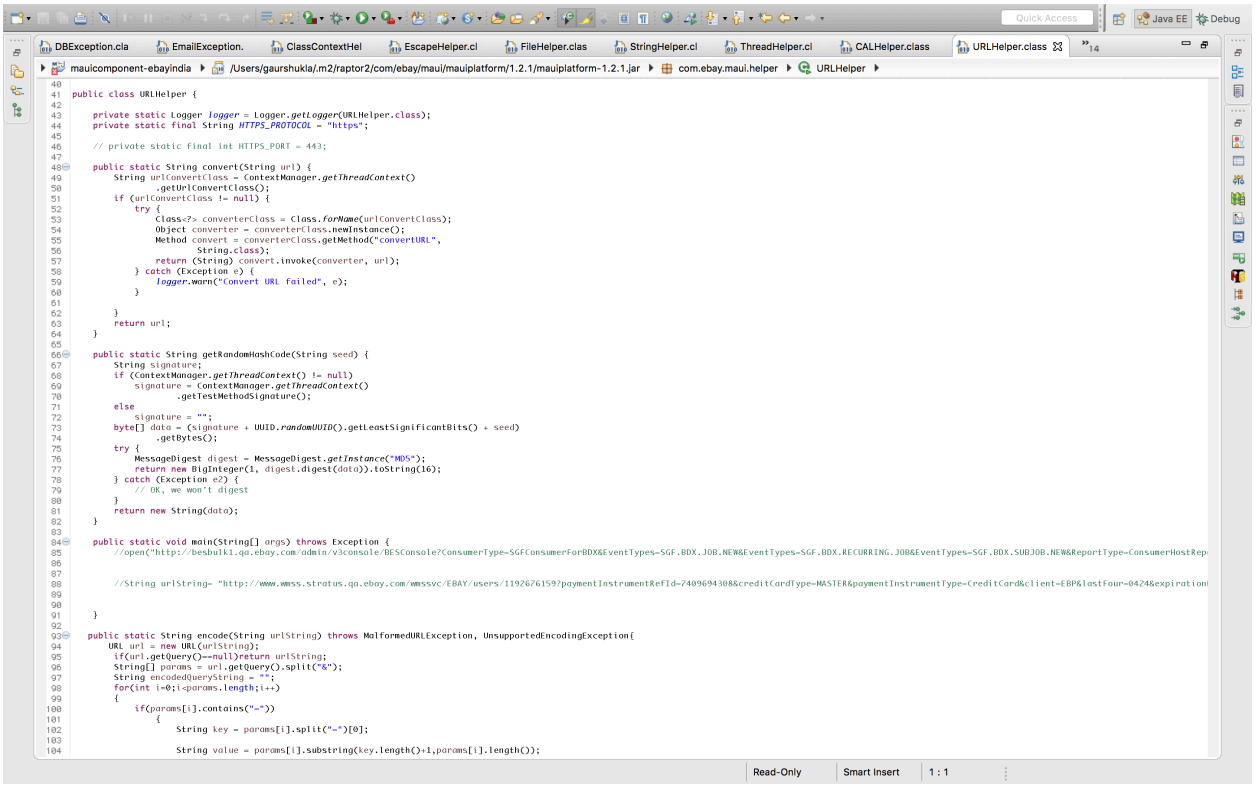
Read-Only | Smart Insert | 1 : 1



```
package com.ebay.maui.helper;

import java.lang.reflect.Method;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.util.UUID;

import com.ebay.maui.controller.ContextManager;

public class StringHelper {

    public static String constructMethodSignature(Method method, Object[] parameters) {
        return method.getDeclaringClass().getCanonicalName() + "." + method.getName() + "(" + constructParameterString(parameters) + ")";
    }

    public static String constructParameterString(Object[] parameters) {
        StringBuffer sbParam = new StringBuffer();

        if (parameters != null) {
            for (int i = 0; i < parameters.length; i++) {
                if (parameters[i] == null) {
                    sbParam.append("null, ");
                } else if (parameters[i] instanceof java.lang.String) {
                    sbParam.append("\"").append(parameters[i]).append("\", ");
                } else {
                    sbParam.append(parameters[i]).append(", ");
                }
            }
        }

        if (sbParam.length() > 0)
            sbParam.delete(sbParam.length() - 2, sbParam.length() - 1);

        return sbParam.toString();
    }

    public static String getRandomHashCode(String seed) {
        byte[] data = (ContextManager.getThreadContext().getTestMethodSignature() + UUID.randomUUID().getLeastSignificantBits() + seed).getBytes();
        try {
            MessageDigest digest = MessageDigest.getInstance("MD5");
            return new BigInteger(1, digest.digest(data)).toString(16);
        } catch (Exception e2) {
            // OK, we won't digest
            return new String(data);
        }
    }

}
```

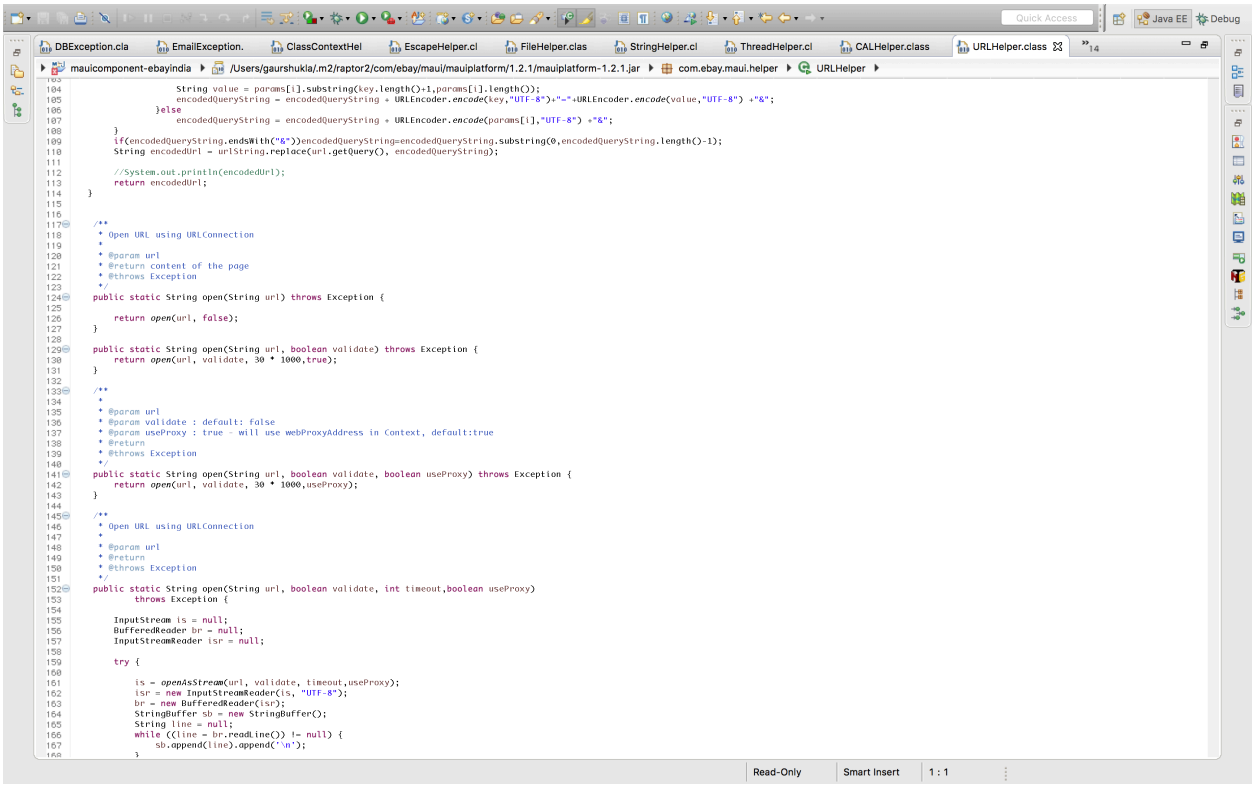


```
package com.ebay.maui.helper;

public class ThreadHelper {
    /**
     * Wait for seconds
     *
     * @param seconds
     */
    public static void waitForSeconds(int seconds) {
        /*
         * long timeout = 1000 * 60 * 3;
         * if(ContextManager.getThreadContext()!=null) timeout =
         * ContextManager.getThreadContext().getWebSessionTimeout(); if (seconds
         * * 1000 > timeout) throw new RuntimeException("Can not wait for " +
         * seconds +
         * " seconds. Because that is longer than then web session timeout of "
         * + timeout / 1000 +
         * " seconds which is defined in the testng configuration file.");
         */
        try {
            Thread.sleep(seconds * 1000);// KEEPME
        } catch (InterruptedException ignore) {

        }
    }
}
```



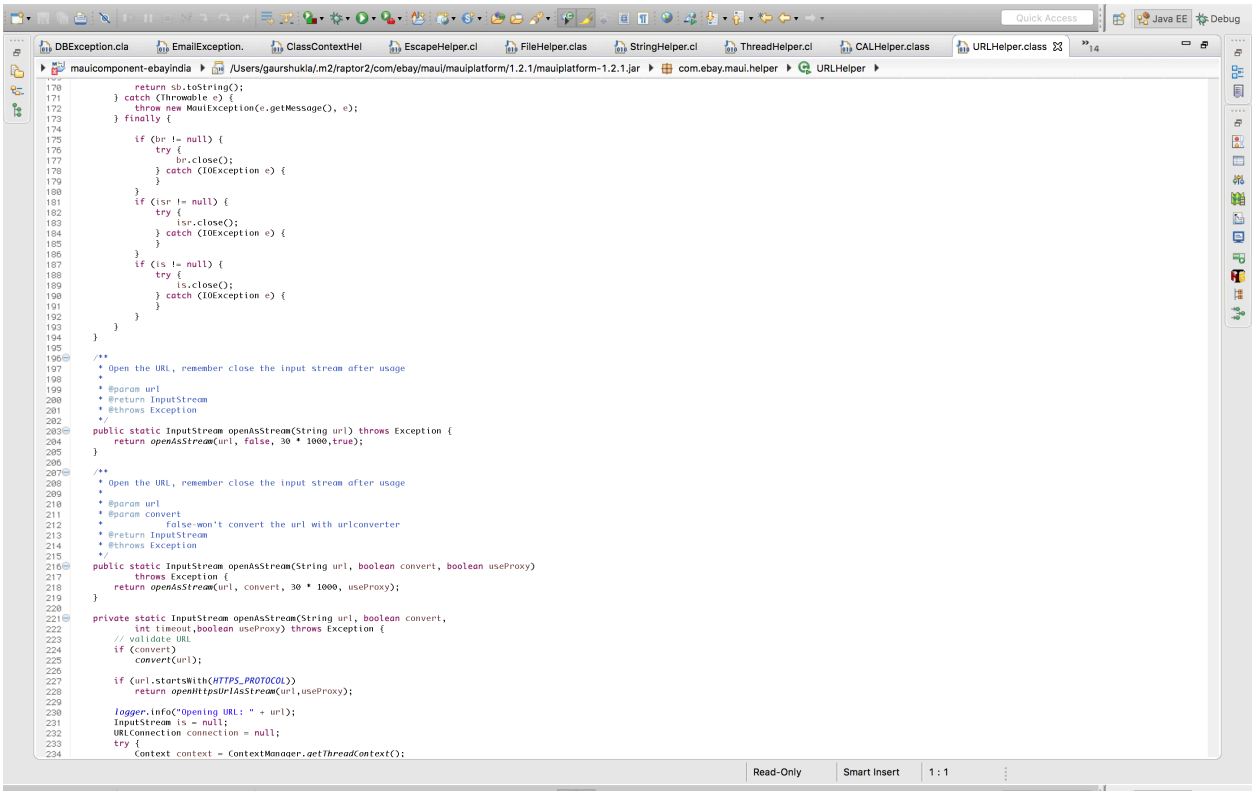
```
40
41  public class URLHelper {
42
43      private static Logger logger = Logger.getLogger(URLHelper.class);
44      private static final String HTTPS_PROTOCOL = "https";
45
46      // private static final int HTTPS_PORT = 443;
47
48      public static String convert(String url) {
49          String urlConvertClass = ContextManager.getThreadContext()
50                  .getUrlConvertClass();
51          if (urlConvertClass != null) {
52              try {
53                  Class<?> converterClass = Class.forName(urlConvertClass);
54                  Object converter = converterClass.newInstance();
55                  Method convert = converterClass.getMethod("convertURL",
56                          String.class);
57                  return (String) convert.invoke(converter, url);
58              } catch (Exception e) {
59                  logger.warn("Convert URL failed", e);
60              }
61
62          }
63          return url;
64      }
65
66      public static String getRandomHashCode(String seed) {
67          String signature;
68          if (ContextManager.getThreadContext() != null)
69              signature = ContextManager.getThreadContext()
70                      .getTestMethodSignature();
71          else
72              signature = "";
73          byte[] data = (signature + UUID.randomUUID().getLeastSignificantBits() + seed)
74                  .getBytes();
75          try {
76              MessageDigest digest = MessageDigest.getInstance("MD5");
77              return new BigInteger(1, digest.digest(data)).toString(16);
78          } catch (Exception e2) {
79              // OK, we won't digest
80          }
81          return new String(data);
82      }
83
84      public static void main(String[] args) throws Exception {
85          //open("http://besbulk1.qa.ebay.com/admin/v3console/BESConsole?ConsumerType=SGFConsumerForBDX&EventTypes=SGF.BDX.JOB.NEW&EventTypes=SGF.BDX.RECURRING.JOB&EventTypes=SGF.BDX.SUBJOB.NEW&ReportType=ConsumerHostRep
86
87
88          //String urlString= "http://www.wmss.stratus.qa.ebay.com/wmssvc/EBAY/users/1192676159?paymentInstrumentRefId=740969430&creditCardType=MASTER&paymentInstrumentType=CreditCard&client=EBP&lastFour=0424&expiration
89
90
91      }
92
93      public static String encode(String urlString) throws MalformedURLException, UnsupportedEncodingException{
94          URL url = new URL(urlString);
95          if(url.getQuery()==null)return urlString;
96          String[] params = url.getQuery().split("&");
97          String encodedQueryString = "";
98          for(int i=0;i<params.length;i++)
99          {
100             if(params[i].contains("="))
101                 {
102                     String key = params[i].split("=")[0];
103
104                     String value = params[i].substring(key.length()+1,params[i].length());
```

Read-Only | Smart Insert | 1 : 1



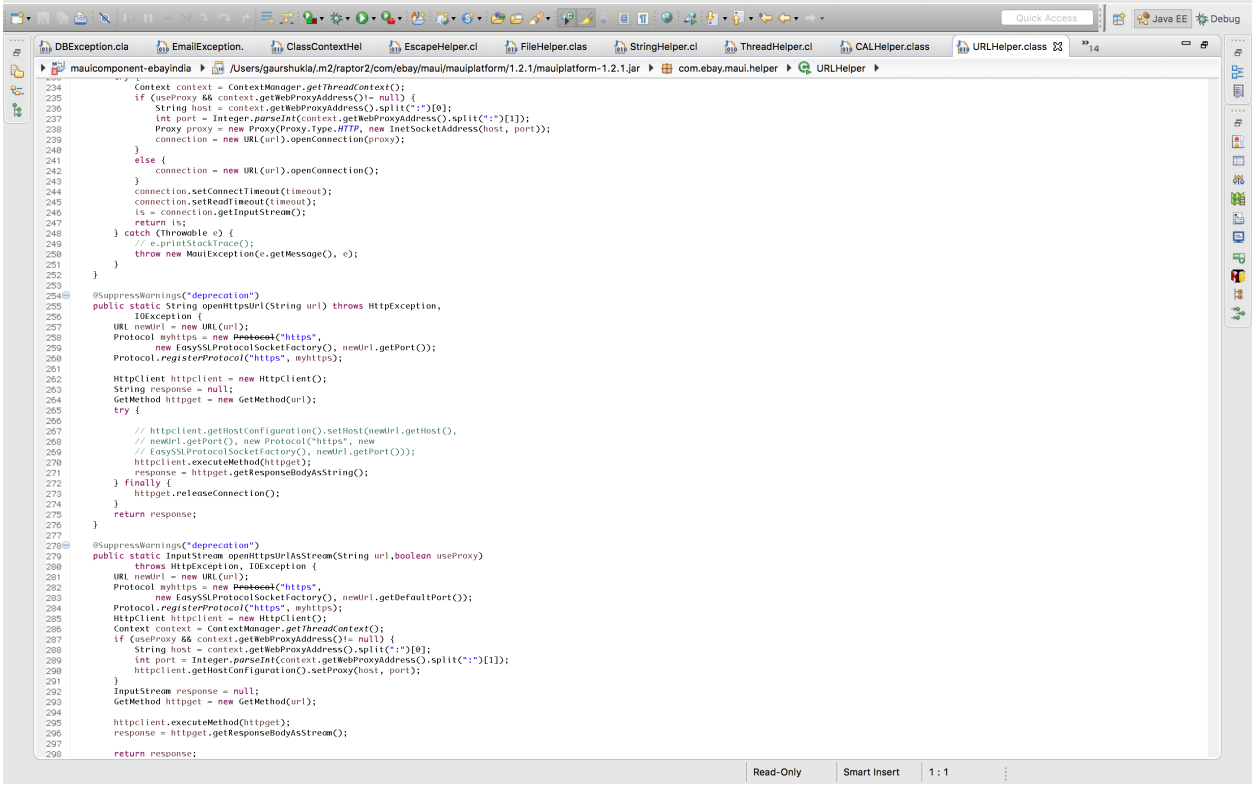
```
104                     String value = params[i].substring(key.length()+1,params[i].length());
105                     encodedQueryString = encodedQueryString + URLEncoder.encode(key,"UTF-8")+"="+URLEncoder.encode(value,"UTF-8") +"&";
106                 }else
107                     encodedQueryString = encodedQueryString + URLEncoder.encode(params[i],"UTF-8") +"&";
108         }
109         if(encodedQueryString.endsWith("&"))encodedQueryString=encodedQueryString.substring(0,encodedQueryString.length()-1);
110         String encodedUrl = urlString.replace(url.getQuery(), encodedQueryString);
111
112         //System.out.println(encodedUrl);
113         return encodedUrl;
114     }
115
116
117     /**
118      * Open URL using URLConnection
119      *
120      * @param url
121      * @return content of the page
122      * @throws Exception
123      */
124     public static String open(String url) throws Exception {
125
126         return open(url, false);
127     }
128
129     public static String open(String url, boolean validate) throws Exception {
130         return open(url, validate, 30 * 1000,true);
131     }
132
133     /**
134      *
135      * @param url
136      * @param validate : default: false
137      * @param useProxy : true - will use webProxyAddress in Context, default:true
138      * @return
139      * @throws Exception
140      */
141     public static String open(String url, boolean validate, boolean useProxy) throws Exception {
142         return open(url, validate, 30 * 1000,useProxy);
143     }
144
145     /**
146      * Open URL using URLConnection
147      *
148      * @param url
149      * @return
150      * @throws Exception
151      */
152     public static String open(String url, boolean validate, int timeout,boolean useProxy)
153             throws Exception {
154
155         InputStream is = null;
156         BufferedReader br = null;
157         InputStreamReader isr = null;
158
159         try {
160
161             is = openAsStream(url, validate, timeout,useProxy);
162             isr = new InputStreamReader(is, "UTF-8");
163             br = new BufferedReader(isr);
164             StringBuffer sb = new StringBuffer();
165             String line = null;
166             while ((line = br.readLine()) != null) {
167                 sb.append(line).append('\n');
168             }
```

Read-Only | Smart Insert | 1 : 1



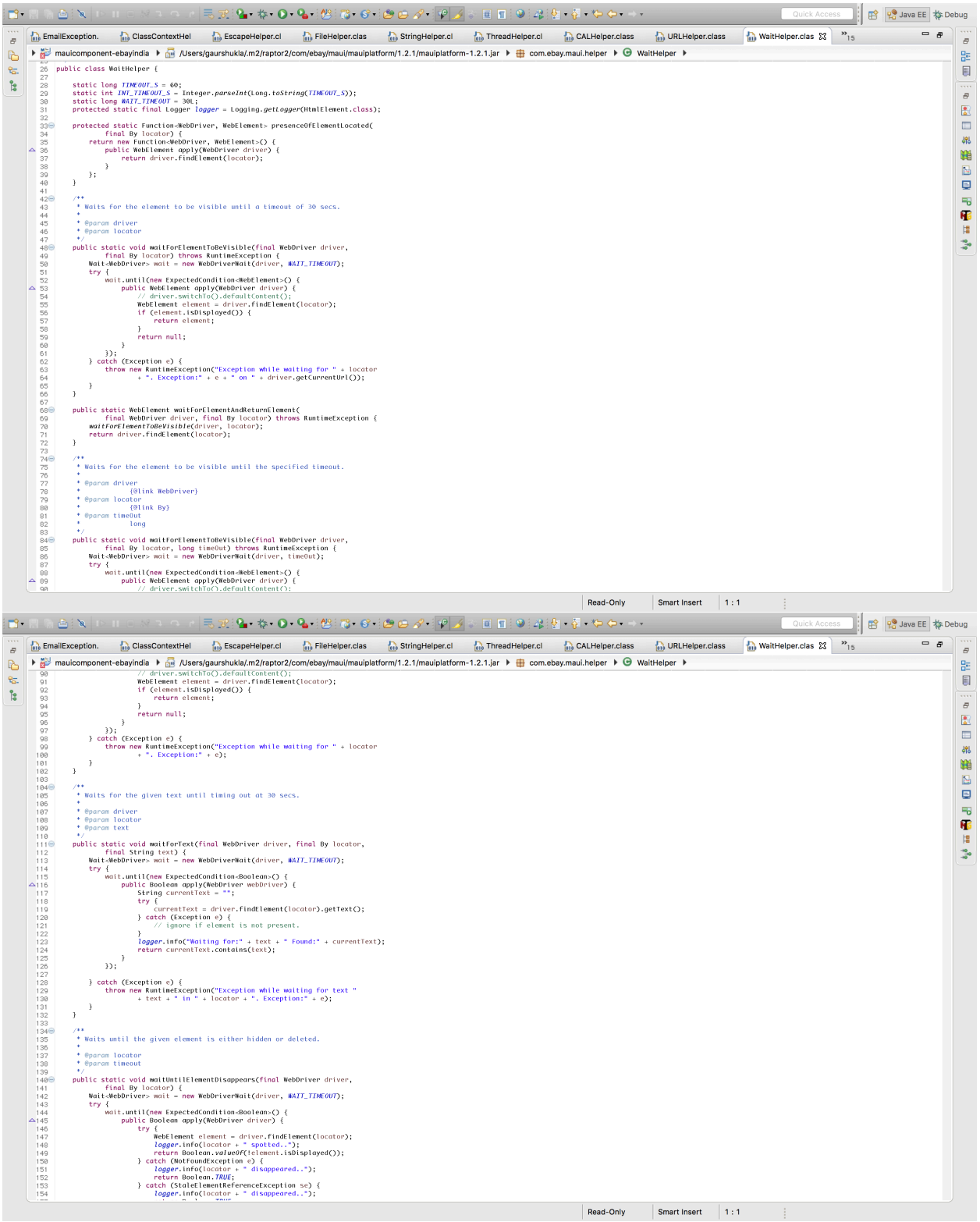
```
            return sb.toString();
        } catch (Throwable e) {
            throw new MauiException(e.getMessage(), e);
        } finally {

            if (br != null) {
                try {
                    br.close();
                } catch (IOException e) {
                }
            }
            if (isr != null) {
                try {
                    isr.close();
                } catch (IOException e) {
                }
            }
            if (is != null) {
                try {
                    is.close();
                } catch (IOException e) {
                }
            }
        }
    }

    /**
     * Open the URL, remember close the input stream after usage
     *
     * @param url
     * @return InputStream
     * @throws Exception
     */
    public static InputStream openAsStream(String url) throws Exception {
        return openAsStream(url, false, 30 * 1000,true);
    }

    /**
     * Open the URL, remember close the input stream after usage
     *
     * @param url
     * @param convert
     *            false-won't convert the url with urlconverter
     * @return InputStream
     * @throws Exception
     */
    public static InputStream openAsStream(String url, boolean convert, boolean useProxy)
            throws Exception {
        return openAsStream(url, convert, 30 * 1000, useProxy);
    }

    private static InputStream openAsStream(String url, boolean convert,
            int timeout,boolean useProxy) throws Exception {
        // validate URL
        if (convert)
            convert(url);

        if (url.startsWith(HTTPS_PROTOCOL))
            return openHttpsUrlAsStream(url,useProxy);

        logger.info("Opening URL: " + url);
        InputStream is = null;
        URLConnection connection = null;
        try {
            Context context = ContextManager.getThreadContext();
```

Read-Only | Smart Insert | 1 : 1

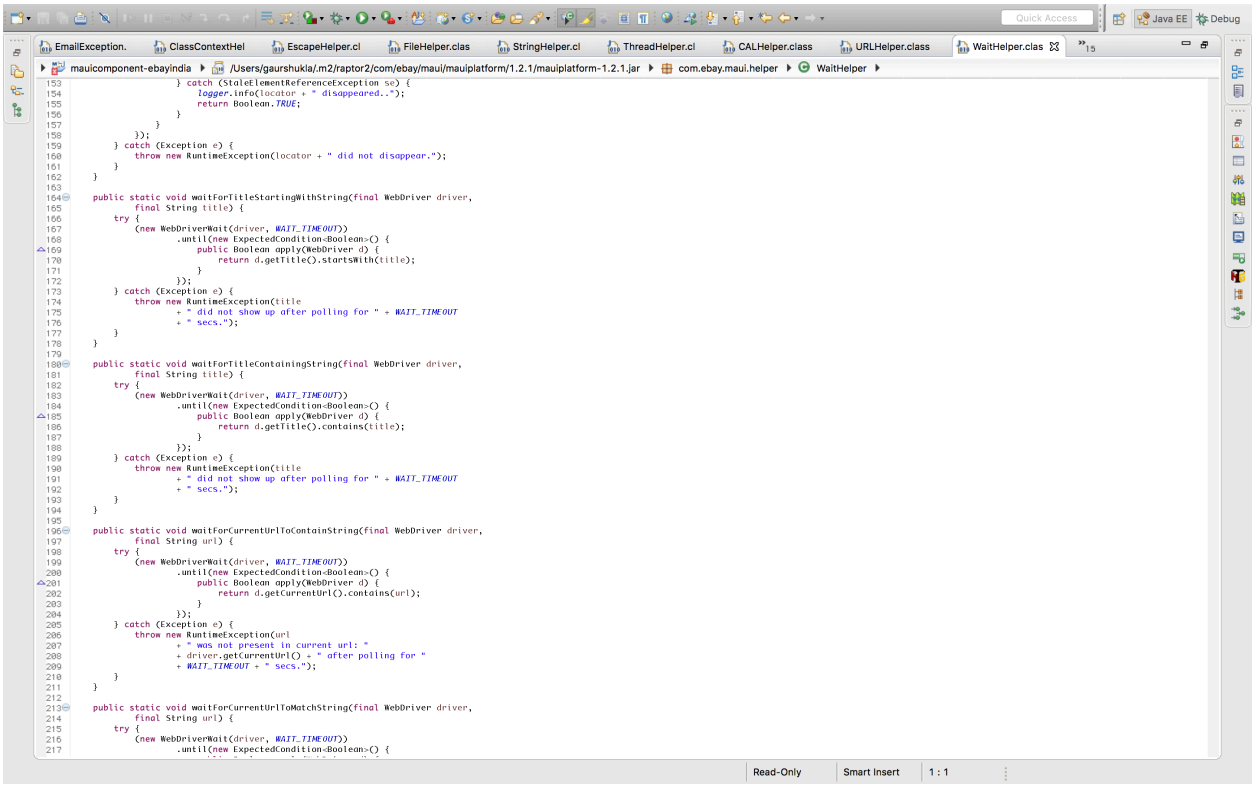


```
            Context context = ContextManager.getThreadContext();
            if (useProxy && context.getWebProxyAddress()!= null) {
                String host = context.getWebProxyAddress().split(":")[0];
                int port = Integer.parseInt(context.getWebProxyAddress().split(":")[1]);
                Proxy proxy = new Proxy(Proxy.Type.HTTP, new InetSocketAddress(host, port));
                connection = new URL(url).openConnection(proxy);
            }
            else {
                connection = new URL(url).openConnection();
            }
            connection.setConnectTimeout(timeout);
            connection.setReadTimeout(timeout);
            is = connection.getInputStream();
            return is;
        } catch (Throwable e) {
            // e.printStackTrace();
            throw new MauiException(e.getMessage(), e);
        }
    }

    @SuppressWarnings("deprecation")
    public static String openHttpsUrl(String url) throws HttpException,
            IOException {
        URL newUrl = new URL(url);
        Protocol myhttps = new Protocol("https",
                new EasySSLProtocolSocketFactory(), newUrl.getPort());
        Protocol.registerProtocol("https", myhttps);

        HttpClient httpclient = new HttpClient();
        String response = null;
        GetMethod httpget = new GetMethod(url);
        try {

            // httpclient.getHostConfiguration().setHost(newUrl.getHost(),
            // newUrl.getPort(), new Protocol("https", new
            // EasySSLProtocolSocketFactory(), newUrl.getPort()));
            httpclient.executeMethod(httpget);
            response = httpget.getResponseBodyAsString();
        } finally {
            httpget.releaseConnection();
        }
        return response;
    }

    @SuppressWarnings("deprecation")
    public static InputStream openHttpsUrlAsStream(String url,boolean useProxy)
            throws HttpException, IOException {
        URL newUrl = new URL(url);
        Protocol myhttps = new Protocol("https",
                new EasySSLProtocolSocketFactory(), newUrl.getDefaultPort());
        Protocol.registerProtocol("https", myhttps);
        HttpClient httpclient = new HttpClient();
        Context context = ContextManager.getThreadContext();
        if (useProxy && context.getWebProxyAddress()!= null) {
            String host = context.getWebProxyAddress().split(":")[0];
            int port = Integer.parseInt(context.getWebProxyAddress().split(":")[1]);
            httpclient.getHostConfiguration().setProxy(host, port);
        }
        InputStream response = null;
        GetMethod httpget = new GetMethod(url);

        httpclient.executeMethod(httpget);
        response = httpget.getResponseBodyAsStream();

        return response;
```

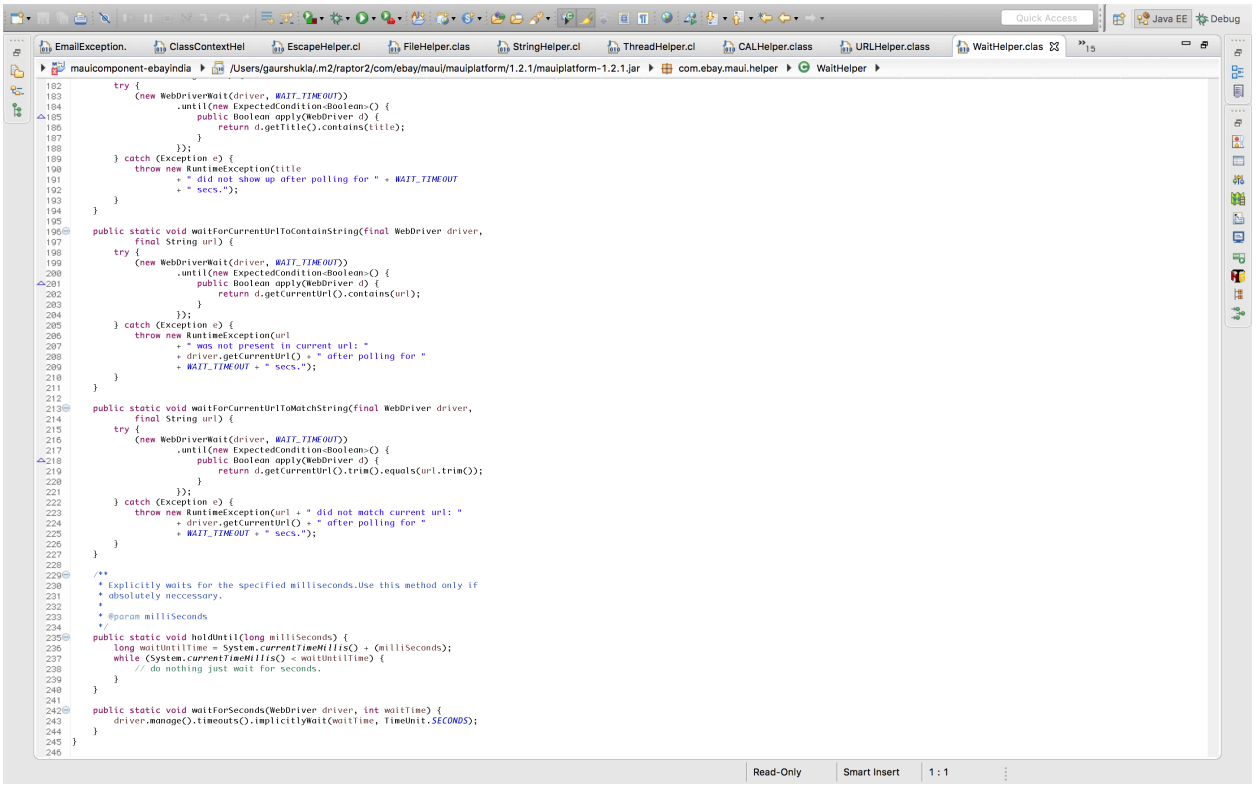
Read-Only | Smart Insert | 1 : 1



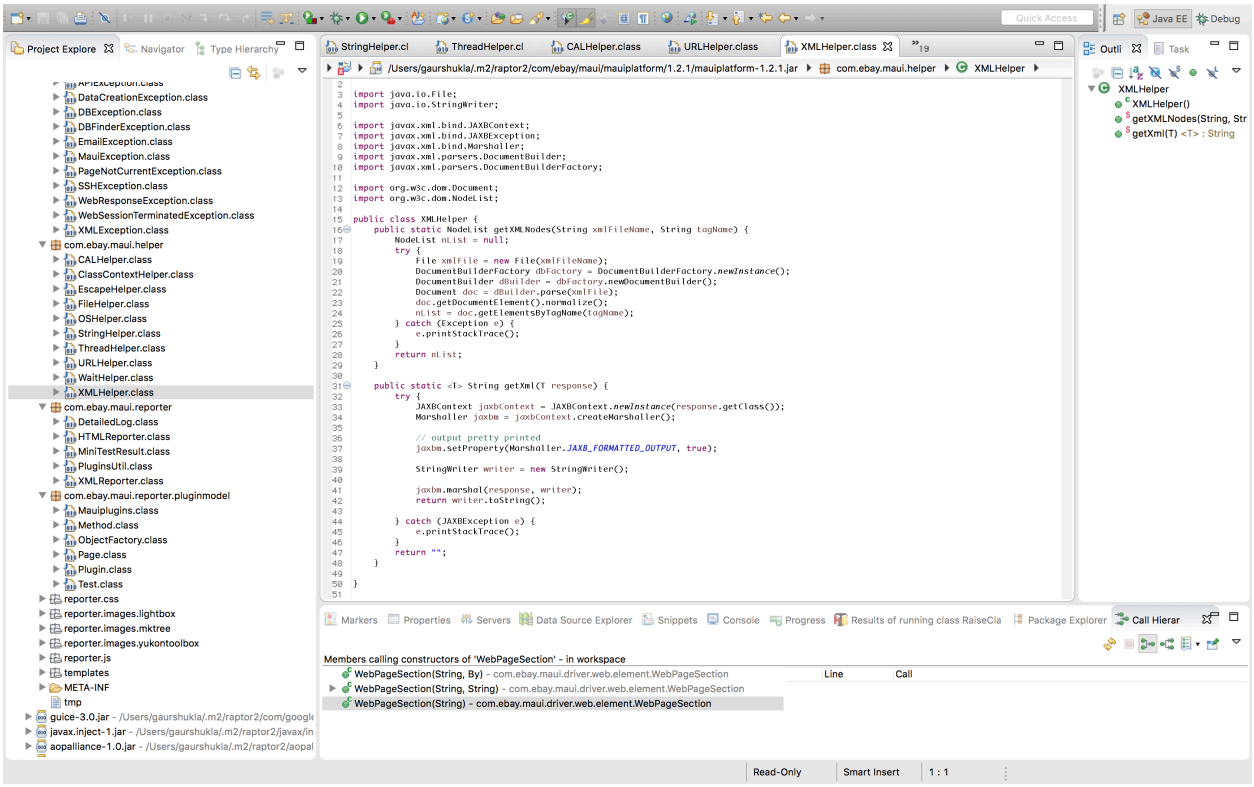
```
public class WaitHelper {

    static long TIMEOUT_S = 60;
    static int INT_TIMEOUT_S = Integer.parseInt(Long.toString(TIMEOUT_S));
    static long WAIT_TIMEOUT = 30L;
    protected static final Logger logger = Logging.getLogger(HtmlElement.class);

    protected static Function<WebDriver, WebElement> presenceOfElementLocated(
            final By locator) {
        return new Function<WebDriver, WebElement>() {
            public WebElement apply(WebDriver driver) {
                return driver.findElement(locator);
            }
        };
    }

    /**
     * Waits for the element to be visible until a timeout of 30 secs.
     *
     * @param driver
     * @param locator
     */
    public static void waitForElementToBeVisible(final WebDriver driver,
            final By locator) throws RuntimeException {
        Wait<WebDriver> wait = new WebDriverWait(driver, WAIT_TIMEOUT);
        try {
            wait.until(new ExpectedCondition<WebElement>() {
                public WebElement apply(WebDriver driver) {
                    // driver.switchTo().defaultContent();
                    WebElement element = driver.findElement(locator);
                    if (element.isDisplayed()) {
                        return element;
                    }
                    return null;
                }
            });
        } catch (Exception e) {
            throw new RuntimeException("Exception while waiting for " + locator
                    + ". Exception:" + e + " on " + driver.getCurrentUrl());
        }
    }

    public static WebElement waitForElementAndReturnElement(
            final WebDriver driver, final By locator) throws RuntimeException {
        waitForElementToBeVisible(driver, locator);
        return driver.findElement(locator);
    }

    /**
     * Waits for the element to be visible until the specified timeout.
     *
     * @param driver
     *            {@link WebDriver}
     * @param locator
     *            {@link By}
     * @param timeOut
     *            long
     */
    public static void waitForElementToBeVisible(final WebDriver driver,
            final By locator, long timeOut) throws RuntimeException {
        Wait<WebDriver> wait = new WebDriverWait(driver, timeOut);
        try {
            wait.until(new ExpectedCondition<WebElement>() {
                public WebElement apply(WebDriver driver) {
                    // driver.switchTo().defaultContent();
                    WebElement element = driver.findElement(locator);
                    if (element.isDisplayed()) {
                        return element;
                    }
                    return null;
                }
            });
        } catch (Exception e) {
            throw new RuntimeException("Exception while waiting for " + locator
                    + ". Exception:" + e);
        }
    }

    /**
     * Waits for the given text until timing out at 30 secs.
     *
     * @param driver
     * @param locator
     * @param text
     */
    public static void waitForText(final WebDriver driver, final By locator,
            final String text) {
        Wait<WebDriver> wait = new WebDriverWait(driver, WAIT_TIMEOUT);
        try {
            wait.until(new ExpectedCondition<Boolean>() {
                public Boolean apply(WebDriver webDriver) {
                    String currentText = "";
                    try {
                        currentText = driver.findElement(locator).getText();
                    } catch (Exception e) {
                        // ignore if element is not present.
                    }
                    logger.info("Waiting for:" + text + " Found:" + currentText);
                    return currentText.contains(text);
                }
            });

        } catch (Exception e) {
            throw new RuntimeException("Exception while waiting for text "
                    + text + " in " + locator + ". Exception:" + e);
        }
    }

    /**
     * Waits until the given element is either hidden or deleted.
     *
     * @param locator
     * @param timeout
     */
    public static void waitUntilElementDisappears(final WebDriver driver,
            final By locator) {
        Wait<WebDriver> wait = new WebDriverWait(driver, WAIT_TIMEOUT);
        try {
            wait.until(new ExpectedCondition<Boolean>() {
                public Boolean apply(WebDriver driver) {
                    try {
                        WebElement element = driver.findElement(locator);
                        logger.info(locator + " spotted..");
                        return Boolean.valueOf(!element.isDisplayed());
                    } catch (NotFoundException e) {
                        logger.info(locator + " disappeared..");
                        return Boolean.TRUE;
                    } catch (StaleElementReferenceException se) {
                        logger.info(locator + " disappeared..");
```



```
                    } catch (StaleElementReferenceException se) {
                        logger.info(locator + " disappeared..");
                        return Boolean.TRUE;
                    }
                }
            });
        } catch (Exception e) {
            throw new RuntimeException(locator + " did not disappear.");
        }
    }

    public static void waitForTitleStartingWithString(final WebDriver driver,
            final String title) {
        try {
            (new WebDriverWait(driver, WAIT_TIMEOUT))
                    .until(new ExpectedCondition<Boolean>() {
                        public Boolean apply(WebDriver d) {
                            return d.getTitle().startsWith(title);
                        }
                    });
        } catch (Exception e) {
            throw new RuntimeException(title
                    + " did not show up after polling for " + WAIT_TIMEOUT
                    + " secs.");
        }
    }

    public static void waitForTitleContainingString(final WebDriver driver,
            final String title) {
        try {
            (new WebDriverWait(driver, WAIT_TIMEOUT))
                    .until(new ExpectedCondition<Boolean>() {
                        public Boolean apply(WebDriver d) {
                            return d.getTitle().contains(title);
                        }
                    });
        } catch (Exception e) {
            throw new RuntimeException(title
                    + " did not show up after polling for " + WAIT_TIMEOUT
                    + " secs.");
        }
    }

    public static void waitForCurrentUrlToContainString(final WebDriver driver,
            final String url) {
        try {
            (new WebDriverWait(driver, WAIT_TIMEOUT))
                    .until(new ExpectedCondition<Boolean>() {
                        public Boolean apply(WebDriver d) {
                            return d.getCurrentUrl().contains(url);
                        }
                    });
        } catch (Exception e) {
            throw new RuntimeException(url
                    + " was not present in current url: "
                    + driver.getCurrentUrl() + " after polling for "
                    + WAIT_TIMEOUT + " secs.");
        }
    }

    public static void waitForCurrentUrlToMatchString(final WebDriver driver,
            final String url) {
        try {
            (new WebDriverWait(driver, WAIT_TIMEOUT))
                    .until(new ExpectedCondition<Boolean>() {
```



```
        try {
            (new WebDriverWait(driver, WAIT_TIMEOUT))
                    .until(new ExpectedCondition<Boolean>() {
                        public Boolean apply(WebDriver d) {
                            return d.getTitle().contains(title);
                        }
                    });
        } catch (Exception e) {
            throw new RuntimeException(title
                    + " did not show up after polling for " + WAIT_TIMEOUT
                    + " secs.");
        }
    }

    public static void waitForCurrentUrlToContainString(final WebDriver driver,
            final String url) {
        try {
            (new WebDriverWait(driver, WAIT_TIMEOUT))
                    .until(new ExpectedCondition<Boolean>() {
                        public Boolean apply(WebDriver d) {
                            return d.getCurrentUrl().contains(url);
                        }
                    });
        } catch (Exception e) {
            throw new RuntimeException(url
                    + " was not present in current url: "
                    + driver.getCurrentUrl() + " after polling for "
                    + WAIT_TIMEOUT + " secs.");
        }
    }

    public static void waitForCurrentUrlToMatchString(final WebDriver driver,
            final String url) {
        try {
            (new WebDriverWait(driver, WAIT_TIMEOUT))
                    .until(new ExpectedCondition<Boolean>() {
                        public Boolean apply(WebDriver d) {
                            return d.getCurrentUrl().trim().equals(url.trim());
                        }
                    });
        } catch (Exception e) {
            throw new RuntimeException(url + " did not match current url: "
                    + driver.getCurrentUrl() + " after polling for "
                    + WAIT_TIMEOUT + " secs.");
        }
    }

    /**
     * Explicitly waits for the specified milliseconds.Use this method only if
     * absolutely neccessary.
     *
     * @param milliSeconds
     */
    public static void holdUntil(long milliSeconds) {
        long waitUntilTime = System.currentTimeMillis() + (milliSeconds);
        while (System.currentTimeMillis() < waitUntilTime) {
            // do nothing just wait for seconds.
        }
    }

    public static void waitForSeconds(WebDriver driver, int waitTime) {
        driver.manage().timeouts().implicitlyWait(waitTime, TimeUnit.SECONDS);
    }
}
```


Project Explore
Navigator
Type Hierarchy
DataCreationException.class
DBException.class
DBFinderException.class
EmailException.class
MauiException.class
PageNotCurrentException.class
SSHException.class
WebResponseException.class
WebSessionTerminatedException.class
XMLException.class
com.ebay.maui.helper
CALHelper.class
ClassContextHelper.class
EscapeHelper.class
FileHelper.class
OSHelper.class
StringHelper.class
ThreadHelper.class
URLHelper.class
WaitHelper.class
XMLHelper.class
com.ebay.maui.reporter
DetailedLog.class
HTMLReporter.class
MiniTestResult.class
PluginsUtil.class
XMLReporter.class
com.ebay.maui.reporter.pluginmodel
Mauiplugins.class
Method.class
ObjectFactory.class
Page.class
Plugin.class
Test.class
reporter.css
reporter.images.lightbox
reporter.images.mktree
reporter.images.yukontoolbox
reporter.js
templates
META-INF
tmp
guice-3.0.jar - /Users/gaurshukla/.m2/raptor2/com/google
javax.inject-1.jar - /Users/gaurshukla/.m2/raptor2/javax/in
aopalliance-1.0.jar - /Users/gaurshukla/.m2/raptor2/aopal
StringHelper.cl
ThreadHelper.cl
CALHelper.class
URLHelper.class
XMLHelper.class
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.helper
XMLHelper
import java.io.File;
import java.io.StringWriter;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
public class XMLHelper {
public static NodeList getXMLNodes(String xmlFileName, String tagName) {
NodeList nList = null;
try {
File xmlFile = new File(xmlFileName);
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document doc = dBuilder.parse(xmlFile);
doc.getDocumentElement().normalize();
nList = doc.getElementsByTagName(tagName);
} catch (Exception e) {
e.printStackTrace();
}
return nList;
}
public static <T> String getXml(T response) {
try {
JAXBContext jaxbContext = JAXBContext.newInstance(response.getClass());
Marshaller jaxbm = jaxbContext.createMarshaller();
// output pretty printed
jaxbm.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
StringWriter writer = new StringWriter();
jaxbm.marshal(response, writer);
return writer.toString();
} catch (JAXBException e) {
e.printStackTrace();
}
return "";
}
}
Outli
Task
XMLHelper
XMLHelper()
getXMLNodes(String, Str
getXml(T) <T> : String
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
Line
Call
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
Read-Only
Smart Insert
1 : 1