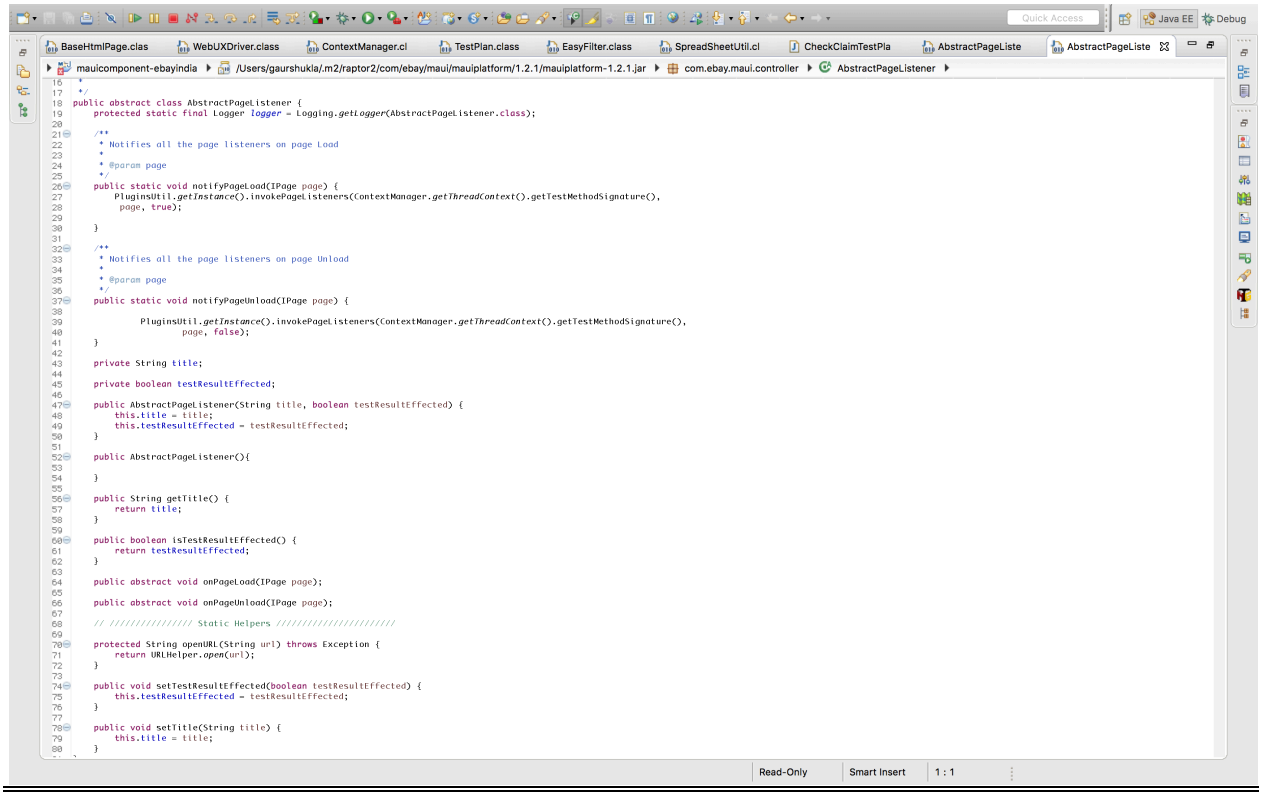
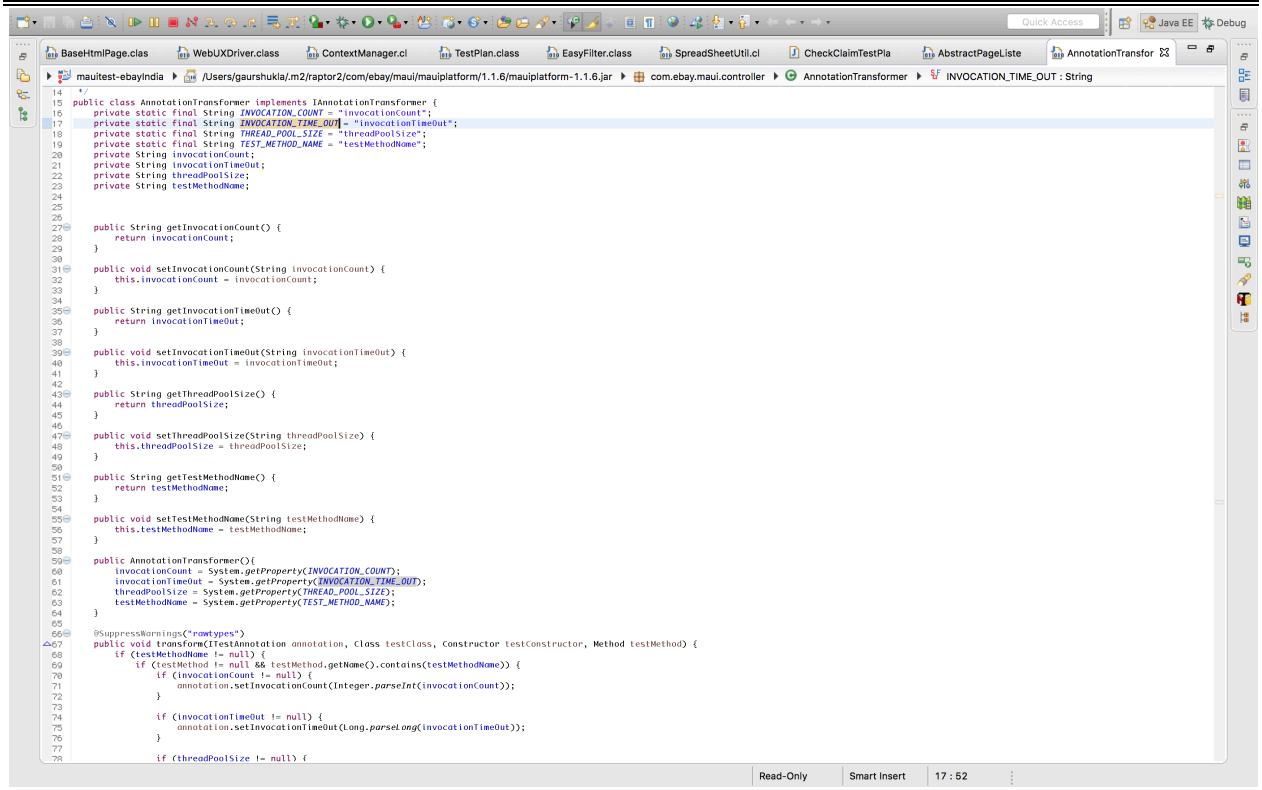
Controller:

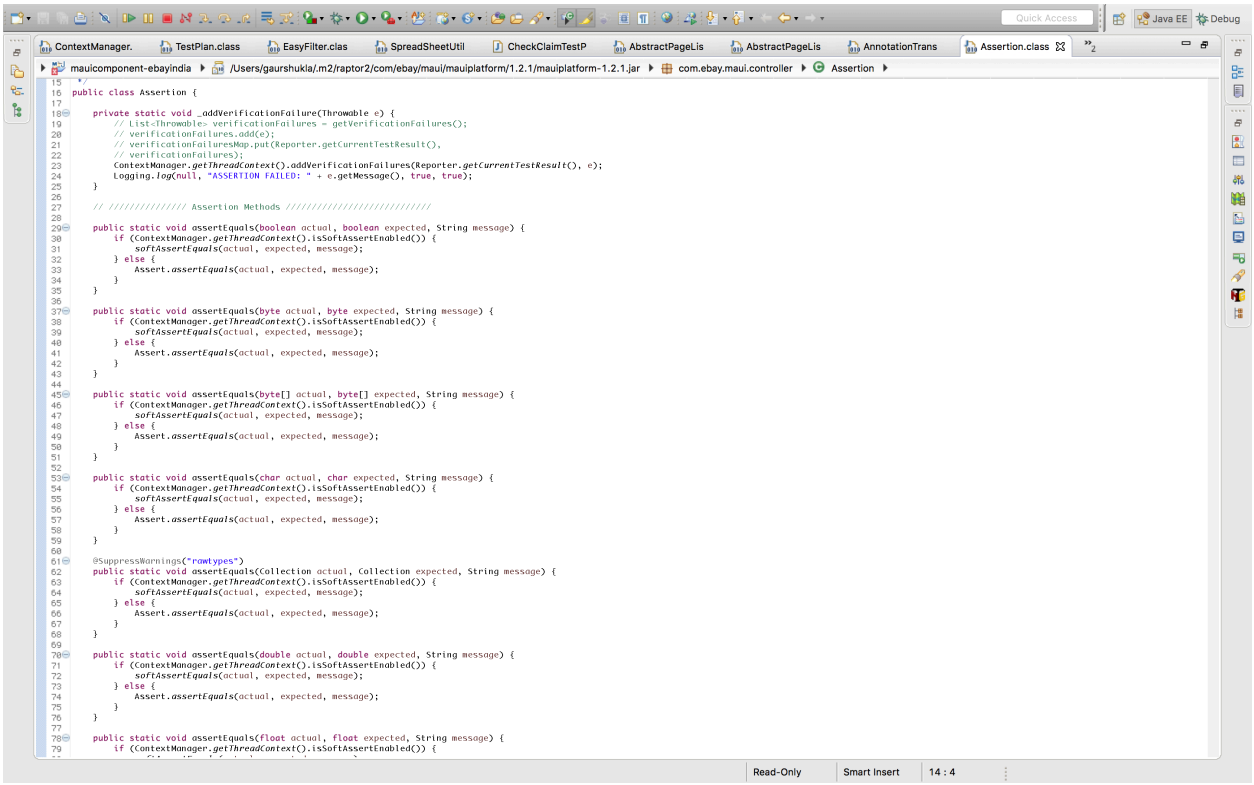


```
 *
 */
public abstract class AbstractPageListener {
    protected static final Logger logger = Logging.getLogger(AbstractPageListener.class);

    /**
     * Notifies all the page listeners on page Load
     *
     * @param page
     */
    public static void notifyPageLoad(IPage page) {
        PluginsUtil.getInstance().invokePageListeners(ContextManager.getThreadContext().getTestMethodSignature(),
         page, true);

    }

    /**
     * Notifies all the page listeners on page Unload
     *
     * @param page
     */
    public static void notifyPageUnload(IPage page) {

            PluginsUtil.getInstance().invokePageListeners(ContextManager.getThreadContext().getTestMethodSignature(),
                    page, false);
    }

    private String title;

    private boolean testResultEffected;

    public AbstractPageListener(String title, boolean testResultEffected) {
        this.title = title;
        this.testResultEffected = testResultEffected;
    }

    public AbstractPageListener(){

    }

    public String getTitle() {
        return title;
    }

    public boolean isTestResultEffected() {
        return testResultEffected;
    }

    public abstract void onPageLoad(IPage page);

    public abstract void onPageUnload(IPage page);

    // /////////////// Static Helpers //////////////////////

    protected String openURL(String url) throws Exception {
        return URLHelper.open(url);
    }

    public void setTestResultEffected(boolean testResultEffected) {
        this.testResultEffected = testResultEffected;
    }

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

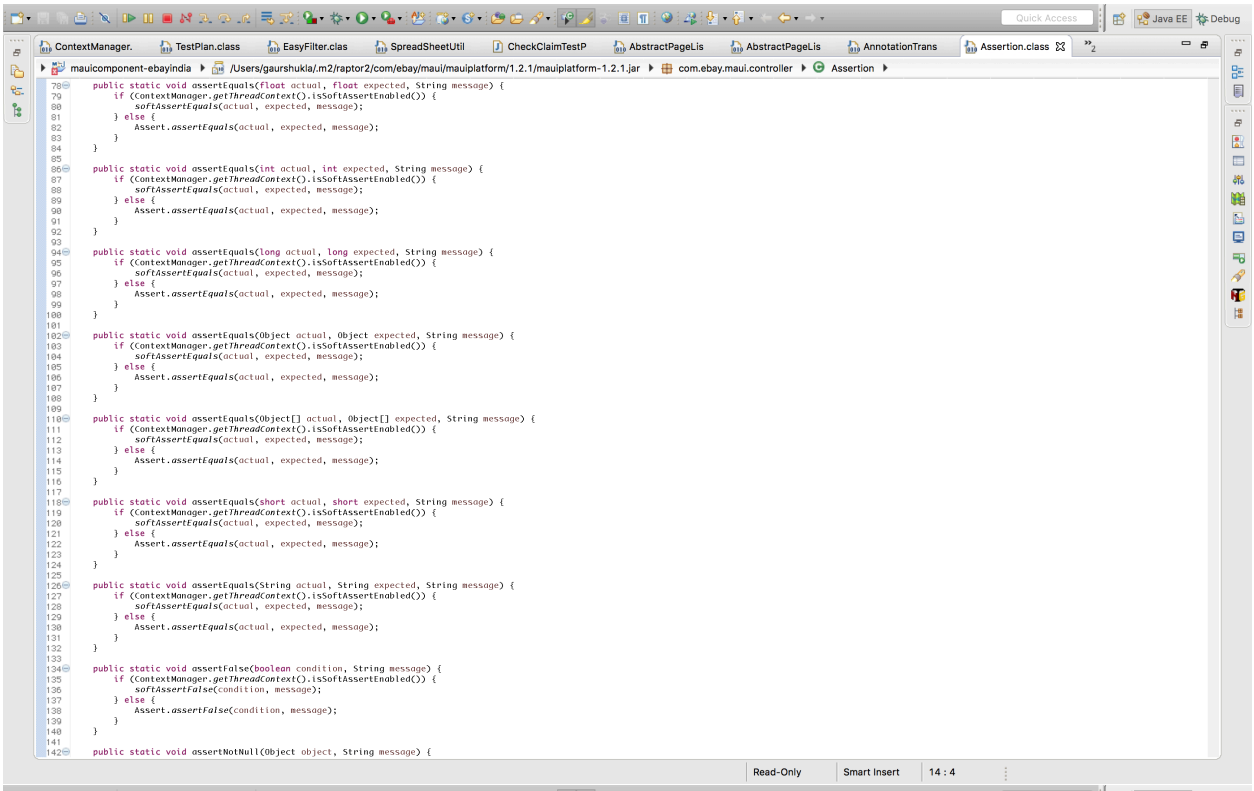


```
 */
public class AnnotationTransformer implements IAnnotationTransformer {
    private static final String INVOCATION_COUNT = "invocationCount";
    private static final String INVOCATION_TIME_OUT = "invocationTimeOut";
    private static final String THREAD_POOL_SIZE = "threadPoolSize";
    private static final String TEST_METHOD_NAME = "testMethodName";
    private String invocationCount;
    private String invocationTimeOut;
    private String threadPoolSize;
    private String testMethodName;


    public String getInvocationCount() {
        return invocationCount;
    }

    public void setInvocationCount(String invocationCount) {
        this.invocationCount = invocationCount;
    }

    public String getInvocationTimeOut() {
        return invocationTimeOut;
    }

    public void setInvocationTimeOut(String invocationTimeOut) {
        this.invocationTimeOut = invocationTimeOut;
    }

    public String getThreadPoolSize() {
        return threadPoolSize;
    }

    public void setThreadPoolSize(String threadPoolSize) {
        this.threadPoolSize = threadPoolSize;
    }

    public String getTestMethodName() {
        return testMethodName;
    }

    public void setTestMethodName(String testMethodName) {
        this.testMethodName = testMethodName;
    }

    public AnnotationTransformer(){
        invocationCount = System.getProperty(INVOCATION_COUNT);
        invocationTimeOut = System.getProperty(INVOCATION_TIME_OUT);
        threadPoolSize = System.getProperty(THREAD_POOL_SIZE);
        testMethodName = System.getProperty(TEST_METHOD_NAME);
    }

    @SuppressWarnings("rawtypes")
    public void transform(ITestAnnotation annotation, Class testClass, Constructor testConstructor, Method testMethod) {
        if (testMethodName != null) {
            if (testMethod != null && testMethod.getName().contains(testMethodName)) {
                if (invocationCount != null) {
                    annotation.setInvocationCount(Integer.parseInt(invocationCount));
                }

                if (invocationTimeOut != null) {
                    annotation.setInvocationTimeOut(Long.parseLong(invocationTimeOut));
                }

                if (threadPoolSize != null) {
```

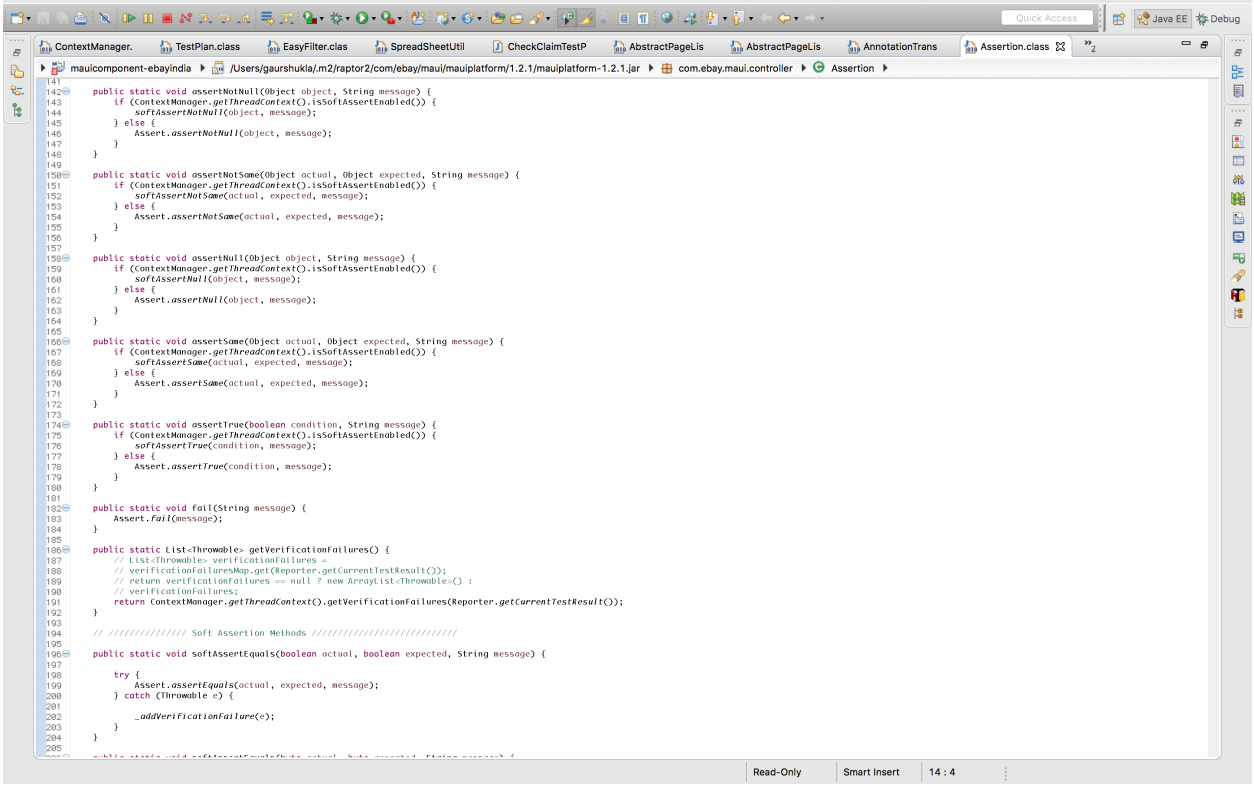
```
    public String getInvocationTimeOut() {
        return invocationTimeOut;
    }

    public void setInvocationTimeOut(String invocationTimeOut) {
        this.invocationTimeOut = invocationTimeOut;
    }

    public String getThreadPoolSize() {
        return threadPoolSize;
    }

    public void setThreadPoolSize(String threadPoolSize) {
        this.threadPoolSize = threadPoolSize;
    }

    public String getTestMethodName() {
        return testMethodName;
    }

    public void setTestMethodName(String testMethodName) {
        this.testMethodName = testMethodName;
    }

    public AnnotationTransformer(){
        invocationCount = System.getProperty(INVOCATION_COUNT);
        invocationTimeOut = System.getProperty(INVOCATION_TIME_OUT);
        threadPoolSize = System.getProperty(THREAD_POOL_SIZE);
        testMethodName = System.getProperty(TEST_METHOD_NAME);
    }

    @SuppressWarnings("rawtypes")
    public void transform(ITestAnnotation annotation, Class testClass, Constructor testConstructor, Method testMethod) {
        if (testMethodName != null) {
            if (testMethod != null && testMethod.getName().contains(testMethodName)) {
                if (invocationCount != null) {
                    annotation.setInvocationCount(Integer.parseInt(invocationCount));
                }

                if (invocationTimeOut != null) {
                    annotation.setInvocationTimeOut(Long.parseLong(invocationTimeOut));
                }

                if (threadPoolSize != null) {
                    annotation.setThreadPoolSize(Integer.parseInt(threadPoolSize));
                }
            }
        } else {
            if (invocationCount != null) {
                annotation.setInvocationCount(Integer.parseInt(invocationCount));
            }

            if (invocationTimeOut != null) {
                annotation.setInvocationTimeOut(Long.parseLong(invocationTimeOut));
            }

            if (threadPoolSize != null) {
                annotation.setThreadPoolSize(Integer.parseInt(threadPoolSize));
            }
        }

    }
}
```



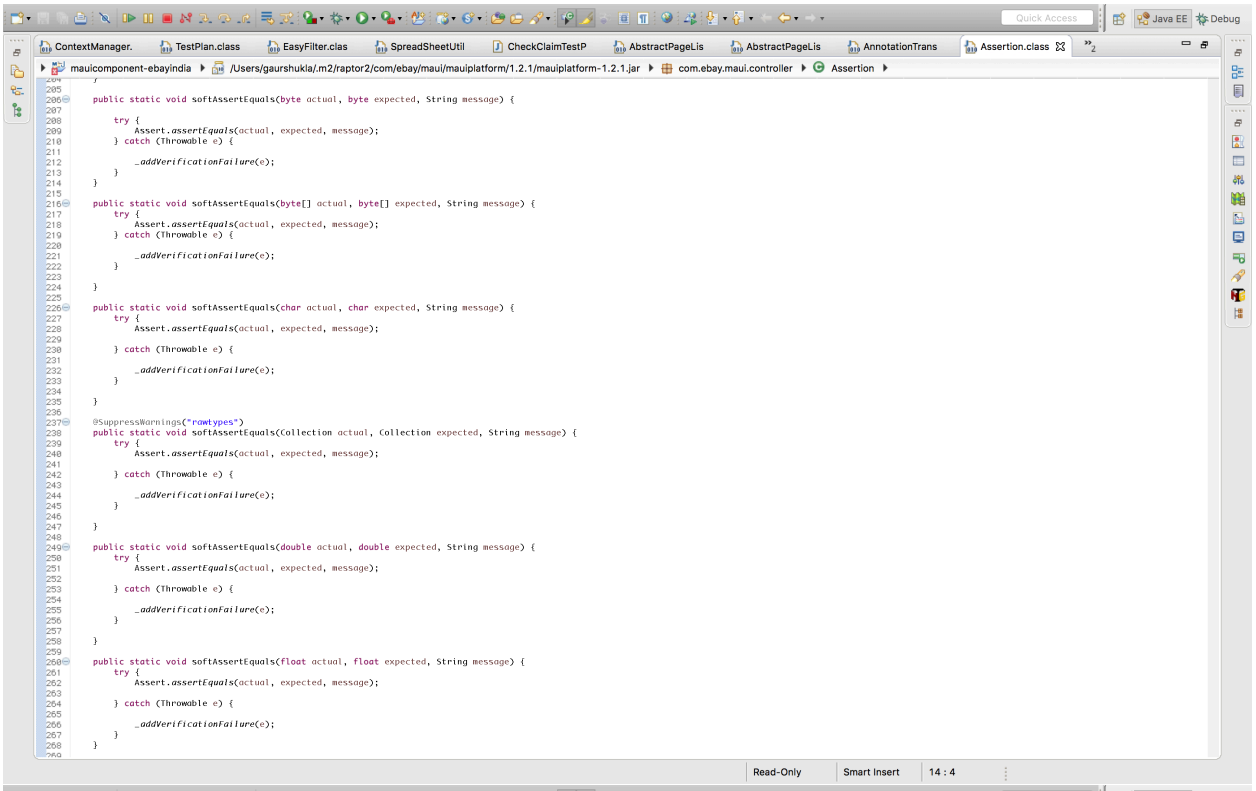

```
public class Assertion {

    private static void _addVerificationFailure(Throwable e) {
        // List<Throwable> verificationFailures = getVerificationFailures();
        // verificationFailures.add(e);
        // verificationFailuresMap.put(Reporter.getCurrentTestResult(),
        // verificationFailures);
        ContextManager.getThreadContext().addVerificationFailures(Reporter.getCurrentTestResult(), e);
        Logging.log(null, "ASSERTION FAILED: " + e.getMessage(), true, true);
    }

    // ############### Assertion Methods ############################

    public static void assertEquals(boolean actual, boolean expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertEquals(actual, expected, message);
        } else {
            Assert.assertEquals(actual, expected, message);
        }
    }

    public static void assertEquals(byte actual, byte expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertEquals(actual, expected, message);
        } else {
            Assert.assertEquals(actual, expected, message);
        }
    }

    public static void assertEquals(byte[] actual, byte[] expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertEquals(actual, expected, message);
        } else {
            Assert.assertEquals(actual, expected, message);
        }
    }

    public static void assertEquals(char actual, char expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertEquals(actual, expected, message);
        } else {
            Assert.assertEquals(actual, expected, message);
        }
    }

    @SuppressWarnings("rawtypes")
    public static void assertEquals(Collection actual, Collection expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertEquals(actual, expected, message);
        } else {
            Assert.assertEquals(actual, expected, message);
        }
    }

    public static void assertEquals(double actual, double expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertEquals(actual, expected, message);
        } else {
            Assert.assertEquals(actual, expected, message);
        }
    }

    public static void assertEquals(float actual, float expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
```



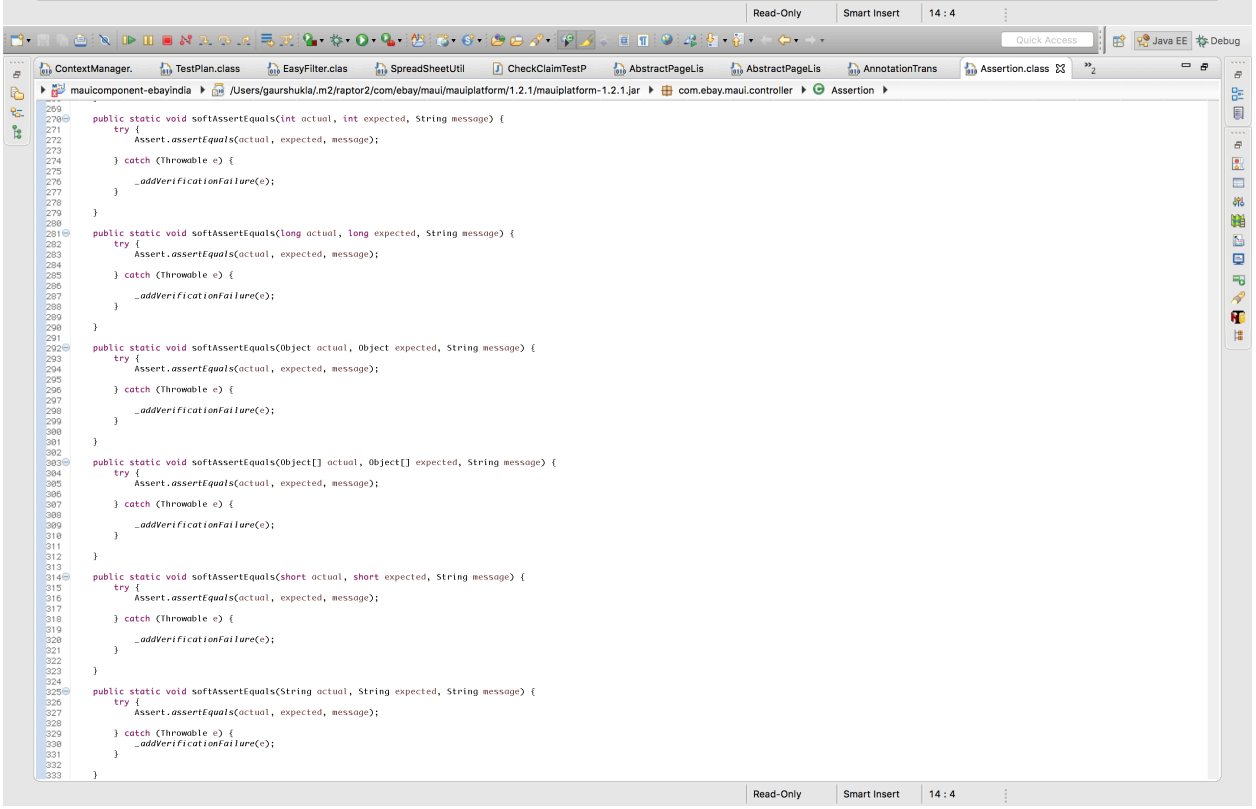
```
    public static void assertEquals(float actual, float expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertEquals(actual, expected, message);
        } else {
            Assert.assertEquals(actual, expected, message);
        }
    }

    public static void assertEquals(int actual, int expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertEquals(actual, expected, message);
        } else {
            Assert.assertEquals(actual, expected, message);
        }
    }

    public static void assertEquals(long actual, long expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertEquals(actual, expected, message);
        } else {
            Assert.assertEquals(actual, expected, message);
        }
    }

    public static void assertEquals(Object actual, Object expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertEquals(actual, expected, message);
        } else {
            Assert.assertEquals(actual, expected, message);
        }
    }

    public static void assertEquals(Object[] actual, Object[] expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertEquals(actual, expected, message);
        } else {
            Assert.assertEquals(actual, expected, message);
        }
    }

    public static void assertEquals(short actual, short expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertEquals(actual, expected, message);
        } else {
            Assert.assertEquals(actual, expected, message);
        }
    }

    public static void assertEquals(String actual, String expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertEquals(actual, expected, message);
        } else {
            Assert.assertEquals(actual, expected, message);
        }
    }

    public static void assertFalse(boolean condition, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertFalse(condition, message);
        } else {
            Assert.assertFalse(condition, message);
        }
    }

    public static void assertNotNull(Object object, String message) {
```



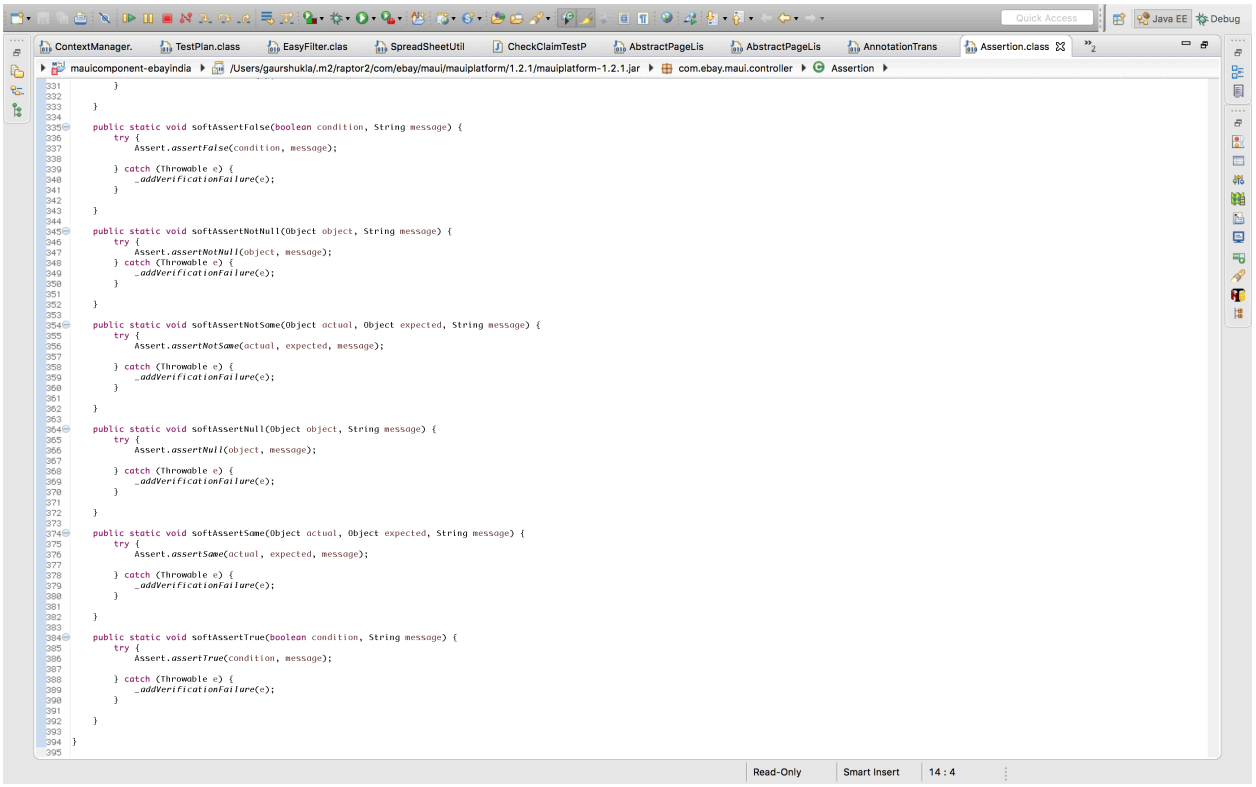
```
    public static void assertNotNull(Object object, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertNotNull(object, message);
        } else {
            Assert.assertNotNull(object, message);
        }
    }

    public static void assertNotSame(Object actual, Object expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertNotSame(actual, expected, message);
        } else {
            Assert.assertNotSame(actual, expected, message);
        }
    }

    public static void assertNull(Object object, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertNull(object, message);
        } else {
            Assert.assertNull(object, message);
        }
    }

    public static void assertSame(Object actual, Object expected, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertSame(actual, expected, message);
        } else {
            Assert.assertSame(actual, expected, message);
        }
    }

    public static void assertTrue(boolean condition, String message) {
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
            softAssertTrue(condition, message);
        } else {
            Assert.assertTrue(condition, message);
        }
    }

    public static void fail(String message) {
        Assert.fail(message);
    }

    public static List<Throwable> getVerificationFailures() {
        // List<Throwable> verificationFailures =
        // verificationFailuresMap.get(Reporter.getCurrentTestResult());
        // return verificationFailures == null ? new ArrayList<Throwable>() :
        // verificationFailures;
        return ContextManager.getThreadContext().getVerificationFailures(Reporter.getCurrentTestResult());
    }

    // ////////////// Soft Assertion Methods ////////////////////////////

    public static void softAssertEquals(boolean actual, boolean expected, String message) {

        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {

            _addVerificationFailure(e);
        }
    }
```



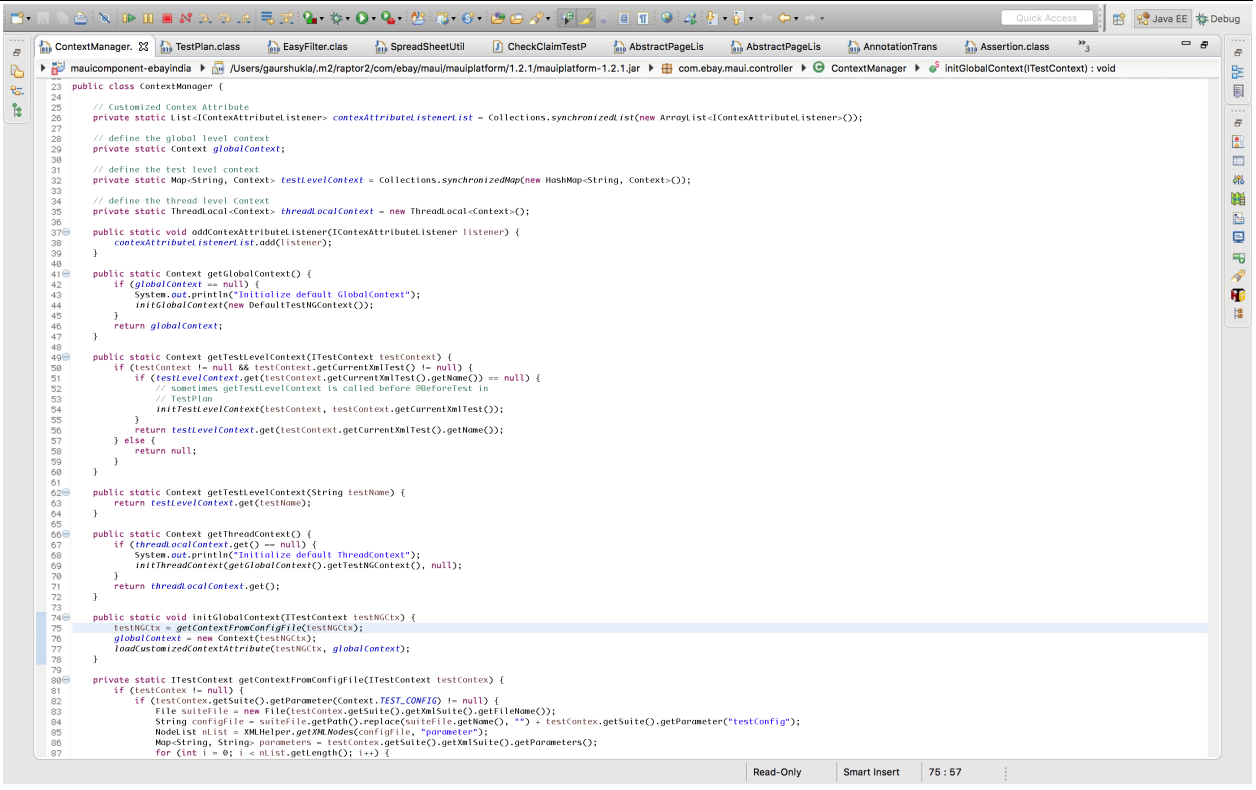
```
    public static void softAssertEquals(byte actual, byte expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }
    public static void softAssertEquals(byte[] actual, byte[] expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }
    public static void softAssertEquals(char actual, char expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }
    @SuppressWarnings("rawtypes")
    public static void softAssertEquals(Collection actual, Collection expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }
    public static void softAssertEquals(double actual, double expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }
    public static void softAssertEquals(float actual, float expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }
```

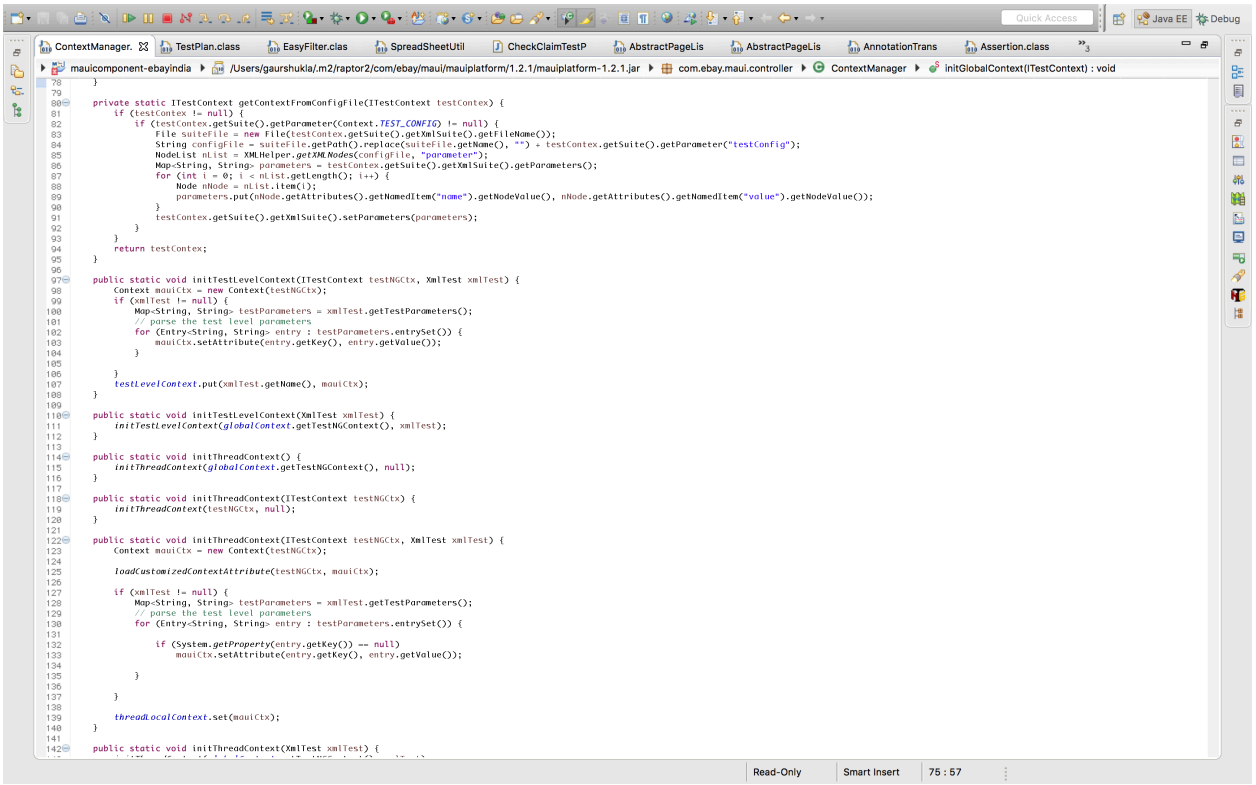


```
    public static void softAssertEquals(int actual, int expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }
    public static void softAssertEquals(long actual, long expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }
    public static void softAssertEquals(Object actual, Object expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }
    public static void softAssertEquals(Object[] actual, Object[] expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }
    public static void softAssertEquals(short actual, short expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }
    public static void softAssertEquals(String actual, String expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }
```

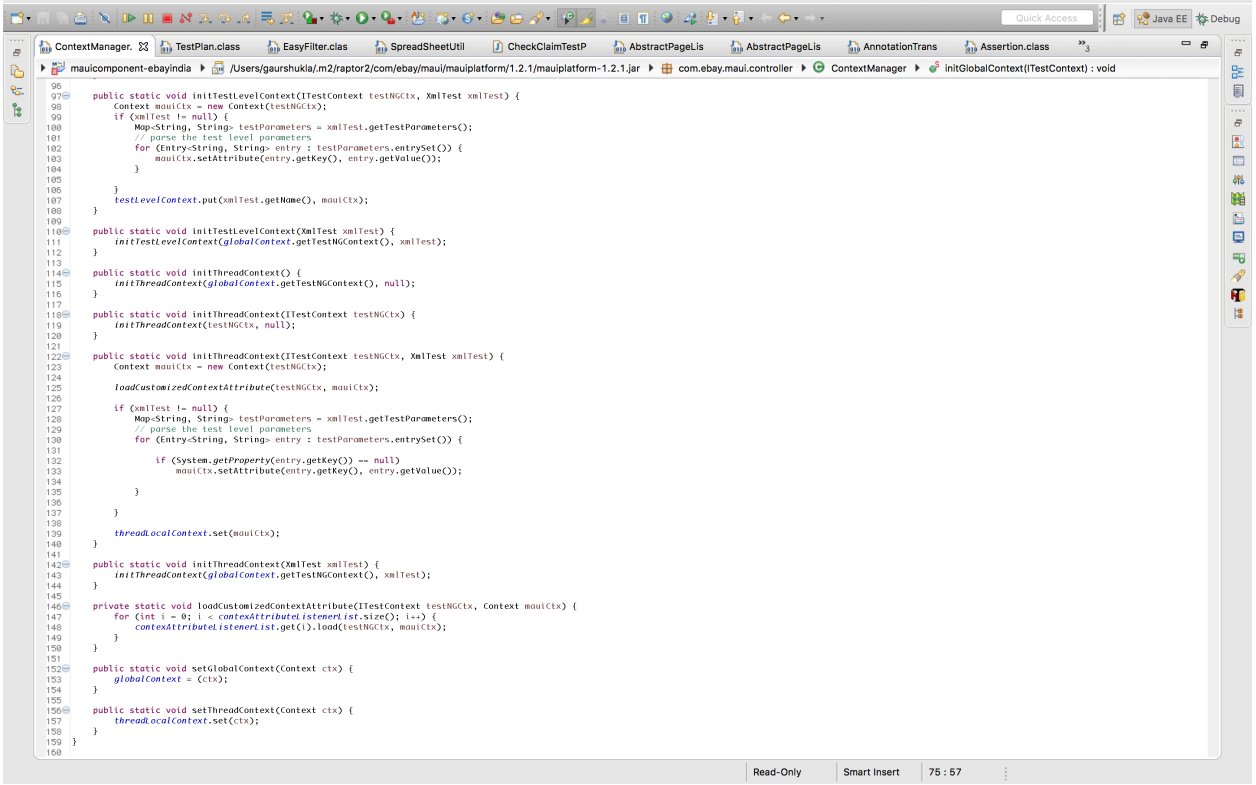

ContextManager.
TestPlan.class
EasyFilter.clas
SpreadSheetUtil
CheckClaimTestP
AbstractPageLis
AbstractPageLis
AnnotationTrans
Assertion.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.controller
Assertion
Read-Only
Smart Insert
14 : 4

Context:

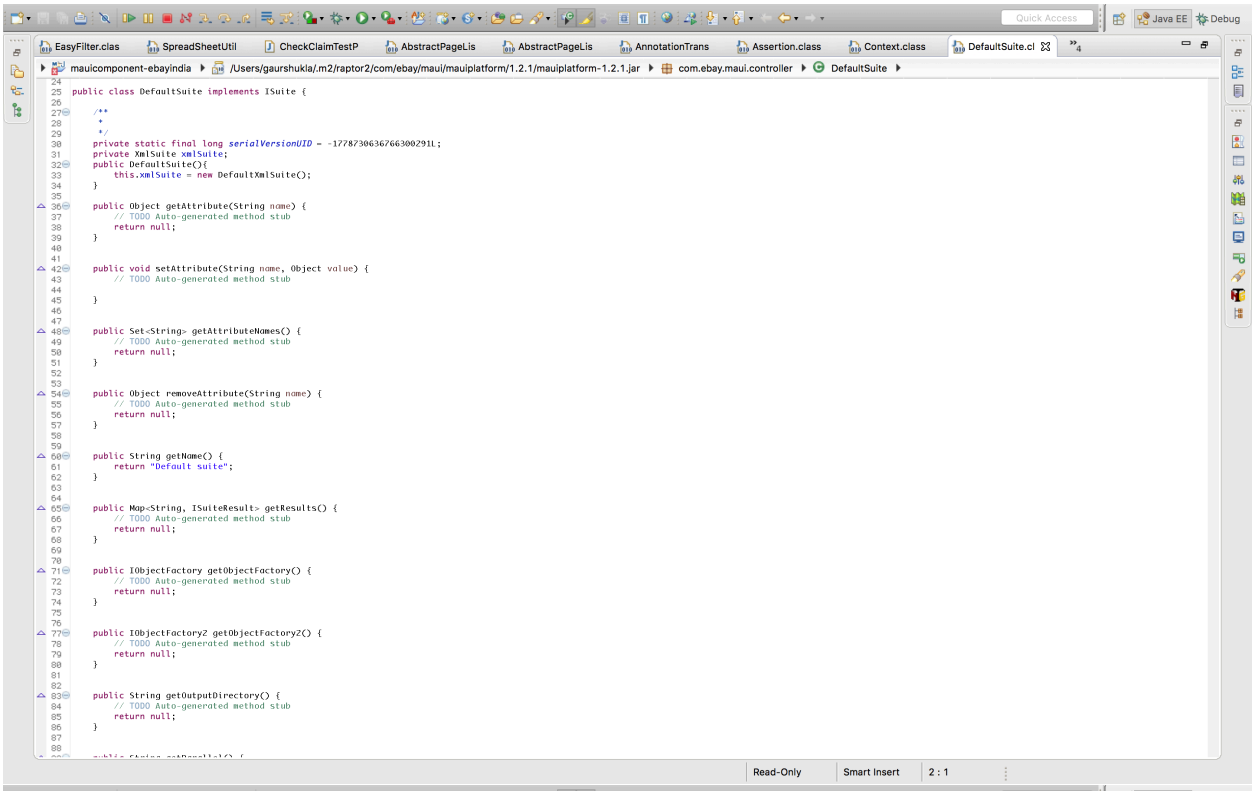

ContextManager.
TestPlan.class
EasyFilter.clas
SpreadSheetUtil
CheckClaimTestP
AbstractPageLis
AbstractPageLis
AnnotationTrans
Assertion.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.controller
ContextManager
initGlobalContext(ITestContext) : void
Read-Only
Smart Insert
75 : 57



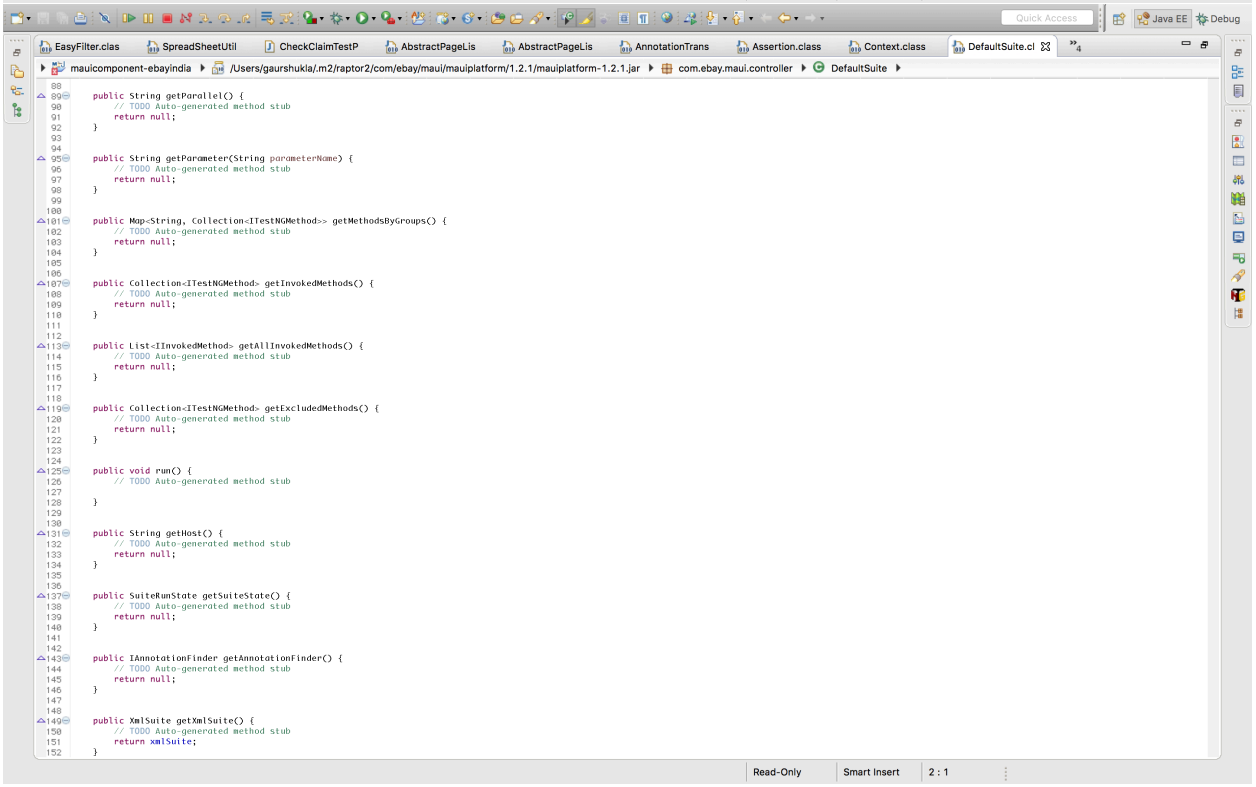

```
    }

    private static ITestContext getContextFromConfigFile(ITestContext testContex) {
        if (testContex != null) {
            if (testContex.getSuite().getParameter(Context.TEST_CONFIG) != null) {
                File suiteFile = new File(testContex.getSuite().getXmlSuite().getFileName());
                String configFile = suiteFile.getPath().replace(suiteFile.getName(), "") + testContex.getSuite().getParameter("testConfig");
                NodeList nList = XMLHelper.getXMLNodes(configFile, "parameter");
                Map<String, String> parameters = testContex.getSuite().getXmlSuite().getParameters();
                for (int i = 0; i < nList.getLength(); i++) {
                    Node nNode = nList.item(i);
                    parameters.put(nNode.getAttributes().getNamedItem("name").getNodeValue(), nNode.getAttributes().getNamedItem("value").getNodeValue());
                }
                testContex.getSuite().getXmlSuite().setParameters(parameters);
            }
        }
        return testContex;
    }

    public static void initTestLevelContext(ITestContext testNGCtx, XmlTest xmlTest) {
        Context mauiCtx = new Context(testNGCtx);
        if (xmlTest != null) {
            Map<String, String> testParameters = xmlTest.getTestParameters();
            // parse the test level parameters
            for (Entry<String, String> entry : testParameters.entrySet()) {
                mauiCtx.setAttribute(entry.getKey(), entry.getValue());
            }

        }
        testLevelContext.put(xmlTest.getName(), mauiCtx);
    }

    public static void initTestLevelContext(XmlTest xmlTest) {
        initTestLevelContext(globalContext.getTestNGContext(), xmlTest);
    }

    public static void initThreadContext() {
        initThreadContext(globalContext.getTestNGContext(), null);
    }

    public static void initThreadContext(ITestContext testNGCtx) {
        initThreadContext(testNGCtx, null);
    }

    public static void initThreadContext(ITestContext testNGCtx, XmlTest xmlTest) {
        Context mauiCtx = new Context(testNGCtx);

        loadCustomizedContextAttribute(testNGCtx, mauiCtx);

        if (xmlTest != null) {
            Map<String, String> testParameters = xmlTest.getTestParameters();
            // parse the test level parameters
            for (Entry<String, String> entry : testParameters.entrySet()) {

                if (System.getProperty(entry.getKey()) == null)
                    mauiCtx.setAttribute(entry.getKey(), entry.getValue());

            }

        }

        threadLocalContext.set(mauiCtx);
    }

    public static void initThreadContext(XmlTest xmlTest) {
```



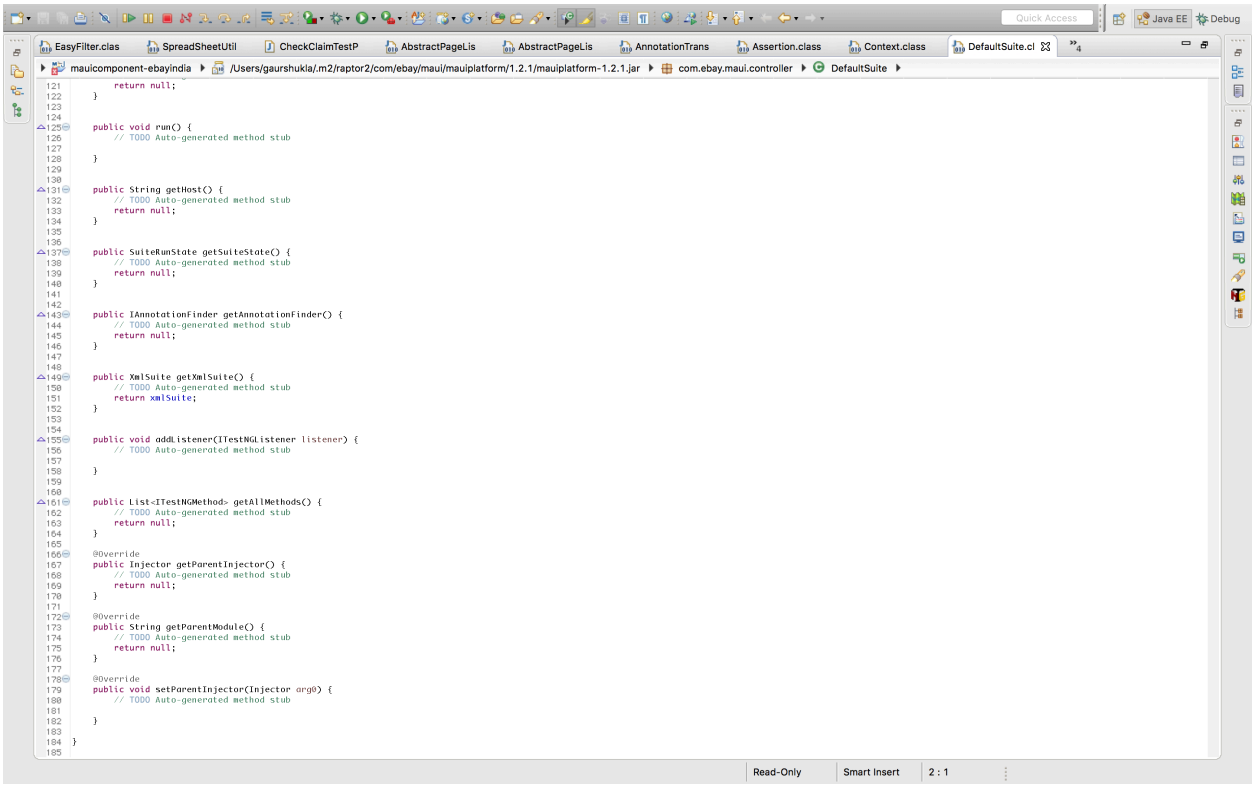

```
    public static void initTestLevelContext(ITestContext testNGCtx, XmlTest xmlTest) {
        Context mauiCtx = new Context(testNGCtx);
        if (xmlTest != null) {
            Map<String, String> testParameters = xmlTest.getTestParameters();
            // parse the test level parameters
            for (Entry<String, String> entry : testParameters.entrySet()) {
                mauiCtx.setAttribute(entry.getKey(), entry.getValue());
            }

        }
        testLevelContext.put(xmlTest.getName(), mauiCtx);
    }

    public static void initTestLevelContext(XmlTest xmlTest) {
        initTestLevelContext(globalContext.getTestNGContext(), xmlTest);
    }

    public static void initThreadContext() {
        initThreadContext(globalContext.getTestNGContext(), null);
    }

    public static void initThreadContext(ITestContext testNGCtx) {
        initThreadContext(testNGCtx, null);
    }

    public static void initThreadContext(ITestContext testNGCtx, XmlTest xmlTest) {
        Context mauiCtx = new Context(testNGCtx);

        loadCustomizedContextAttribute(testNGCtx, mauiCtx);

        if (xmlTest != null) {
            Map<String, String> testParameters = xmlTest.getTestParameters();
            // parse the test level parameters
            for (Entry<String, String> entry : testParameters.entrySet()) {

                if (System.getProperty(entry.getKey()) == null)
                    mauiCtx.setAttribute(entry.getKey(), entry.getValue());

            }

        }

        threadLocalContext.set(mauiCtx);
    }

    public static void initThreadContext(XmlTest xmlTest) {
        initThreadContext(globalContext.getTestNGContext(), xmlTest);
    }

    private static void loadCustomizedContextAttribute(ITestContext testNGCtx, Context mauiCtx) {
        for (int i = 0; i < contexAttributeListenerList.size(); i++) {
            contexAttributeListenerList.get(i).load(testNGCtx, mauiCtx);
        }
    }

    public static void setGlobalContext(Context ctx) {
        globalContext = (ctx);
    }

    public static void setThreadContext(Context ctx) {
        threadLocalContext.set(ctx);
    }
}
```



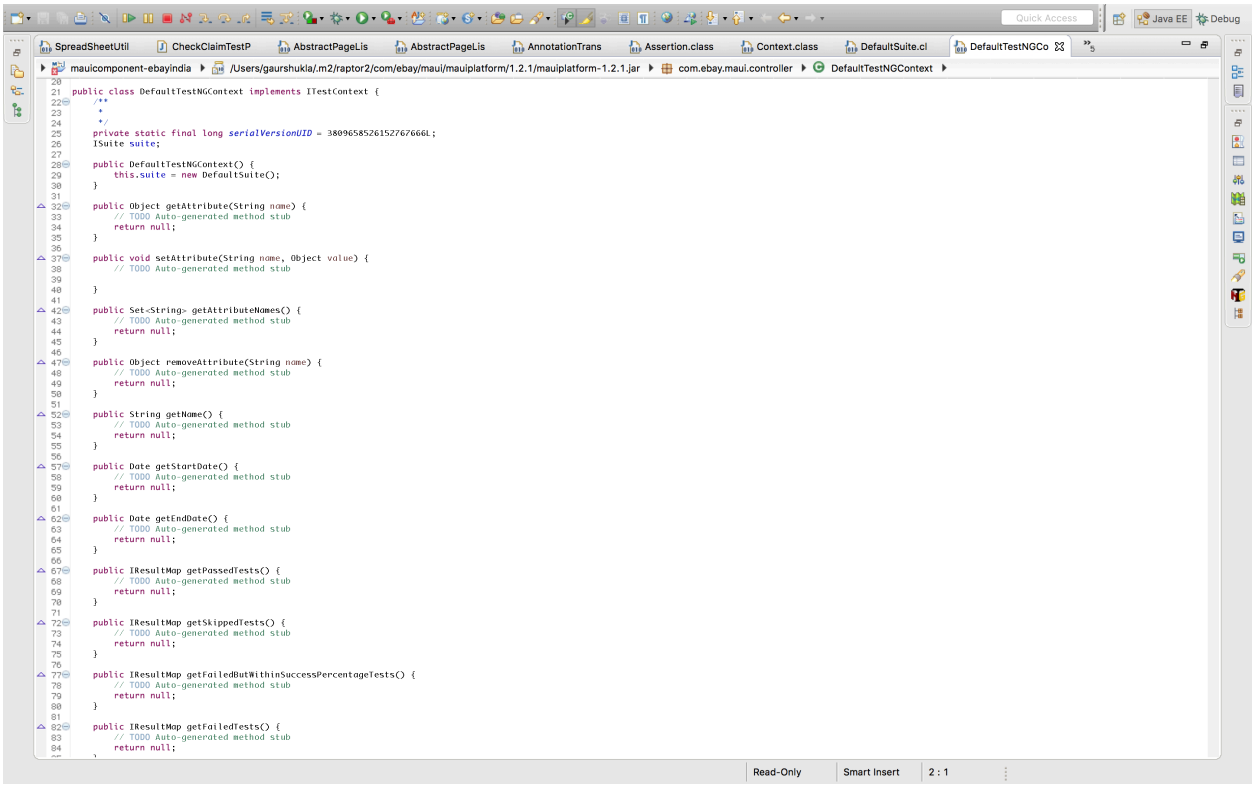
```
24
25  public class DefaultSuite implements ISuite {
26
27      /**
28       *
29       */
30      private static final long serialVersionUID = -1778730636766300291L;
31      private XmlSuite xmlSuite;
32      public DefaultSuite(){
33          this.xmlSuite = new DefaultXmlSuite();
34      }
35
36      public Object getAttribute(String name) {
37          // TODO Auto-generated method stub
38          return null;
39      }
40
41
42      public void setAttribute(String name, Object value) {
43          // TODO Auto-generated method stub
44
45      }
46
47
48      public Set<String> getAttributeNames() {
49          // TODO Auto-generated method stub
50          return null;
51      }
52
53
54      public Object removeAttribute(String name) {
55          // TODO Auto-generated method stub
56          return null;
57      }
58
59
60      public String getName() {
61          return "Default suite";
62      }
63
64
65      public Map<String, ISuiteResult> getResults() {
66          // TODO Auto-generated method stub
67          return null;
68      }
69
70
71      public IObjectFactory getObjectFactory() {
72          // TODO Auto-generated method stub
73          return null;
74      }
75
76
77      public IObjectFactory2 getObjectFactory2() {
78          // TODO Auto-generated method stub
79          return null;
80      }
81
82
83      public String getOutputDirectory() {
84          // TODO Auto-generated method stub
85          return null;
86      }
87
88
```



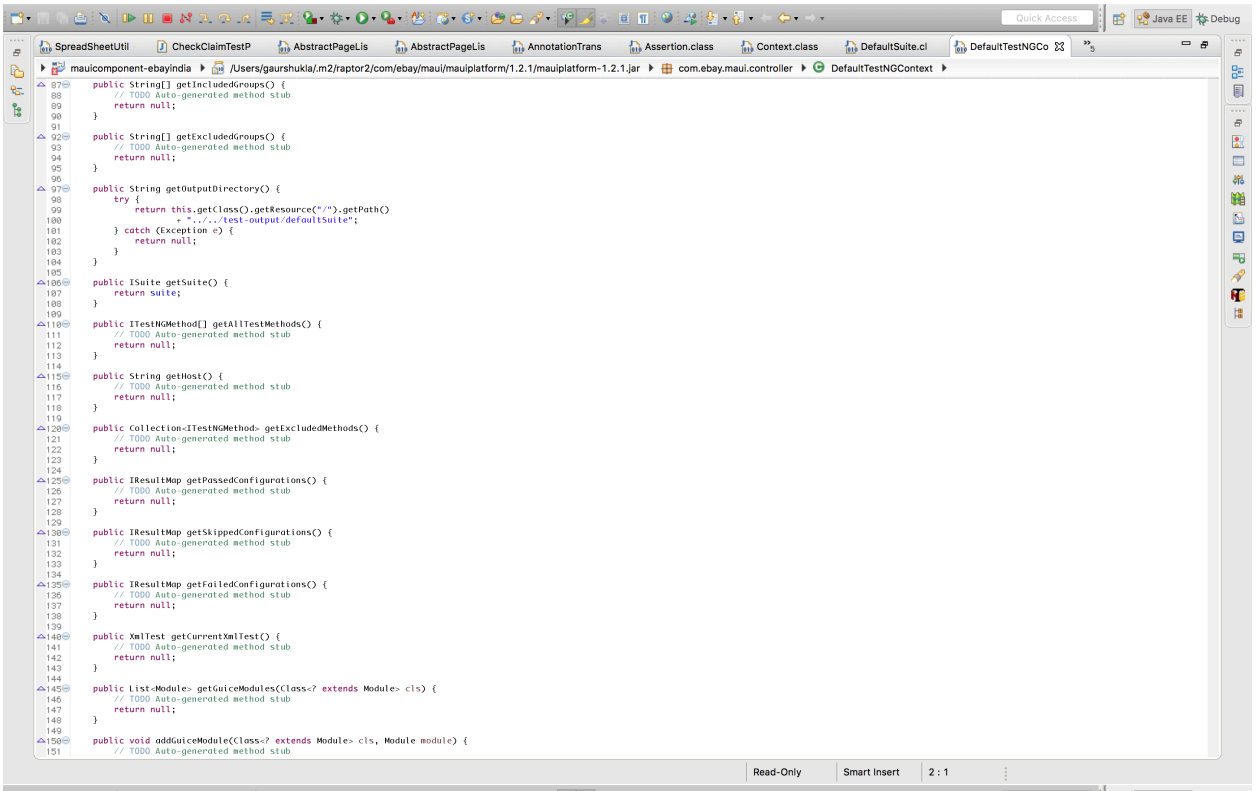
```
88
89      public String getParallel() {
90          // TODO Auto-generated method stub
91          return null;
92      }
93
94
95      public String getParameter(String parameterName) {
96          // TODO Auto-generated method stub
97          return null;
98      }
99
100
101     public Map<String, Collection<ITestNGMethod>> getMethodsByGroups() {
102         // TODO Auto-generated method stub
103         return null;
104     }
105
106
107     public Collection<ITestNGMethod> getInvokedMethods() {
108         // TODO Auto-generated method stub
109         return null;
110     }
111
112
113     public List<IInvokedMethod> getAllInvokedMethods() {
114         // TODO Auto-generated method stub
115         return null;
116     }
117
118
119     public Collection<ITestNGMethod> getExcludedMethods() {
120         // TODO Auto-generated method stub
121         return null;
122     }
123
124
125     public void run() {
126         // TODO Auto-generated method stub
127
128     }
129
130
131     public String getHost() {
132         // TODO Auto-generated method stub
133         return null;
134     }
135
136
137     public SuiteRunState getSuiteState() {
138         // TODO Auto-generated method stub
139         return null;
140     }
141
142
143     public IAnnotationFinder getAnnotationFinder() {
144         // TODO Auto-generated method stub
145         return null;
146     }
147
148
149     public XmlSuite getXmlSuite() {
150         // TODO Auto-generated method stub
151         return xmlSuite;
152     }
```

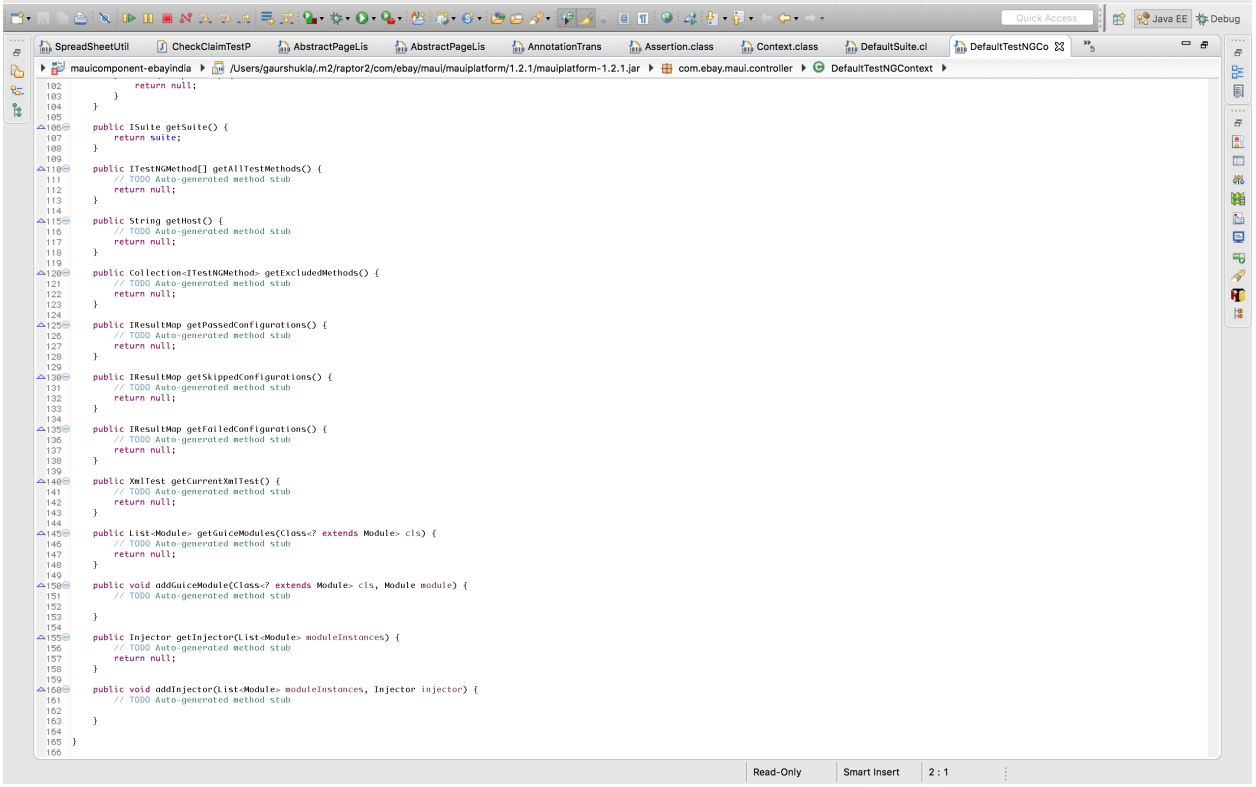


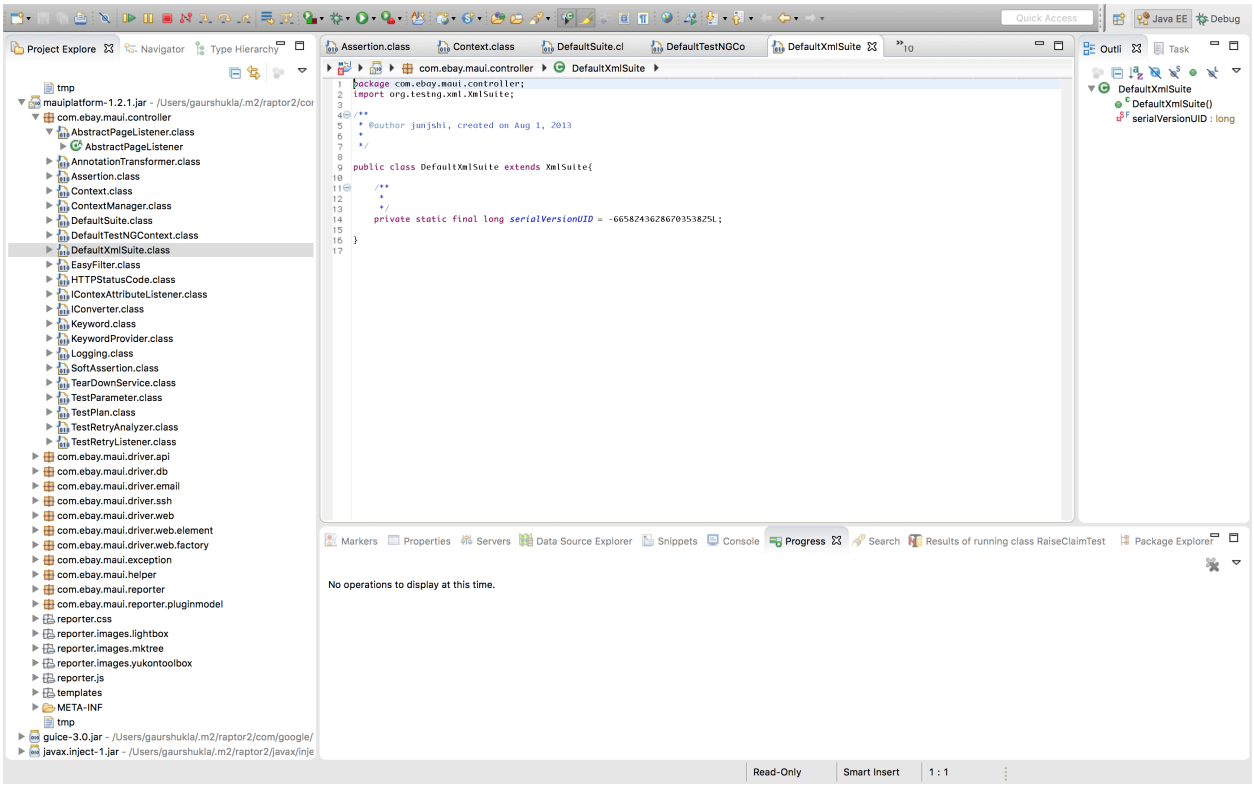
```
        return null;
    }

    public void run() {
        // TODO Auto-generated method stub

    }

    public String getHost() {
        // TODO Auto-generated method stub
        return null;
    }

    public SuiteRunState getSuiteState() {
        // TODO Auto-generated method stub
        return null;
    }

    public IAnnotationFinder getAnnotationFinder() {
        // TODO Auto-generated method stub
        return null;
    }

    public XmlSuite getXmlSuite() {
        // TODO Auto-generated method stub
        return xmlSuite;
    }

    public void addListener(ITestNGListener listener) {
        // TODO Auto-generated method stub

    }

    public List<ITestNGMethod> getAllMethods() {
        // TODO Auto-generated method stub
        return null;
    }
    @Override
    public Injector getParentInjector() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public String getParentModule() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void setParentInjector(Injector arg0) {
        // TODO Auto-generated method stub

    }
}
```

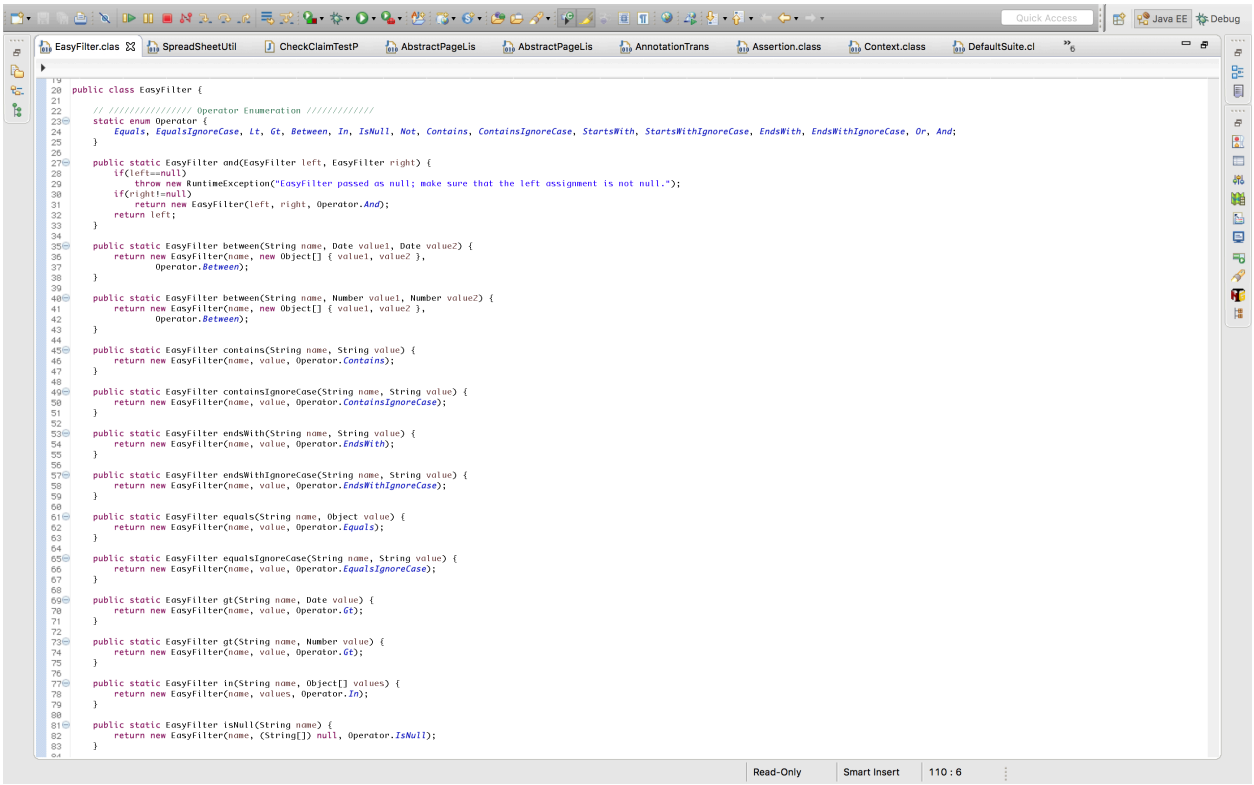


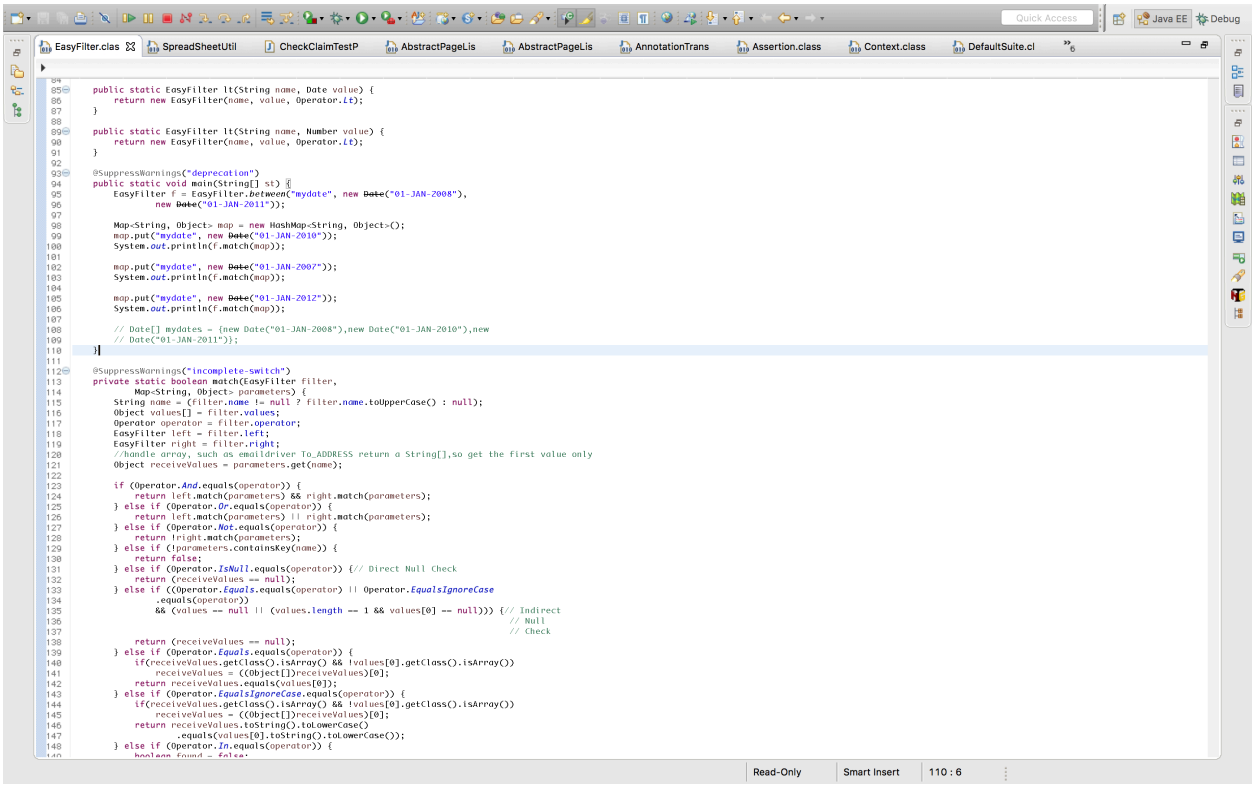
```
public class DefaultTestNGContext implements ITestContext {
    /**
     *
     */
    private static final long serialVersionUID = 3809658526152767666L;
    ISuite suite;

    public DefaultTestNGContext() {
        this.suite = new DefaultSuite();
    }

    public Object getAttribute(String name) {
        // TODO Auto-generated method stub
        return null;
    }

    public void setAttribute(String name, Object value) {
        // TODO Auto-generated method stub

    }

    public Set<String> getAttributeNames() {
        // TODO Auto-generated method stub
        return null;
    }

    public Object removeAttribute(String name) {
        // TODO Auto-generated method stub
        return null;
    }

    public String getName() {
        // TODO Auto-generated method stub
        return null;
    }

    public Date getStartDate() {
        // TODO Auto-generated method stub
        return null;
    }

    public Date getEndDate() {
        // TODO Auto-generated method stub
        return null;
    }

    public IResultMap getPassedTests() {
        // TODO Auto-generated method stub
        return null;
    }

    public IResultMap getSkippedTests() {
        // TODO Auto-generated method stub
        return null;
    }

    public IResultMap getFailedButWithinSuccessPercentageTests() {
        // TODO Auto-generated method stub
        return null;
    }

    public IResultMap getFailedTests() {
        // TODO Auto-generated method stub
        return null;
```

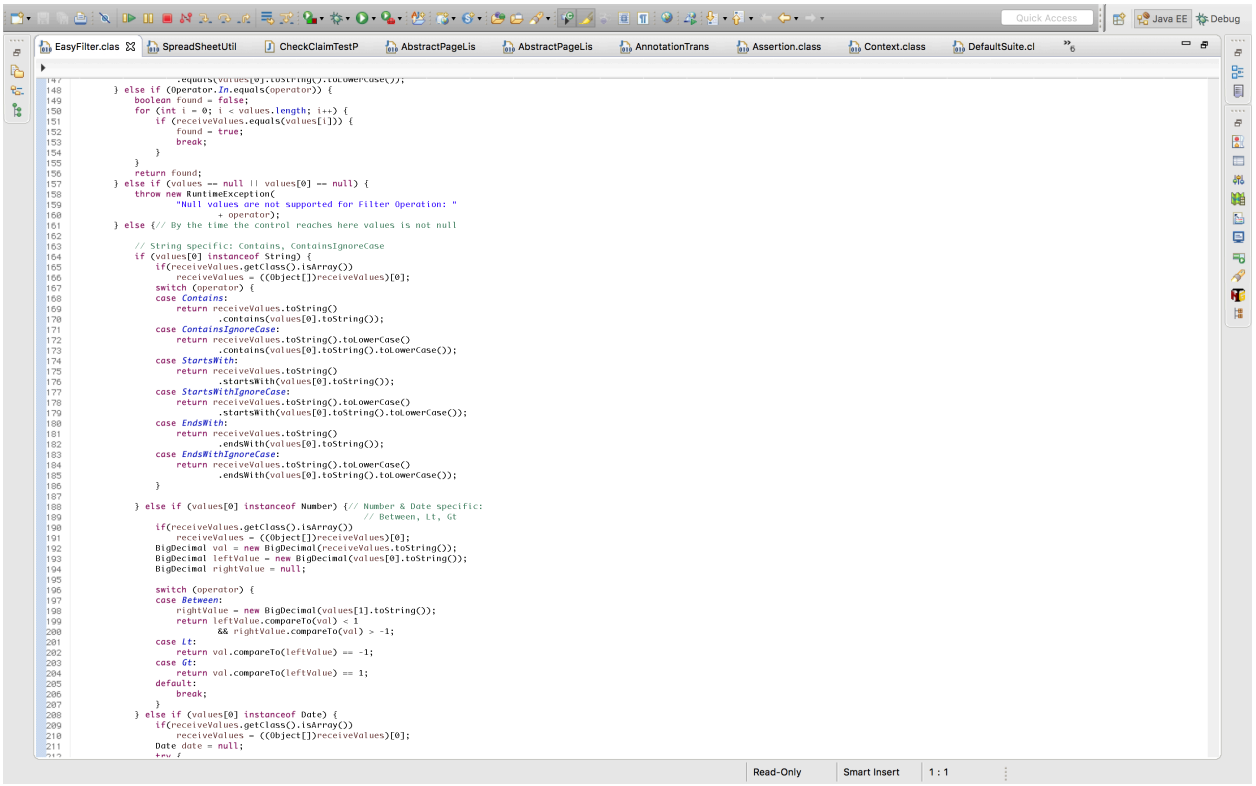

SpreadSheetUtil
CheckClaimTestP
AbstractPageLis
AbstractPageLis
AnnotationTrans
Assertion.class
Context.class
DefaultSuite.cl
DefaultTestNGCo
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.controller
DefaultTestNGContext
Read-Only
Smart Insert
2 : 1

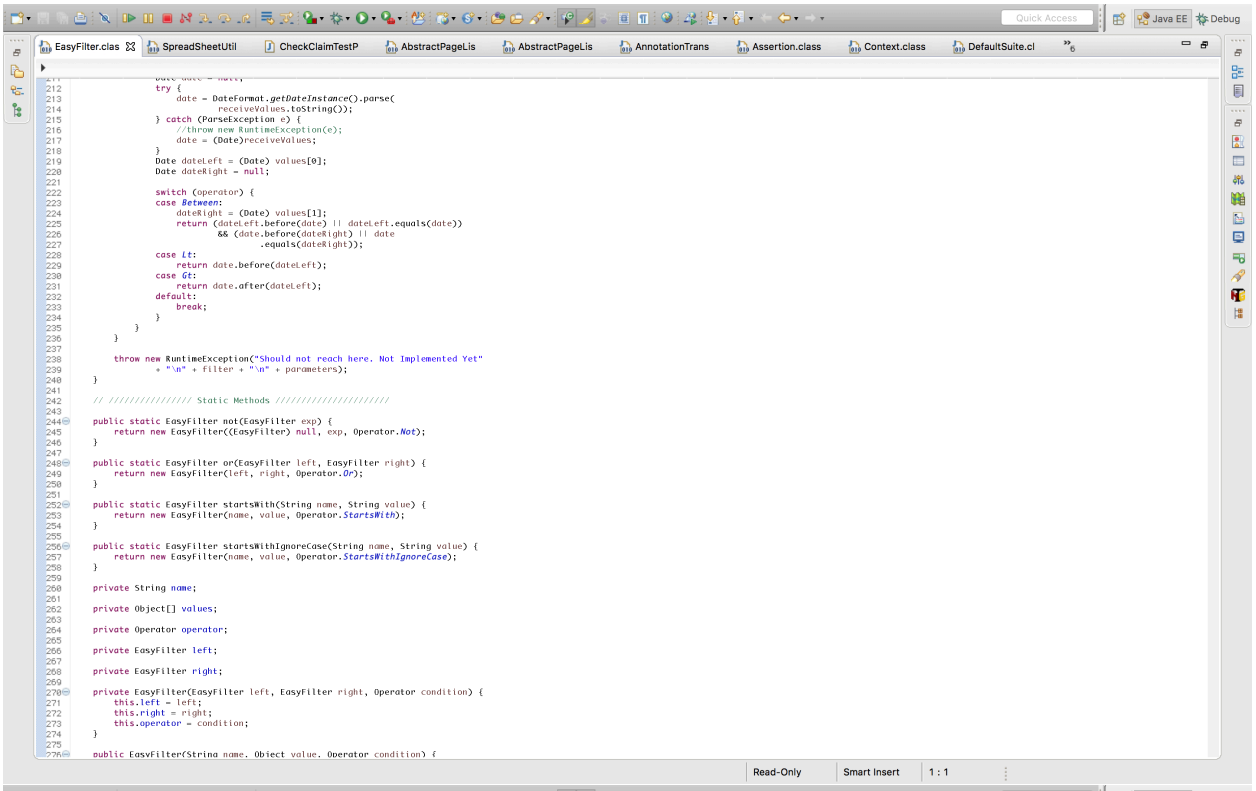

SpreadSheetUtil
CheckClaimTestP
AbstractPageLis
AbstractPageLis
AnnotationTrans
Assertion.class
Context.class
DefaultSuite.cl
DefaultTestNGCo
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.controller
DefaultTestNGContext
Read-Only
Smart Insert
2 : 1

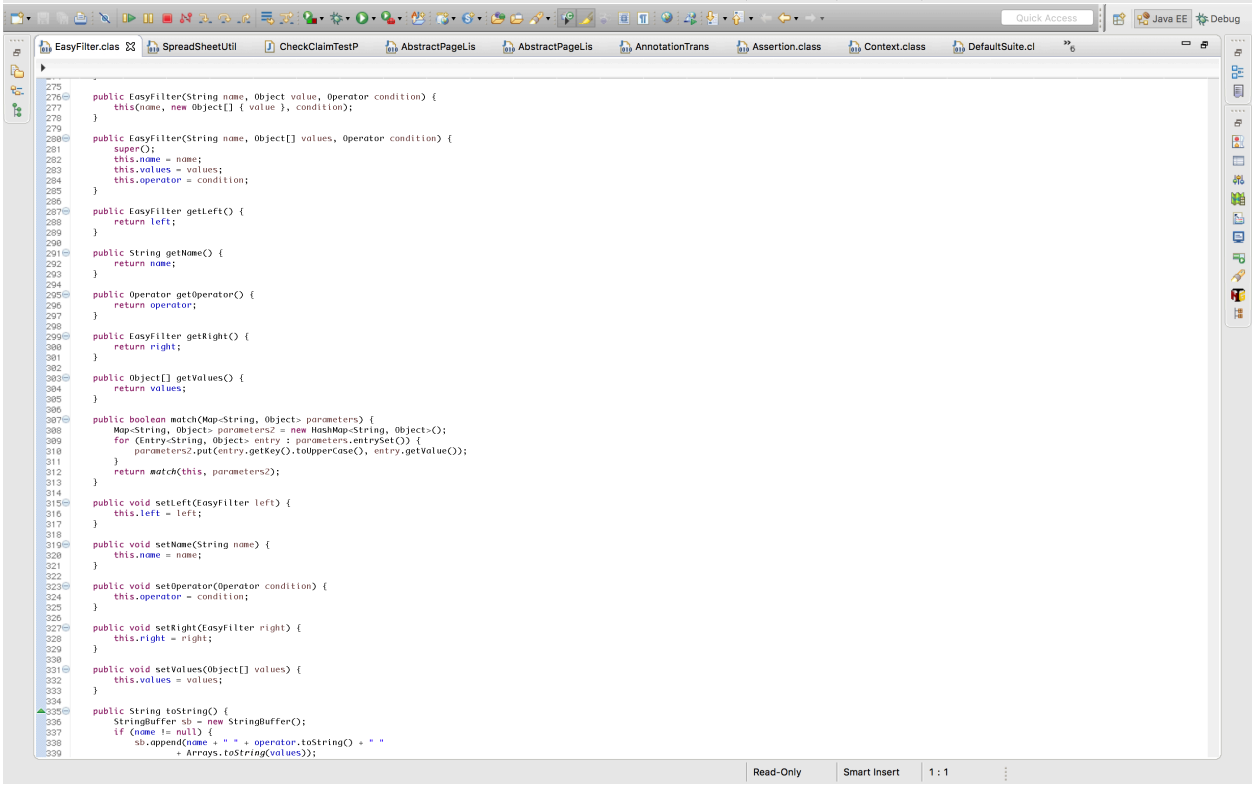

Project Explore
Navigator
Type Hierarchy
tmp
mauiplatform-1.2.1.jar - /Users/gaurshukla/.m2/raptor2/cor
com.ebay.maui.controller
AbstractPageListener.class
AbstractPageListener
AnnotationTransformer.class
Assertion.class
Context.class
ContextManager.class
DefaultSuite.class
DefaultTestNGContext.class
DefaultXmlSuite.class
EasyFilter.class
HTTPStatusCode.class
IContextAttributeListener.class
IConverter.class
Keyword.class
KeywordProvider.class
Logging.class
SoftAssertion.class
TearDownService.class
TestParameter.class
TestPlan.class
TestRetryAnalyzer.class
TestRetryListener.class
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
com.ebay.maui.driver.web.factory
com.ebay.maui.exception
com.ebay.maui.helper
com.ebay.maui.reporter
com.ebay.maui.reporter.pluginmodel
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/inje
Assertion.class
Context.class
DefaultSuite.cl
DefaultTestNGCo
DefaultXmlSuite
com.ebay.maui.controller
DefaultXmlSuite
package com.ebay.maui.controller;
import org.testng.xml.XmlSuite;
/**
* @author junjshi, created on Aug 1, 2013
*
*/
public class DefaultXmlSuite extends XmlSuite{
/**
*
*/
private static final long serialVersionUID = -6658243628670353825L;
}
Outli
Task
DefaultXmlSuite
DefaultXmlSuite()
serialVersionUID : long
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

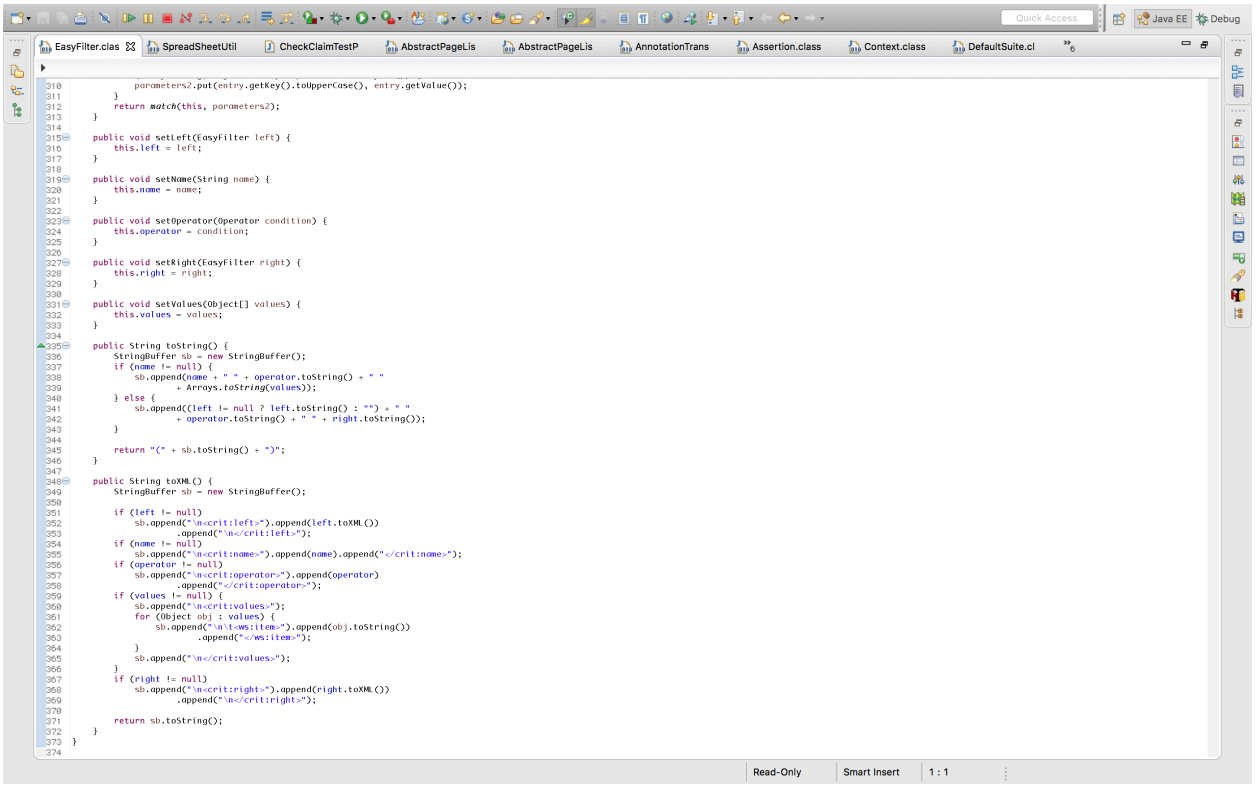

EasyFilter.clas
SpreadSheetUtil
CheckClaimTestP
AbstractPageLis
AbstractPageLis
AnnotationTrans
Assertion.class
Context.class
DefaultSuite.cl
public class EasyFilter {
// ================ Operator Enumeration =============
static enum Operator {
Equals, EqualsIgnoreCase, Lt, Gt, Between, In, IsNull, Not, Contains, ContainsIgnoreCase, StartsWith, StartsWithIgnoreCase, EndsWith, EndsWithIgnoreCase, Or, And;
}
public static EasyFilter and(EasyFilter left, EasyFilter right) {
if(left==null)
throw new RuntimeException("EasyFilter passed as null; make sure that the left assignment is not null.");
if(right!=null)
return new EasyFilter(left, right, Operator.And);
return left;
}
public static EasyFilter between(String name, Date value1, Date value2) {
return new EasyFilter(name, new Object[] { value1, value2 },
Operator.Between);
}
public static EasyFilter between(String name, Number value1, Number value2) {
return new EasyFilter(name, new Object[] { value1, value2 },
Operator.Between);
}
public static EasyFilter contains(String name, String value) {
return new EasyFilter(name, value, Operator.Contains);
}
public static EasyFilter containsIgnoreCase(String name, String value) {
return new EasyFilter(name, value, Operator.ContainsIgnoreCase);
}
public static EasyFilter endsWith(String name, String value) {
return new EasyFilter(name, value, Operator.EndsWith);
}
public static EasyFilter endsWithIgnoreCase(String name, String value) {
return new EasyFilter(name, value, Operator.EndsWithIgnoreCase);
}
public static EasyFilter equals(String name, Object value) {
return new EasyFilter(name, value, Operator.Equals);
}
public static EasyFilter equalsIgnoreCase(String name, String value) {
return new EasyFilter(name, value, Operator.EqualsIgnoreCase);
}
public static EasyFilter gt(String name, Date value) {
return new EasyFilter(name, value, Operator.Gt);
}
public static EasyFilter gt(String name, Number value) {
return new EasyFilter(name, value, Operator.Gt);
}
public static EasyFilter in(String name, Object[] values) {
return new EasyFilter(name, values, Operator.In);
}
public static EasyFilter isNull(String name) {
return new EasyFilter(name, (String[]) null, Operator.IsNull);
}
Read-Only
Smart Insert
110 : 6


EasyFilter.clas
SpreadSheetUtil
CheckClaimTestP
AbstractPageLis
AbstractPageLis
AnnotationTrans
Assertion.class
Context.class
DefaultSuite.cl
Quick Access
Java EE
Debug
Read-Only
Smart Insert
110 : 6

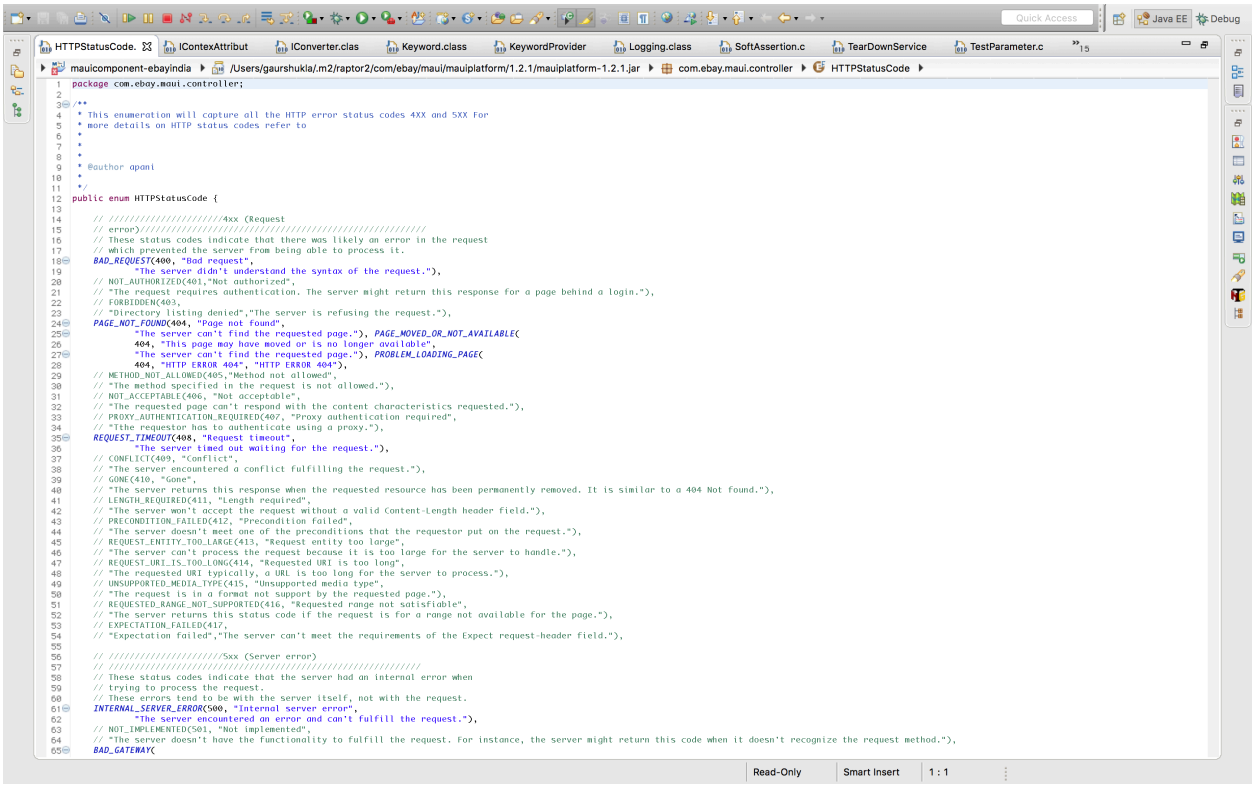

EasyFilter.clas
SpreadSheetUtil
CheckClaimTestP
AbstractPageLis
AbstractPageLis
AnnotationTrans
Assertion.class
Context.class
DefaultSuite.cl
Quick Access
Java EE
Debug
Read-Only
Smart Insert
1 : 1


EasyFilter.clas
SpreadSheetUtil
CheckClaimTestP
AbstractPageLis
AbstractPageLis
AnnotationTrans
Assertion.class
Context.class
DefaultSuite.cl
Quick Access
Java EE
Debug
Read-Only
Smart Insert
1 : 1

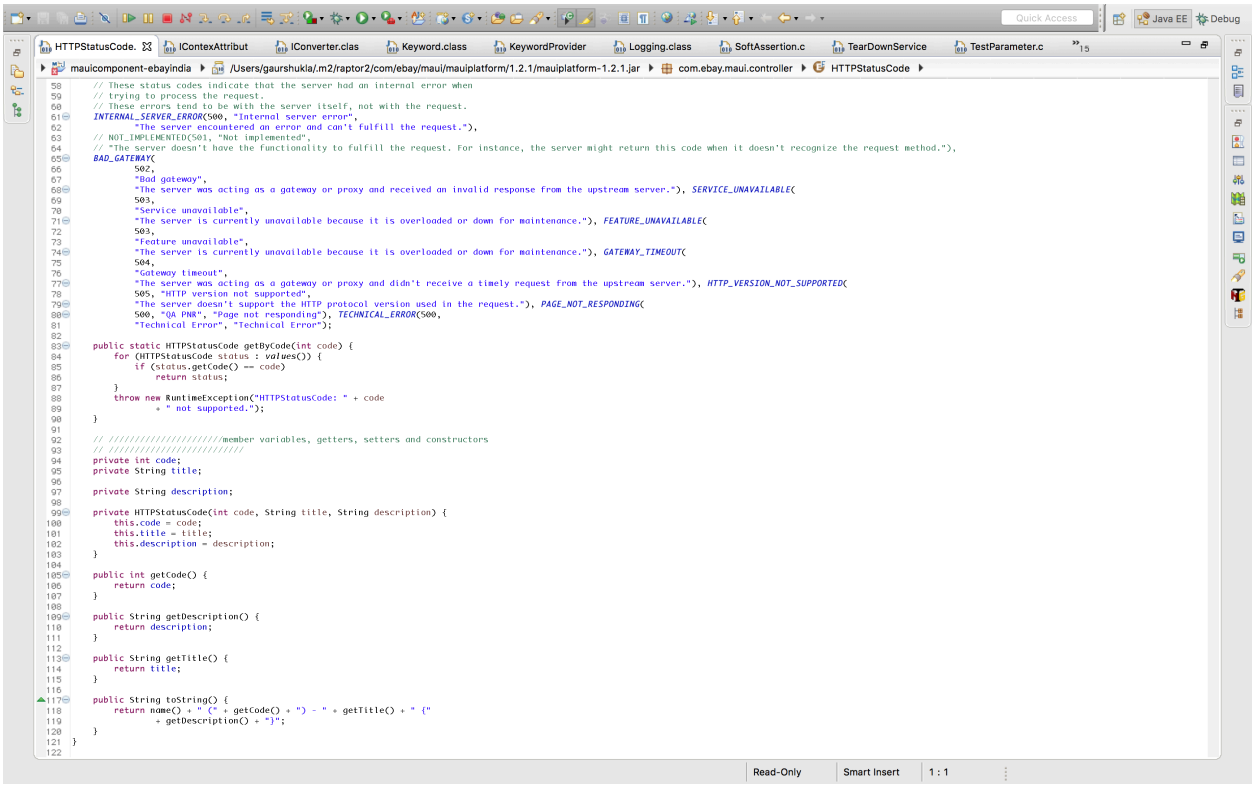

EasyFilter.clas
SpreadSheetUtil
CheckClaimTestP
AbstractPageLis
AbstractPageLis
AnnotationTrans
Assertion.class
Context.class
DefaultSuite.cl
Quick Access
Java EE
Debug
Read-Only
Smart Insert
1 : 1



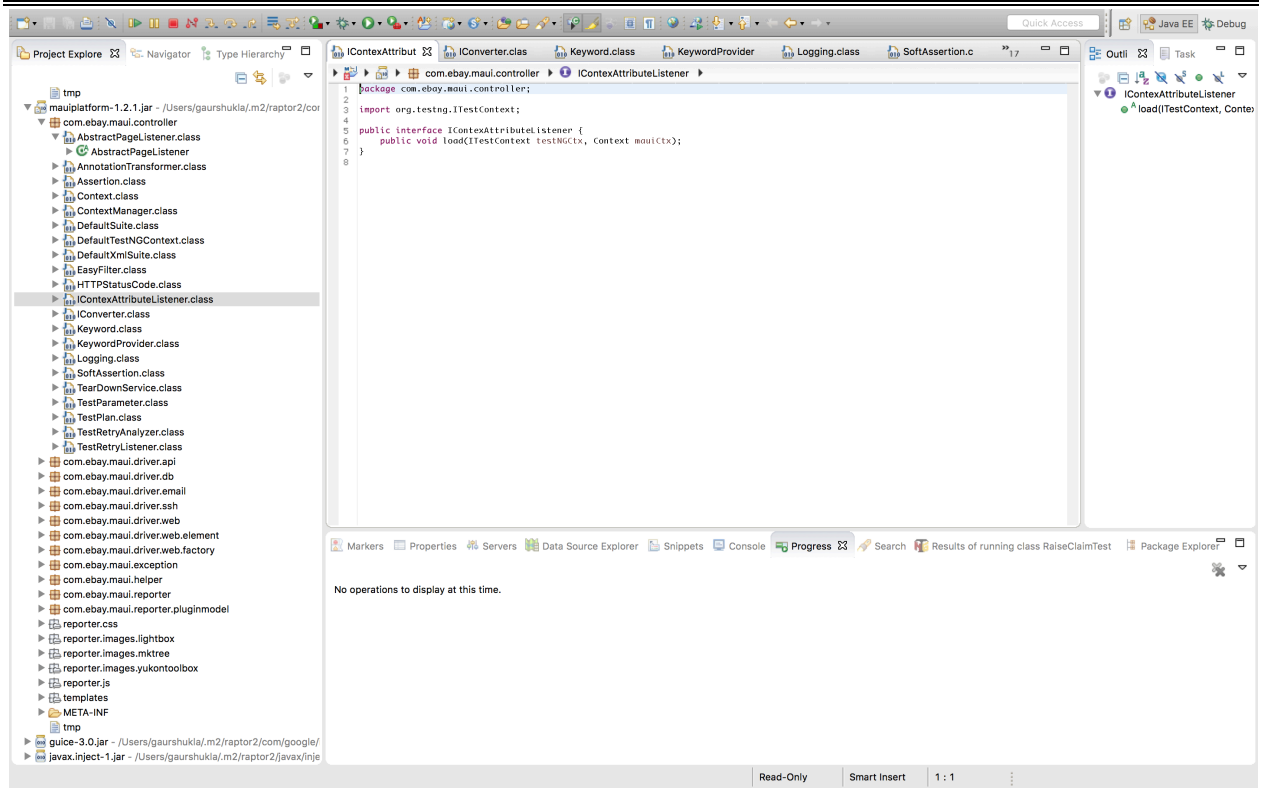
```
310             parameters2.put(entry.getKey().toUpperCase(), entry.getValue());
311         }
312         return match(this, parameters2);
313     }
314
315     public void setLeft(EasyFilter left) {
316         this.left = left;
317     }
318
319     public void setName(String name) {
320         this.name = name;
321     }
322
323     public void setOperator(Operator condition) {
324         this.operator = condition;
325     }
326
327     public void setRight(EasyFilter right) {
328         this.right = right;
329     }
330
331     public void setValues(Object[] values) {
332         this.values = values;
333     }
334
335     public String toString() {
336         StringBuffer sb = new StringBuffer();
337         if (name != null) {
338             sb.append(name + " " + operator.toString() + " "
339                     + Arrays.toString(values));
340         } else {
341             sb.append((left != null ? left.toString() : "") + " "
342                     + operator.toString() + " " + right.toString());
343         }
344
345         return "(" + sb.toString() + ")";
346     }
347
348     public String toXML() {
349         StringBuffer sb = new StringBuffer();
350
351         if (left != null)
352             sb.append("\n<crit:left>").append(left.toXML())
353                     .append("\n</crit:left>");
354         if (name != null)
355             sb.append("\n<crit:name>").append(name).append("</crit:name>");
356         if (operator != null)
357             sb.append("\n<crit:operator>").append(operator)
358                     .append("</crit:operator>");
359         if (values != null) {
360             sb.append("\n<crit:values>");
361             for (Object obj : values) {
362                 sb.append("\n\t<ws:item>").append(obj.toString())
363                         .append("</ws:item>");
364             }
365             sb.append("\n</crit:values>");
366         }
367         if (right != null)
368             sb.append("\n<crit:right>").append(right.toXML())
369                     .append("\n</crit:right>");
370
371         return sb.toString();
372     }
373 }
374
```



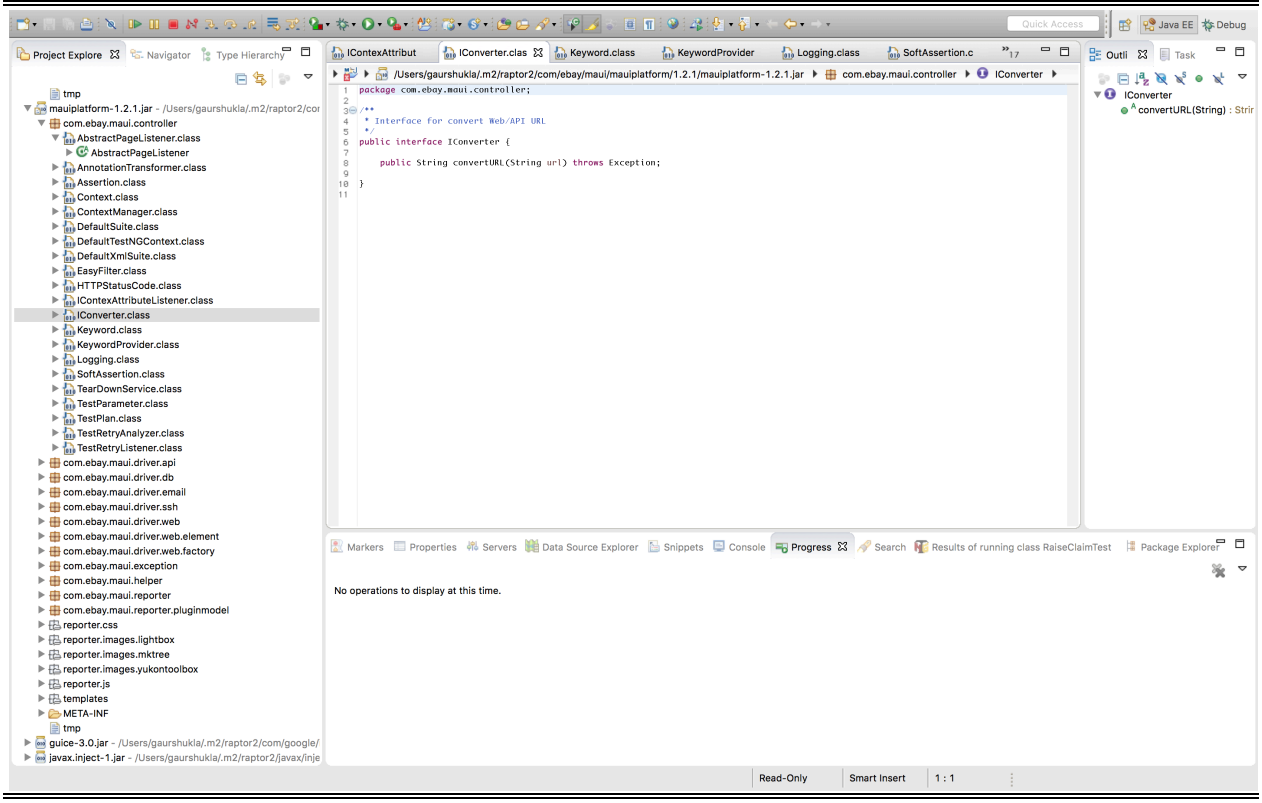
```
1  package com.ebay.maui.controller;
2
3  /**
4   * This enumeration will capture all the HTTP error status codes 4XX and 5XX for
5   * more details on HTTP status codes refer to
6   *
7   *
8   *
9   * @author apani
10  *
11  */
12 public enum HTTPStatusCode {
13
14      // ////////////////////4xx (Request
15      // error)////////////////////////////////////////////////////////
16      // These status codes indicate that there was likely an error in the request
17      // which prevented the server from being able to process it.
18      BAD_REQUEST(400, "Bad request",
19              "The server didn't understand the syntax of the request."),
20      // NOT_AUTHORIZED(401,"Not authorized",
21      // "The request requires authentication. The server might return this response for a page behind a login."),
22      // FORBIDDEN(403,
23      // "Directory listing denied","The server is refusing the request."),
24      PAGE_NOT_FOUND(404, "Page not found",
25              "The server can't find the requested page."), PAGE_MOVED_OR_NOT_AVAILABLE(
26              404, "This page may have moved or is no longer available",
27              "The server can't find the requested page."), PROBLEM_LOADING_PAGE(
28              404, "HTTP ERROR 404", "HTTP ERROR 404"),
29      // METHOD_NOT_ALLOWED(405,"Method not allowed",
30      // "The method specified in the request is not allowed."),
31      // NOT_ACCEPTABLE(406, "Not acceptable",
32      // "The requested page can't respond with the content characteristics requested."),
33      // PROXY_AUTHENTICATION_REQUIRED(407, "Proxy authentication required",
34      // "Tthe requestor has to authenticate using a proxy."),
35      REQUEST_TIMEOUT(408, "Request timeout",
36              "The server timed out waiting for the request."),
37      // CONFLICT(409, "Conflict",
38      // "The server encountered a conflict fulfilling the request."),
39      // GONE(410, "Gone",
40      // "The server returns this response when the requested resource has been permanently removed. It is similar to a 404 Not found."),
41      // LENGTH_REQUIRED(411, "Length required",
42      // "The server won't accept the request without a valid Content-Length header field."),
43      // PRECONDITION_FAILED(412, "Precondition failed",
44      // "The server doesn't meet one of the preconditions that the requestor put on the request."),
45      // REQUEST_ENTITY_TOO_LARGE(413, "Request entity too large",
46      // "The server can't process the request because it is too large for the server to handle."),
47      // REQUEST_URI_IS_TOO_LONG(414, "Requested URI is too long",
48      // "The requested URI typically, a URL is too long for the server to process."),
49      // UNSUPPORTED_MEDIA_TYPE(415, "Unsupported media type",
50      // "The request is in a format not support by the requested page."),
51      // REQUESTED_RANGE_NOT_SUPPORTED(416, "Requested range not satisfiable",
52      // "The server returns this status code if the request is for a range not available for the page."),
53      // EXPECTATION_FAILED(417,
54      // "Expectation failed","The server can't meet the requirements of the Expect request-header field."),
55
56      // ////////////////////5xx (Server error)
57      // ////////////////////////////////////////////////////////////////
58      // These status codes indicate that the server had an internal error when
59      // trying to process the request.
60      // These errors tend to be with the server itself, not with the request.
61      INTERNAL_SERVER_ERROR(500, "Internal server error",
62              "The server encountered an error and can't fulfill the request."),
63      // NOT_IMPLEMENTED(501, "Not implemented",
64      // "The server doesn't have the functionality to fulfill the request. For instance, the server might return this code when it doesn't recognize the request method."),
65      BAD_GATEWAY(
```

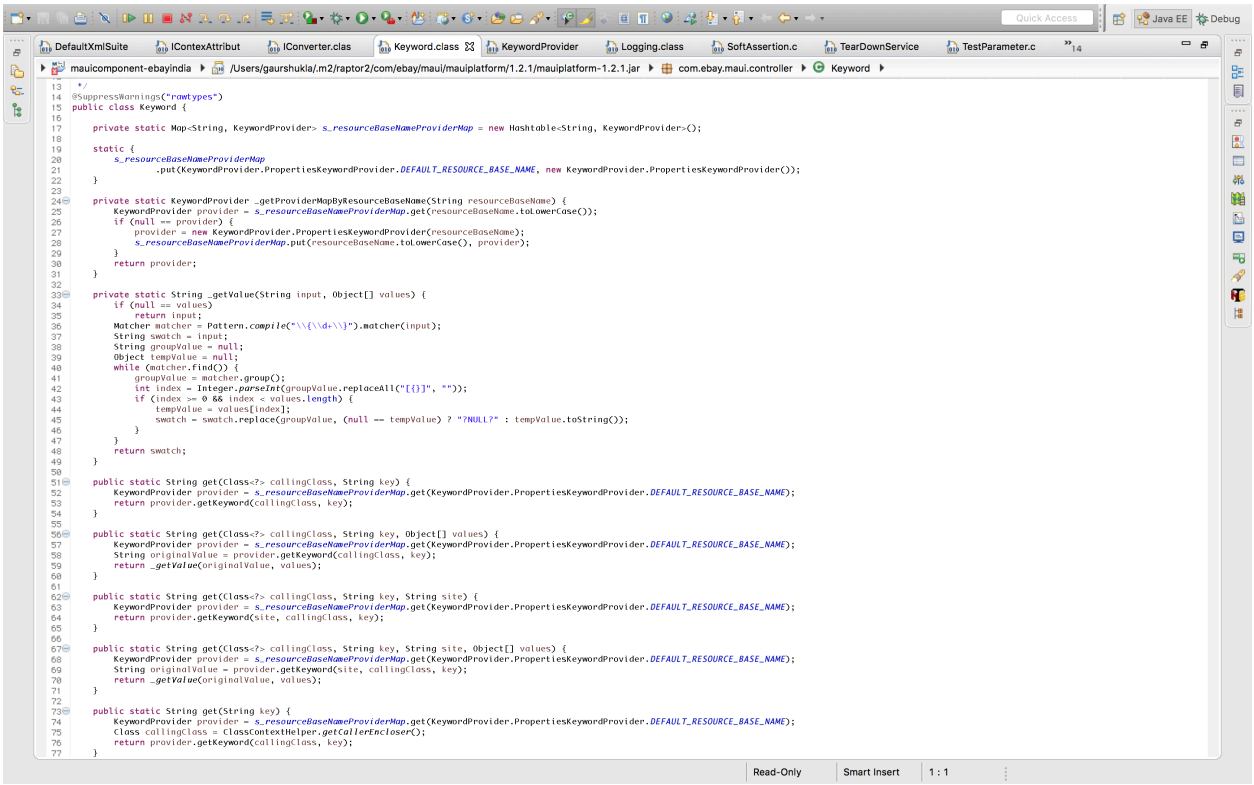


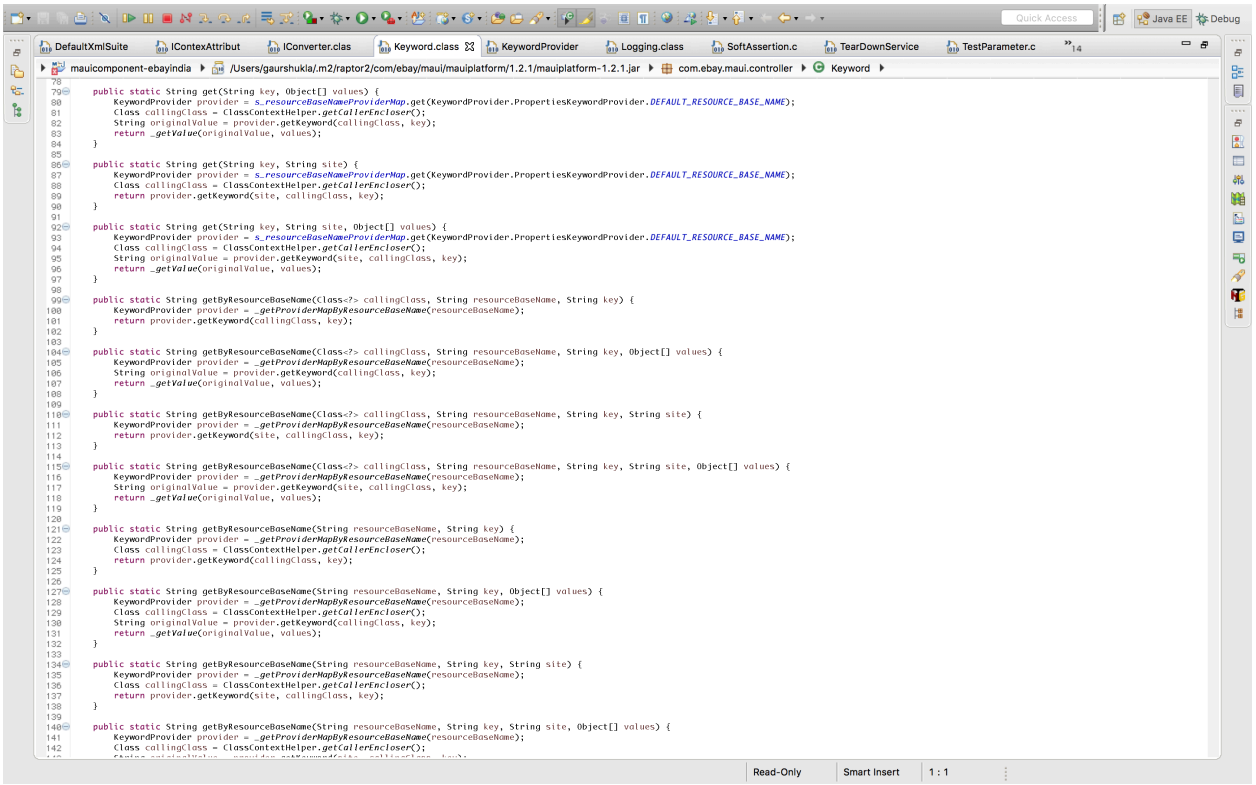
```
// These status codes indicate that the server had an internal error when
// trying to process the request.
// These errors tend to be with the server itself, not with the request.
INTERNAL_SERVER_ERROR(500, "Internal server error",
        "The server encountered an error and can't fulfill the request."),
// NOT_IMPLEMENTED(501, "Not implemented",
// "The server doesn't have the functionality to fulfill the request. For instance, the server might return this code when it doesn't recognize the request method."),
BAD_GATEWAY(
        502,
        "Bad gateway",
        "The server was acting as a gateway or proxy and received an invalid response from the upstream server."), SERVICE_UNAVAILABLE(
        503,
        "Service unavailable",
        "The server is currently unavailable because it is overloaded or down for maintenance."), FEATURE_UNAVAILABLE(
        503,
        "Feature unavailable",
        "The server is currently unavailable because it is overloaded or down for maintenance."), GATEWAY_TIMEOUT(
        504,
        "Gateway timeout",
        "The server was acting as a gateway or proxy and didn't receive a timely request from the upstream server."), HTTP_VERSION_NOT_SUPPORTED(
        505, "HTTP version not supported",
        "The server doesn't support the HTTP protocol version used in the request."), PAGE_NOT_RESPONDING(
        500, "QA PNR", "Page not responding"), TECHNICAL_ERROR(500,
        "Technical Error", "Technical Error");

public static HTTPStatusCode getByCode(int code) {
    for (HTTPStatusCode status : values()) {
        if (status.getCode() == code)
            return status;
    }
    throw new RuntimeException("HTTPStatusCode: " + code
            + " not supported.");
}

// ////////////////////member variables, getters, setters and constructors
// ////////////////////////
private int code;
private String title;

private String description;

private HTTPStatusCode(int code, String title, String description) {
    this.code = code;
    this.title = title;
    this.description = description;
}

public int getCode() {
    return code;
}

public String getDescription() {
    return description;
}

public String getTitle() {
    return title;
}

public String toString() {
    return name() + " (" + getCode() + ") - " + getTitle() + " {"
            + getDescription() + "}";
}
}
```

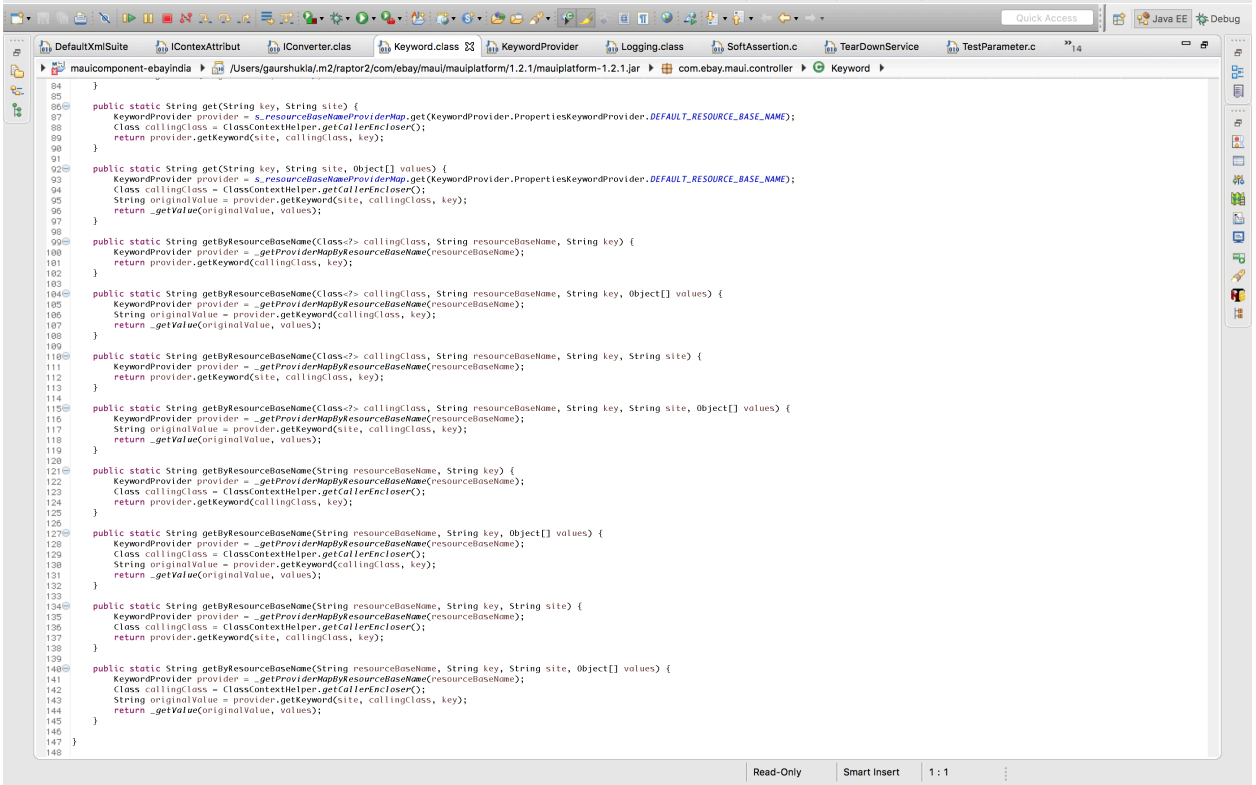


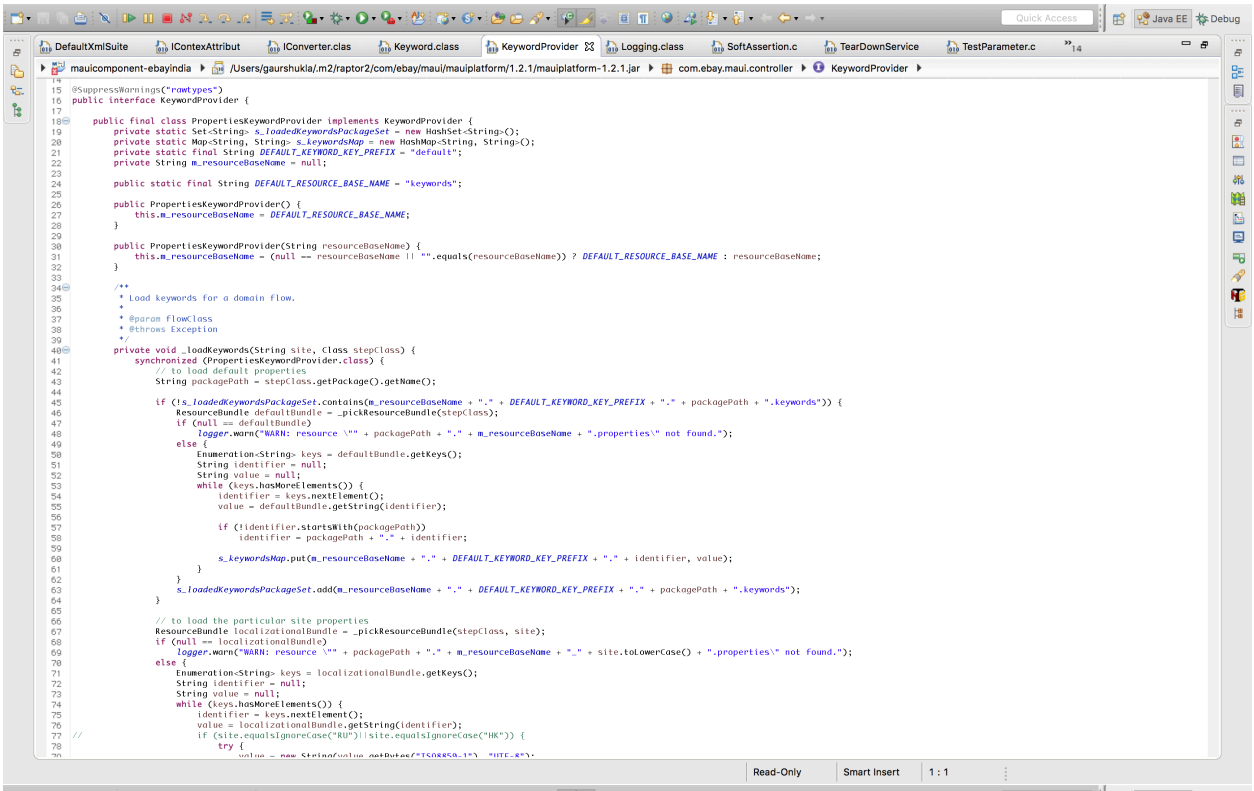
```
package com.ebay.maui.controller;

import org.testng.ITestContext;

public interface IContextAttributeListener {
    public void load(ITestContext testNGCtx, Context mauiCtx);
}
```


Project Explore
Navigator
Type Hierarchy
tmp
mauiplatform-1.2.1.jar - /Users/gaurshukla/.m2/raptor2/cor
com.ebay.maui.controller
AbstractPageListener.class
AbstractPageListener
AnnotationTransformer.class
Assertion.class
Context.class
ContextManager.class
DefaultSuite.class
DefaultTestNGContext.class
DefaultXmlSuite.class
EasyFilter.class
HTTPStatusCode.class
IContexAttributeListener.class
IConverter.class
Keyword.class
KeywordProvider.class
Logging.class
SoftAssertion.class
TearDownService.class
TestParameter.class
TestPlan.class
TestRetryAnalyzer.class
TestRetryListener.class
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
com.ebay.maui.driver.web.factory
com.ebay.maui.exception
com.ebay.maui.helper
com.ebay.maui.reporter
com.ebay.maui.reporter.pluginmodel
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/inje
IContexAttribut
IConverter.clas
Keyword.class
KeywordProvider
Logging.class
SoftAssertion.c
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.controller
IConverter
package com.ebay.maui.controller;
/**
* Interface for convert Web/API URL
*/
public interface IConverter {
public String convertURL(String url) throws Exception;
}
Outli
Task
IConverter
convertURL(String) : Strir
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

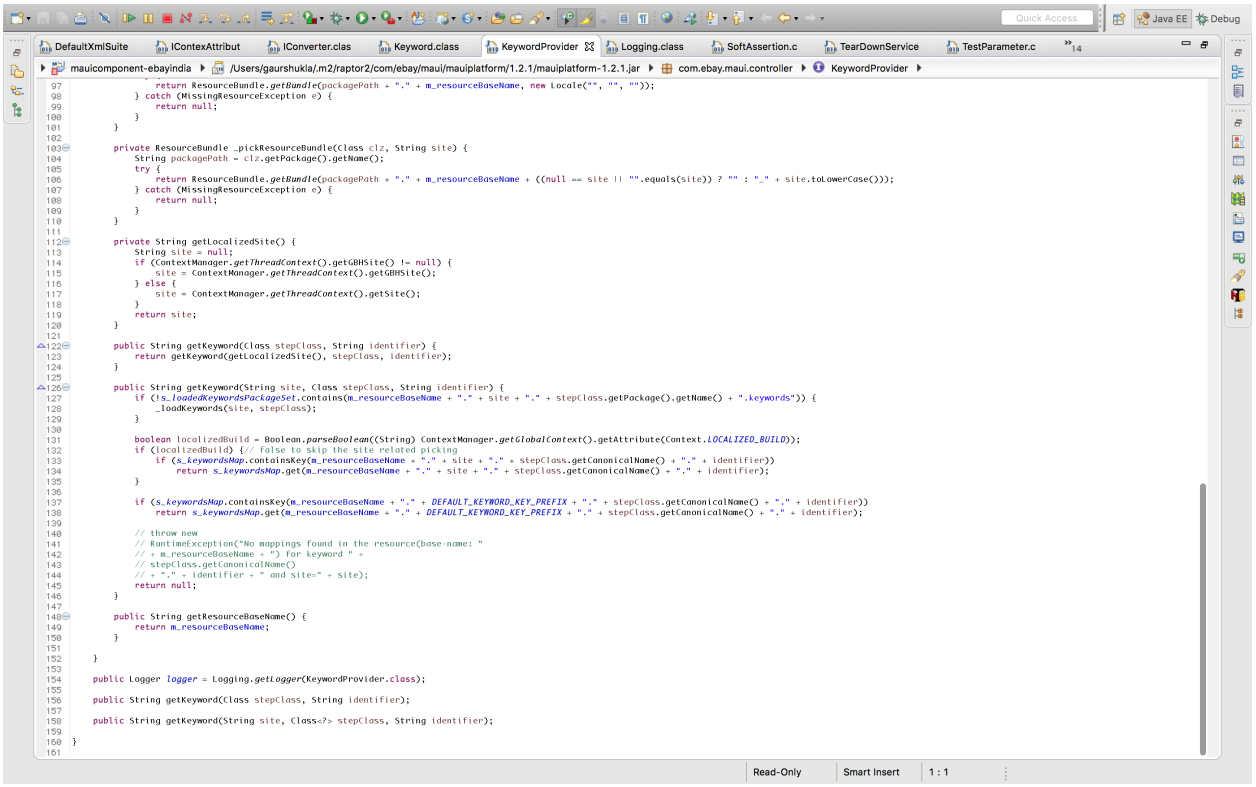

DefaultXmlSuite
IContexAttribut
IConverter.clas
Keyword.class
KeywordProvider
Logging.class
SoftAssertion.c
TearDownService
TestParameter.c
maucomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.controller
Keyword
*/
@SuppressWarnings("rawtypes")
public class Keyword {
private static Map<String, KeywordProvider> s_resourceBaseNameProviderMap = new Hashtable<String, KeywordProvider>();
static {
s_resourceBaseNameProviderMap
.put(KeywordProvider.PropertiesKeywordProvider.DEFAULT_RESOURCE_BASE_NAME, new KeywordProvider.PropertiesKeywordProvider());
}
private static KeywordProvider _getProviderMapByResourceBaseName(String resourceBaseName) {
KeywordProvider provider = s_resourceBaseNameProviderMap.get(resourceBaseName.toLowerCase());
if (null == provider) {
provider = new KeywordProvider.PropertiesKeywordProvider(resourceBaseName);
s_resourceBaseNameProviderMap.put(resourceBaseName.toLowerCase(), provider);
}
return provider;
}
private static String _getValue(String input, Object[] values) {
if (null == values)
return input;
Matcher matcher = Pattern.compile("\\{\\d+\\}").matcher(input);
String swatch = input;
String groupValue = null;
Object tempValue = null;
while (matcher.find()) {
groupValue = matcher.group();
int index = Integer.parseInt(groupValue.replaceAll("[{}]", ""));
if (index >= 0 && index < values.length) {
tempValue = values[index];
swatch = swatch.replace(groupValue, (null == tempValue) ? "?NULL?" : tempValue.toString());
}
}
return swatch;
}
public static String get(Class<?> callingClass, String key) {
KeywordProvider provider = s_resourceBaseNameProviderMap.get(KeywordProvider.PropertiesKeywordProvider.DEFAULT_RESOURCE_BASE_NAME);
return provider.getKeyword(callingClass, key);
}
public static String get(Class<?> callingClass, String key, Object[] values) {
KeywordProvider provider = s_resourceBaseNameProviderMap.get(KeywordProvider.PropertiesKeywordProvider.DEFAULT_RESOURCE_BASE_NAME);
String originalValue = provider.getKeyword(callingClass, key);
return _getValue(originalValue, values);
}
public static String get(Class<?> callingClass, String key, String site) {
KeywordProvider provider = s_resourceBaseNameProviderMap.get(KeywordProvider.PropertiesKeywordProvider.DEFAULT_RESOURCE_BASE_NAME);
return provider.getKeyword(site, callingClass, key);
}
public static String get(Class<?> callingClass, String key, String site, Object[] values) {
KeywordProvider provider = s_resourceBaseNameProviderMap.get(KeywordProvider.PropertiesKeywordProvider.DEFAULT_RESOURCE_BASE_NAME);
String originalValue = provider.getKeyword(site, callingClass, key);
return _getValue(originalValue, values);
}
public static String get(String key) {
KeywordProvider provider = s_resourceBaseNameProviderMap.get(KeywordProvider.PropertiesKeywordProvider.DEFAULT_RESOURCE_BASE_NAME);
Class callingClass = ClassContextHelper.getCallerEncloser();
return provider.getKeyword(callingClass, key);
}
Read-Only
Smart Insert
1 : 1

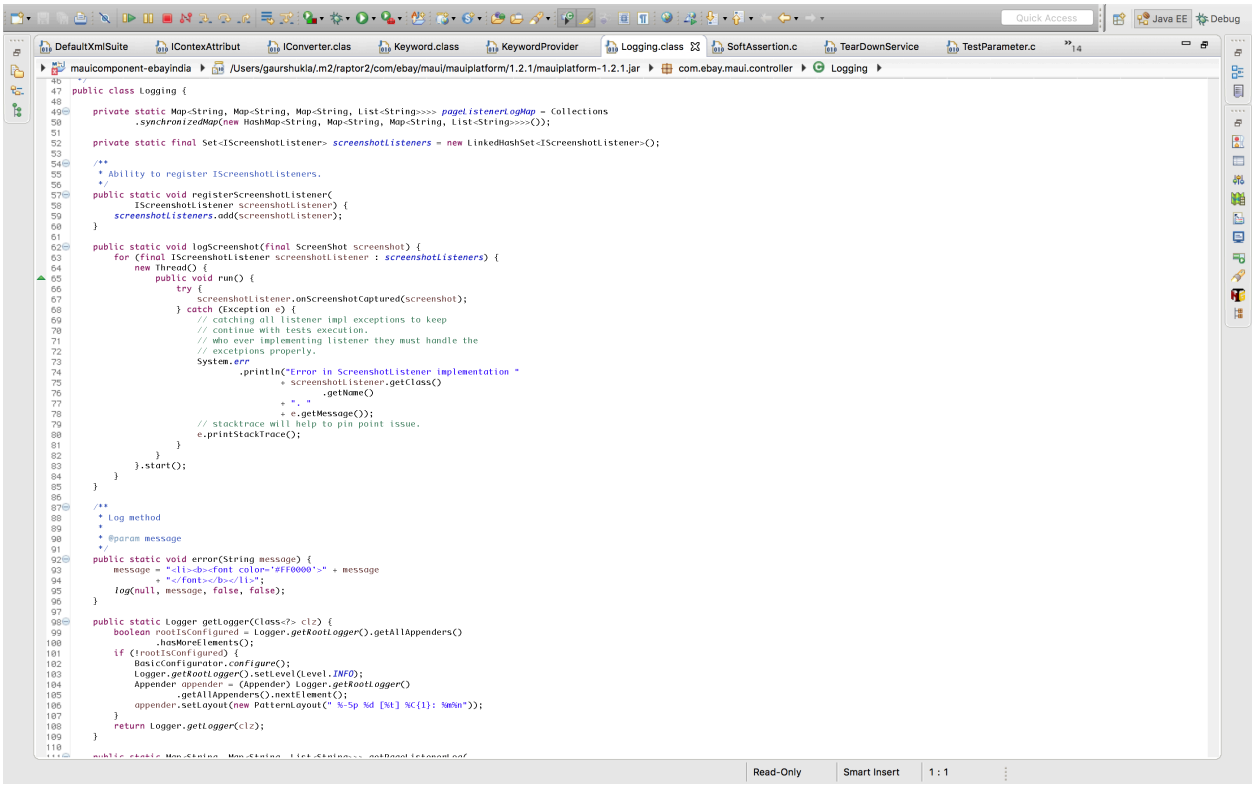

DefaultXmlSuite
IContextAttribut
IConverter.clas
Keyword.class
KeywordProvider
Logging.class
SoftAssertion.c
TearDownService
TestParameter.c
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.controller
Keyword
Read-Only
Smart Insert
1 : 1

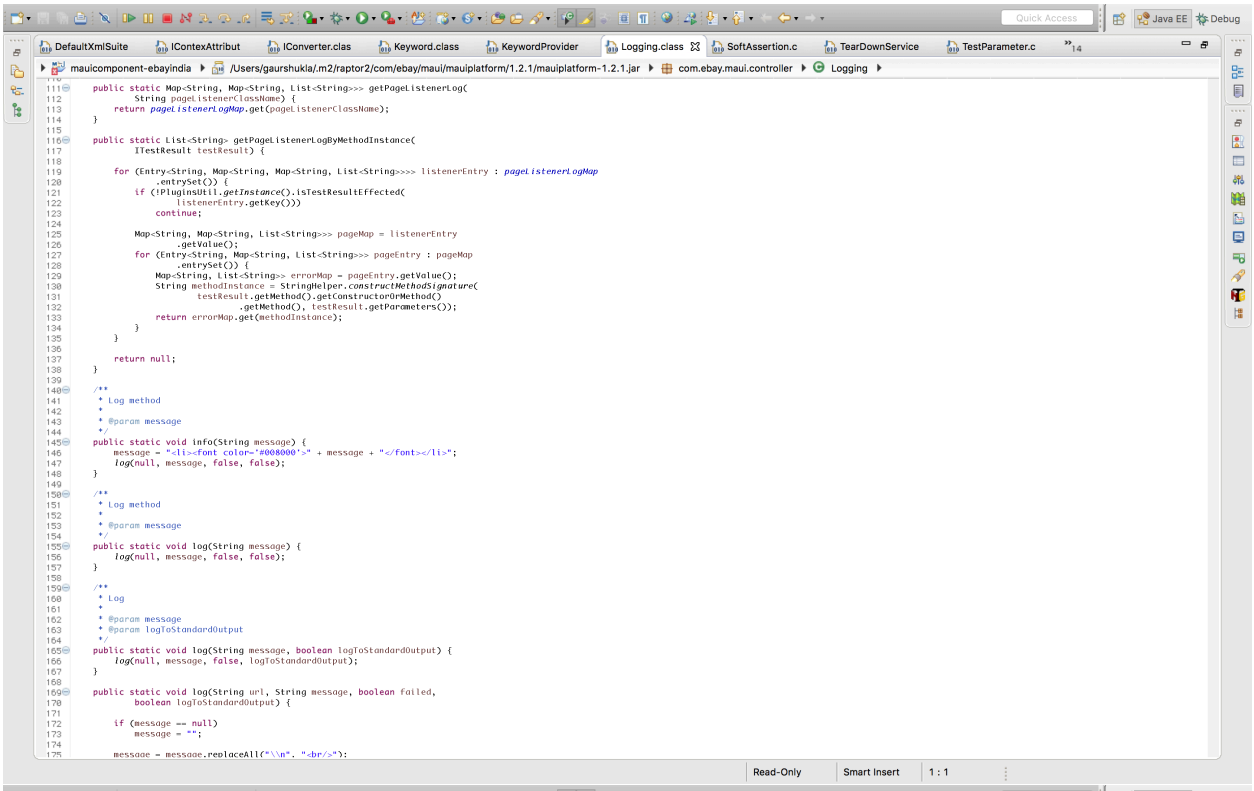

DefaultXmlSuite
IContextAttribut
IConverter.clas
Keyword.class
KeywordProvider
Logging.class
SoftAssertion.c
TearDownService
TestParameter.c
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.controller
Keyword
Read-Only
Smart Insert
1 : 1



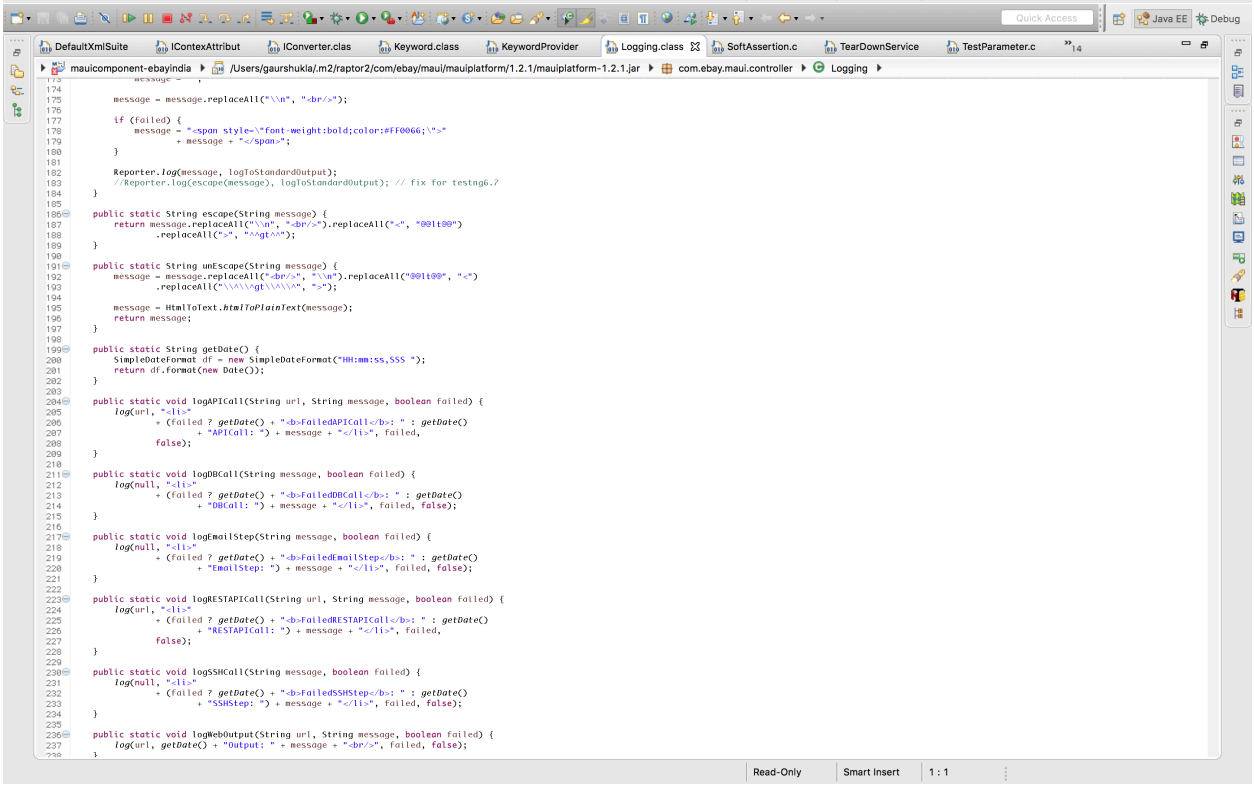
```
@SuppressWarnings("rawtypes")
public interface KeywordProvider {

    public final class PropertiesKeywordProvider implements KeywordProvider {
        private static Set<String> s_loadedKeywordsPackageSet = new HashSet<String>();
        private static Map<String, String> s_keywordsMap = new HashMap<String, String>();
        private static final String DEFAULT_KEYWORD_KEY_PREFIX = "default";
        private String m_resourceBaseName = null;

        public static final String DEFAULT_RESOURCE_BASE_NAME = "keywords";

        public PropertiesKeywordProvider() {
            this.m_resourceBaseName = DEFAULT_RESOURCE_BASE_NAME;
        }

        public PropertiesKeywordProvider(String resourceBaseName) {
            this.m_resourceBaseName = (null == resourceBaseName || "".equals(resourceBaseName)) ? DEFAULT_RESOURCE_BASE_NAME : resourceBaseName;
        }

        /**
         * Load keywords for a domain flow.
         *
         * @param flowClass
         * @throws Exception
         */
        private void _loadKeywords(String site, Class stepClass) {
            synchronized (PropertiesKeywordProvider.class) {
                // to load default properties
                String packagePath = stepClass.getPackage().getName();

                if (!s_loadedKeywordsPackageSet.contains(m_resourceBaseName + "." + DEFAULT_KEYWORD_KEY_PREFIX + "." + packagePath + ".keywords")) {
                    ResourceBundle defaultBundle = _pickResourceBundle(stepClass);
                    if (null == defaultBundle)
                        logger.warn("WARN: resource \"" + packagePath + "." + m_resourceBaseName + ".properties\" not found.");
                    else {
                        Enumeration<String> keys = defaultBundle.getKeys();
                        String identifier = null;
                        String value = null;
                        while (keys.hasMoreElements()) {
                            identifier = keys.nextElement();
                            value = defaultBundle.getString(identifier);

                            if (!identifier.startsWith(packagePath))
                                identifier = packagePath + "." + identifier;

                            s_keywordsMap.put(m_resourceBaseName + "." + DEFAULT_KEYWORD_KEY_PREFIX + "." + identifier, value);
                        }
                    }
                    s_loadedKeywordsPackageSet.add(m_resourceBaseName + "." + DEFAULT_KEYWORD_KEY_PREFIX + "." + packagePath + ".keywords");
                }

                // to load the particular site properties
                ResourceBundle localizationalBundle = _pickResourceBundle(stepClass, site);
                if (null == localizationalBundle)
                    logger.warn("WARN: resource \"" + packagePath + "." + m_resourceBaseName + "_" + site.toLowerCase() + ".properties\" not found.");
                else {
                    Enumeration<String> keys = localizationalBundle.getKeys();
                    String identifier = null;
                    String value = null;
                    while (keys.hasMoreElements()) {
                        identifier = keys.nextElement();
                        value = localizationalBundle.getString(identifier);
//                      if (site.equalsIgnoreCase("RU")||site.equalsIgnoreCase("HK")) {
                            try {
                                value = new String(value.getBytes("ISO8859-1"), "UTF-8");
                            } catch (UnsupportedEncodingException e) {
                            }
//                      }

                        if (!identifier.startsWith(packagePath))
                            identifier = packagePath + "." + identifier;

                        s_keywordsMap.put(m_resourceBaseName + "." + site + "." + identifier, value);
                    }
                }
                s_loadedKeywordsPackageSet.add(m_resourceBaseName + "." + site + "." + packagePath + ".keywords");
            }
        }

        private ResourceBundle _pickResourceBundle(Class clz) {
            String packagePath = clz.getPackage().getName();
            try {
                return ResourceBundle.getBundle(packagePath + "." + m_resourceBaseName, new Locale("", "", ""));
            } catch (MissingResourceException e) {
                return null;
            }
        }

        private ResourceBundle _pickResourceBundle(Class clz, String site) {
            String packagePath = clz.getPackage().getName();
            try {
                return ResourceBundle.getBundle(packagePath + "." + m_resourceBaseName + ((null == site || "".equals(site)) ? "" : "_" + site.toLowerCase()));
            } catch (MissingResourceException e) {
                return null;
            }
        }

        private String getLocalizedSite() {
            String site = null;
            if (ContextManager.getThreadContext().getGBHSite() != null) {
                site = ContextManager.getThreadContext().getGBHSite();
            } else {
                site = ContextManager.getThreadContext().getSite();
            }
            return site;
        }

        public String getKeyword(Class stepClass, String identifier) {
            return getKeyword(getLocalizedSite(), stepClass, identifier);
        }

        public String getKeyword(String site, Class stepClass, String identifier) {
            if (!s_loadedKeywordsPackageSet.contains(m_resourceBaseName + "." + site + "." + stepClass.getPackage().getName() + ".keywords")) {
                _loadKeywords(site, stepClass);
            }

            boolean localizedBuild = Boolean.parseBoolean((String) ContextManager.getGlobalContext().getAttribute(Context.LOCALIZED_BUILD));
            if (localizedBuild) {// false to skip the site related picking
                if (s_keywordsMap.containsKey(m_resourceBaseName + "." + site + "." + stepClass.getCanonicalName() + "." + identifier))
                    return s_keywordsMap.get(m_resourceBaseName + "." + site + "." + stepClass.getCanonicalName() + "." + identifier);
            }

            if (s_keywordsMap.containsKey(m_resourceBaseName + "." + DEFAULT_KEYWORD_KEY_PREFIX + "." + stepClass.getCanonicalName() + "." + identifier))
                return s_keywordsMap.get(m_resourceBaseName + "." + DEFAULT_KEYWORD_KEY_PREFIX + "." + stepClass.getCanonicalName() + "." + identifier);

            // throw new
            // RuntimeException("No mappings found in the resource(base-name: "
```


DefaultXmlSuite
IContextAttribut
IConverter.clas
Keyword.class
KeywordProvider
Logging.class
SoftAssertion.c
TearDownService
TestParameter.c
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.controller
KeywordProvider
Read-Only
Smart Insert
1 : 1

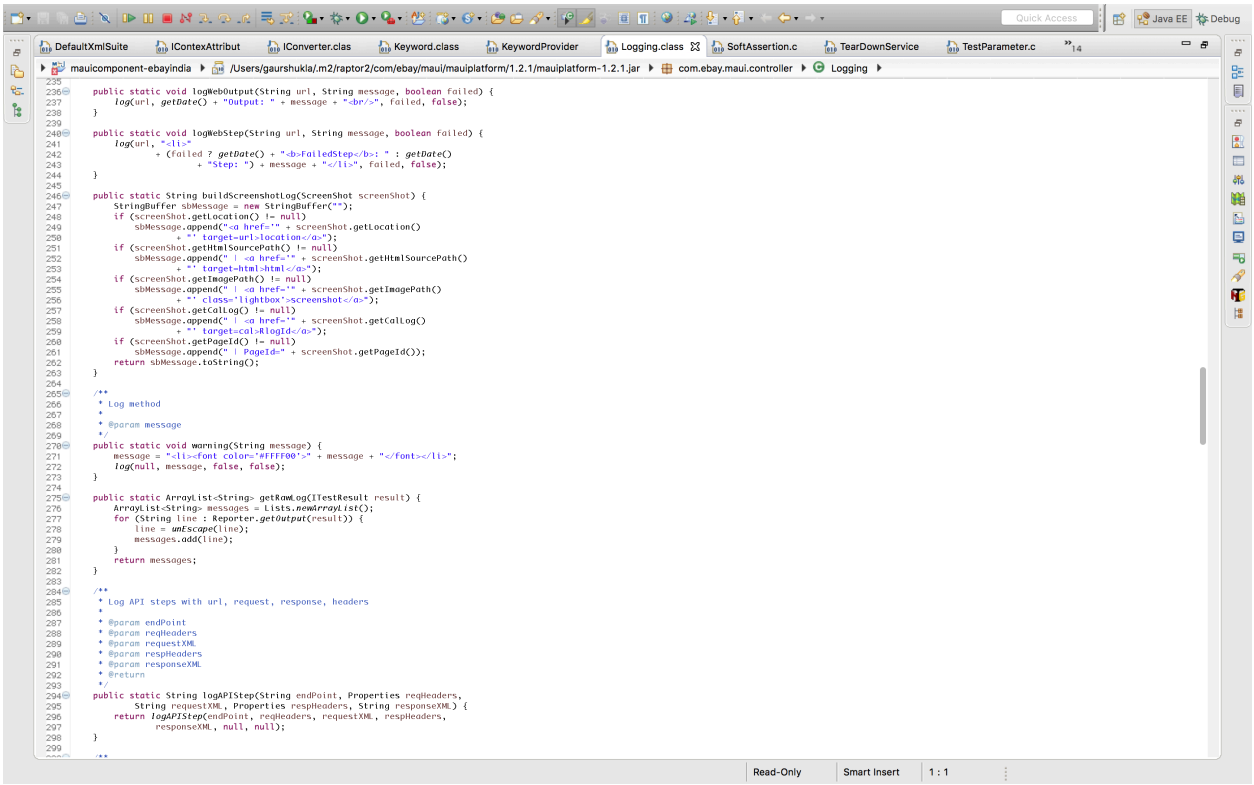

DefaultXmlSuite
IContextAttribut
IConverter.clas
Keyword.class
KeywordProvider
Logging.class
SoftAssertion.c
TearDownService
TestParameter.c
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.controller
Logging
Read-Only
Smart Insert
1 : 1



```
    public static Map<String, Map<String, List<String>>> getPageListenerLog(
            String pageListenerClassName) {
        return pageListenerLogMap.get(pageListenerClassName);
    }

    public static List<String> getPageListenerLogByMethodInstance(
            ITestResult testResult) {

        for (Entry<String, Map<String, Map<String, List<String>>>> listenerEntry : pageListenerLogMap
                .entrySet()) {
            if (!PluginsUtil.getInstance().isTestResultEffected(
                    listenerEntry.getKey()))
                continue;

            Map<String, Map<String, List<String>>> pageMap = listenerEntry
                    .getValue();
            for (Entry<String, Map<String, List<String>>> pageEntry : pageMap
                    .entrySet()) {
                Map<String, List<String>> errorMap = pageEntry.getValue();
                String methodInstance = StringHelper.constructMethodSignature(
                        testResult.getMethod().getConstructorOrMethod()
                                .getMethod(), testResult.getParameters());
                return errorMap.get(methodInstance);
            }
        }

        return null;
    }

    /**
     * Log method
     *
     * @param message
     */
    public static void info(String message) {
        message = "<li><font color='#008000'>" + message + "</font></li>";
        log(null, message, false, false);
    }

    /**
     * Log method
     *
     * @param message
     */
    public static void log(String message) {
        log(null, message, false, false);
    }

    /**
     * Log
     *
     * @param message
     * @param logToStandardOutput
     */
    public static void log(String message, boolean logToStandardOutput) {
        log(null, message, false, logToStandardOutput);
    }

    public static void log(String url, String message, boolean failed,
            boolean logToStandardOutput) {

        if (message == null)
            message = "";

        message = message.replaceAll("\\n", "<br/>");
```

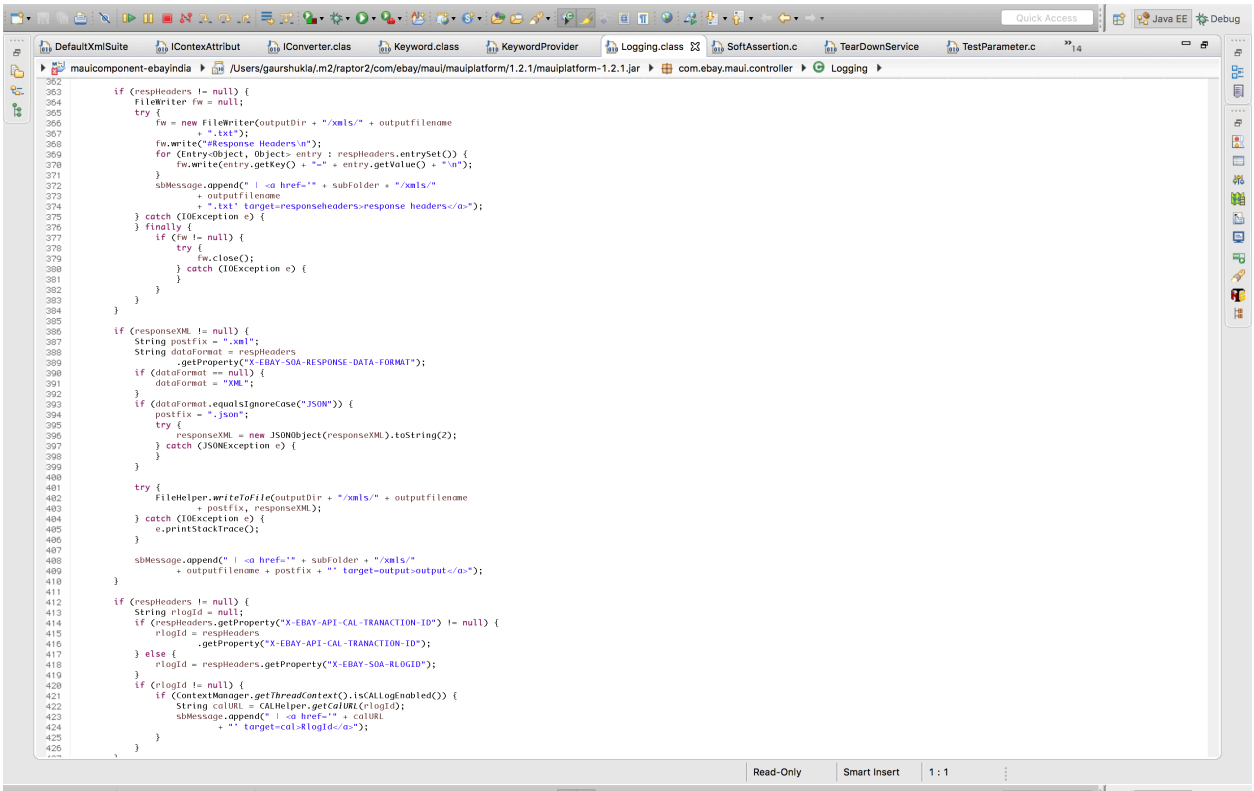


```
        message = message.replaceAll("\\n", "<br/>");

        if (failed) {
            message = "<span style=\"font-weight:bold;color:#FF0066;\">"
                    + message + "</span>";
        }

        Reporter.log(message, logToStandardOutput);
        //Reporter.log(escape(message), logToStandardOutput); // fix for testng6.7
    }

    public static String escape(String message) {
        return message.replaceAll("\\n", "<br/>").replaceAll("<", "@@lt@@")
                .replaceAll(">", "^^gt^^");
    }

    public static String unEscape(String message) {
        message = message.replaceAll("<br/>", "\\n").replaceAll("@@lt@@", "<")
                .replaceAll("\\^\\^gt\\^\\^", ">");

        message = HtmlToText.htmlToPlainText(message);
        return message;
    }

    public static String getDate() {
        SimpleDateFormat df = new SimpleDateFormat("HH:mm:ss,SSS ");
        return df.format(new Date());
    }

    public static void logAPICall(String url, String message, boolean failed) {
        log(url, "<li>"
                + (failed ? getDate() + "<b>FailedAPICall</b>: " : getDate()
                        + "APICall: ") + message + "</li>", failed,
                false);
    }

    public static void logDBCall(String message, boolean failed) {
        log(null, "<li>"
                + (failed ? getDate() + "<b>FailedDBCall</b>: " : getDate()
                        + "DBCall: ") + message + "</li>", failed, false);
    }

    public static void logEmailStep(String message, boolean failed) {
        log(null, "<li>"
                + (failed ? getDate() + "<b>FailedEmailStep</b>: " : getDate()
                        + "EmailStep: ") + message + "</li>", failed, false);
    }

    public static void logRESTAPICall(String url, String message, boolean failed) {
        log(url, "<li>"
                + (failed ? getDate() + "<b>FailedRESTAPICall</b>: " : getDate()
                        + "RESTAPICall: ") + message + "</li>", failed,
                false);
    }

    public static void logSSHCall(String message, boolean failed) {
        log(null, "<li>"
                + (failed ? getDate() + "<b>FailedSSHStep</b>: " : getDate()
                        + "SSHStep: ") + message + "</li>", failed, false);
    }

    public static void logWebOutput(String url, String message, boolean failed) {
        log(url, getDate() + "Output: " + message + "<br/>", failed, false);
    }
```

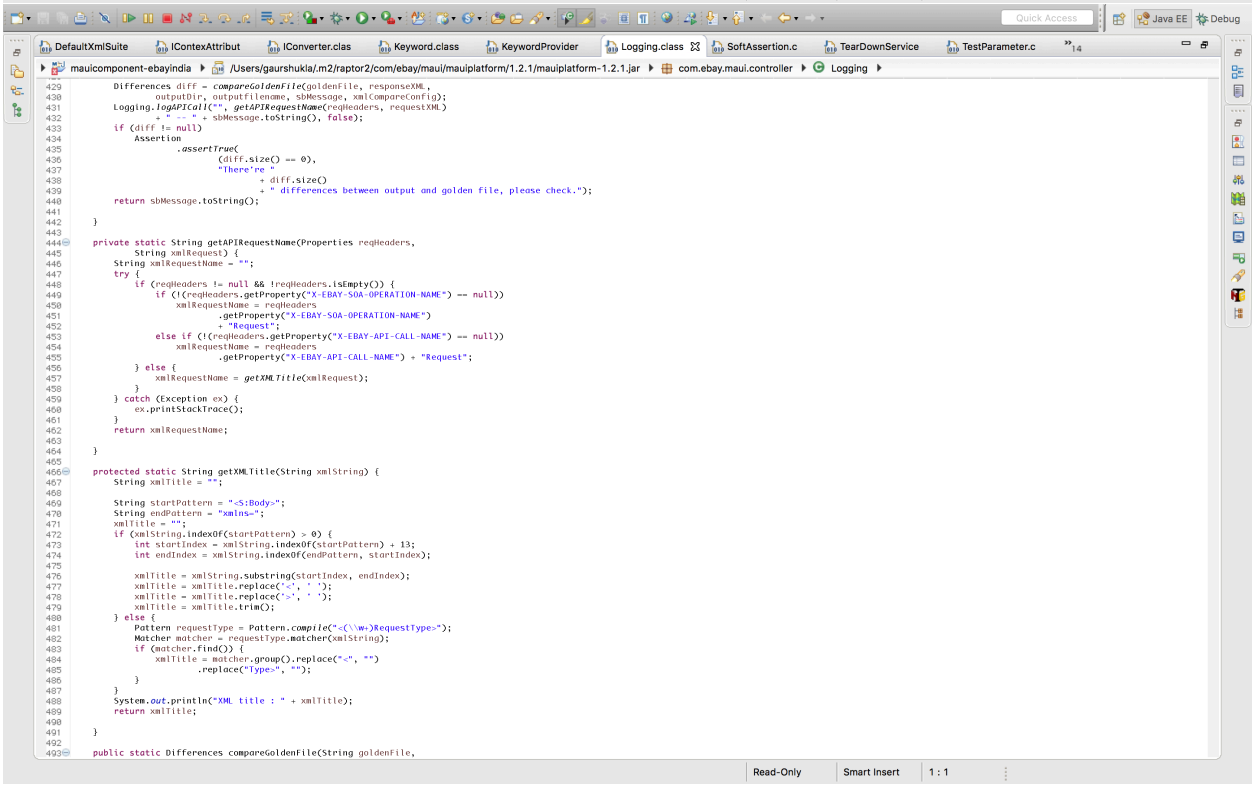


```
    public static void logWebOutput(String url, String message, boolean failed) {
        log(url, getDate() + "Output: " + message + "<br/>", failed, false);
    }

    public static void logWebStep(String url, String message, boolean failed) {
        log(url, "<li>"
                + (failed ? getDate() + "<b>FailedStep</b>: " : getDate()
                        + "Step: ") + message + "</li>", failed, false);
    }

    public static String buildScreenshotLog(ScreenShot screenShot) {
        StringBuffer sbMessage = new StringBuffer("");
        if (screenShot.getLocation() != null)
            sbMessage.append("<a href='" + screenShot.getLocation()
                    + "' target=url>location</a>");
        if (screenShot.getHtmlSourcePath() != null)
            sbMessage.append(" | <a href='" + screenShot.getHtmlSourcePath()
                    + "' target=html>html</a>");
        if (screenShot.getImagePath() != null)
            sbMessage.append(" | <a href='" + screenShot.getImagePath()
                    + "' class='lightbox'>screenshot</a>");
        if (screenShot.getCalLog() != null)
            sbMessage.append(" | <a href='" + screenShot.getCalLog()
                    + "' target=cal>RlogId</a>");
        if (screenShot.getPageId() != null)
            sbMessage.append(" | PageId=" + screenShot.getPageId());
        return sbMessage.toString();
    }

    /**
     * Log method
     *
     * @param message
     */
    public static void warning(String message) {
        message = "<li><font color='#FFFF00'>" + message + "</font></li>";
        log(null, message, false, false);
    }

    public static ArrayList<String> getRawLog(ITestResult result) {
        ArrayList<String> messages = Lists.newArrayList();
        for (String line : Reporter.getOutput(result)) {
            line = unEscape(line);
            messages.add(line);
        }
        return messages;
    }

    /**
     * Log API steps with url, request, response, headers
     *
     * @param endPoint
     * @param reqHeaders
     * @param requestXML
     * @param respHeaders
     * @param responseXML
     * @return
     */
    public static String logAPIStep(String endPoint, Properties reqHeaders,
            String requestXML, Properties respHeaders, String responseXML) {
        return logAPIStep(endPoint, reqHeaders, requestXML, respHeaders,
                responseXML, null, null);
    }

    /**
```

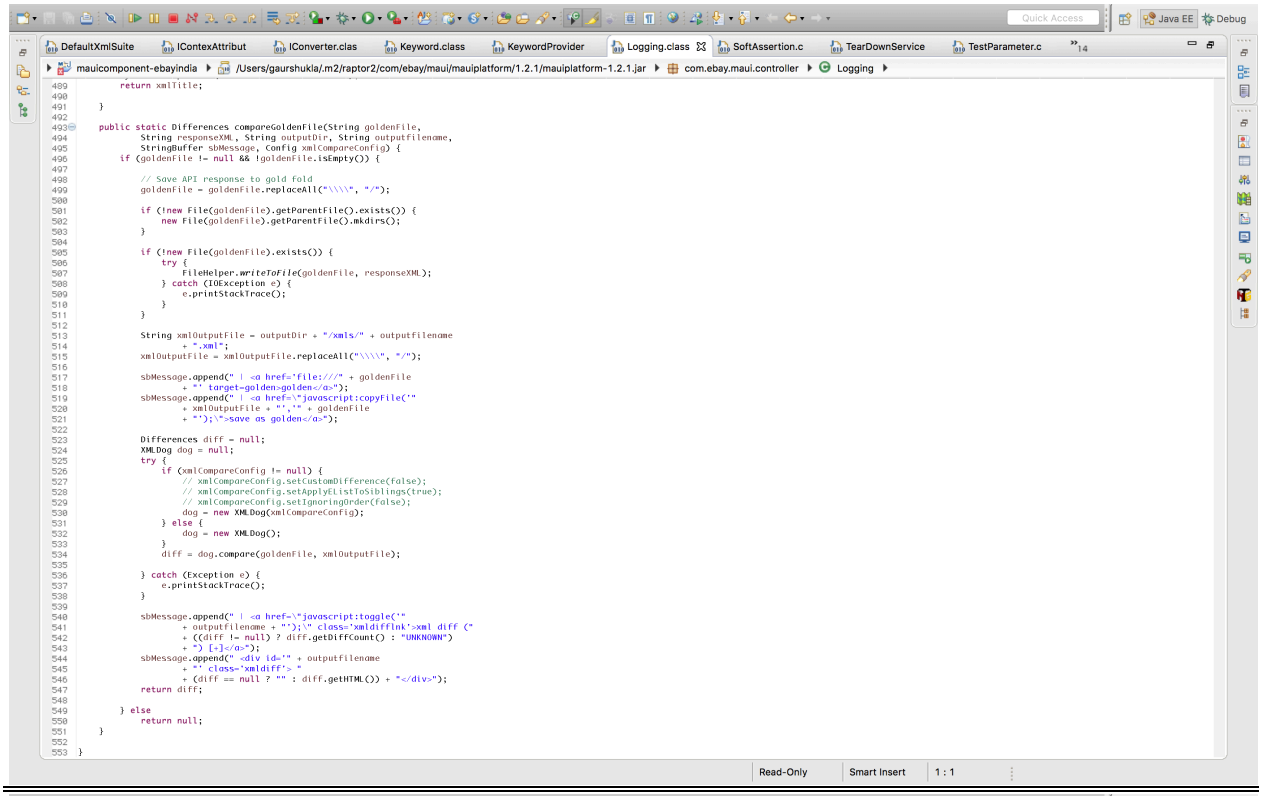
```
    /**
     * Log API steps with url, request, response, headers,golden file
     *
     * @param endPoint
     * @param reqHeaders
     * @param requestXML
     * @param respHeaders
     * @param responseXML
     * @param goldenFile
     * @param xmlCompareConfig
     * @return
     */
    public static String logAPIStep(String endPoint, Properties reqHeaders,
            String requestXML, Properties respHeaders, String responseXML,
            String goldenFile, Config xmlCompareConfig) {
        if (ContextManager.getGlobalContext() == null
                || ContextManager.getGlobalContext().getTestNGContext() == null)
            return "";
        StringBuffer sbMessage = new StringBuffer("");

        sbMessage.append("<a href='" + endPoint + "' target=apiurl>url</a>");

        String inputfilename = URLHelper.getRandomHashCode("soa-input");
        String outputfilename = URLHelper.getRandomHashCode("soa-output");
        String subFolder = ContextManager.getGlobalContext().getTestNGContext()
                .getSuite().getName();
        String outputDir = ContextManager.getGlobalContext()
                .getOutputDirectory();

        if (reqHeaders != null) {
            FileWriter fw = null;
            try {
                fw = new FileWriter(outputDir + "/xmls/" + inputfilename
                        + ".txt");
                fw.write("#Request Headers\n");
                for (Entry<Object, Object> entry : reqHeaders.entrySet()) {
                    fw.write(entry.getKey() + "=" + entry.getValue() + "\n");
                }
                sbMessage.append(" | <a href='" + subFolder + "/xmls/"
                        + inputfilename
                        + ".txt' target=requestheaders>request headers</a>");
            } catch (IOException e) {
            } finally {
                if (fw != null) {
                    try {
                        fw.close();
                    } catch (IOException e) {
                    }
                }
            }
        }

        if (requestXML != null) {
            try {
                FileHelper.writeToFile(outputDir + "/xmls/" + inputfilename
                        + ".xml", requestXML);
            } catch (IOException e) {
                e.printStackTrace();
            }
            sbMessage.append(" | <a href='" + subFolder + "/xmls/"
                    + inputfilename + ".xml' target=input>input</a>");
        }

        if (respHeaders != null) {
```



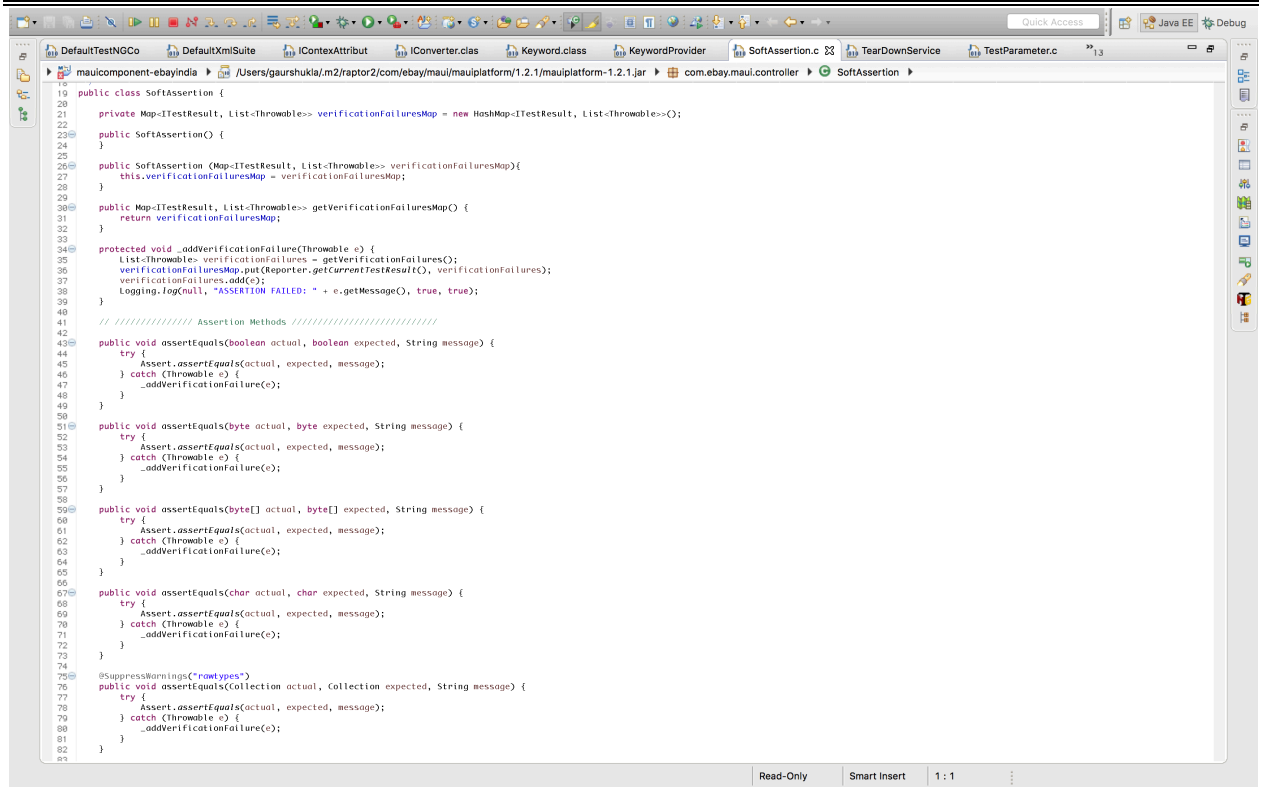
```
        if (respHeaders != null) {
            FileWriter fw = null;
            try {
                fw = new FileWriter(outputDir + "/xmls/" + outputfilename
                        + ".txt");
                fw.write("#Response Headers\n");
                for (Entry<Object, Object> entry : respHeaders.entrySet()) {
                    fw.write(entry.getKey() + "=" + entry.getValue() + "\n");
                }
                sbMessage.append(" | <a href='" + subFolder + "/xmls/"
                        + outputfilename
                        + ".txt' target=responseheaders>response headers</a>");
            } catch (IOException e) {
            } finally {
                if (fw != null) {
                    try {
                        fw.close();
                    } catch (IOException e) {
                    }
                }
            }

            if (responseXML != null) {
                String postfix = ".xml";
                String dataFormat = respHeaders
                        .getProperty("X-EBAY-SOA-RESPONSE-DATA-FORMAT");
                if (dataFormat == null) {
                    dataFormat = "XML";
                }
                if (dataFormat.equalsIgnoreCase("JSON")) {
                    postfix = ".json";
                    try {
                        responseXML = new JSONObject(responseXML).toString(2);
                    } catch (JSONException e) {
                    }
                }

                try {
                    FileHelper.writeToFile(outputDir + "/xmls/" + outputfilename
                            + postfix, responseXML);
                } catch (IOException e) {
                    e.printStackTrace();
                }

                sbMessage.append(" | <a href='" + subFolder + "/xmls/"
                        + outputfilename + postfix + "' target=output>output</a>");
            }

            if (respHeaders != null) {
                String rlogId = null;
                if (respHeaders.getProperty("X-EBAY-API-CAL-TRANACTION-ID") != null) {
                    rlogId = respHeaders
                            .getProperty("X-EBAY-API-CAL-TRANACTION-ID");
                } else {
                    rlogId = respHeaders.getProperty("X-EBAY-SOA-RLOGID");
                }
                if (rlogId != null) {
                    if (ContextManager.getThreadContext().isCALLogEnabled()) {
                        String calURL = CALHelper.getCalURL(rlogId);
                        sbMessage.append(" | <a href='" + calURL
                                + "' target=cal>RlogId</a>");
                    }
                }
```



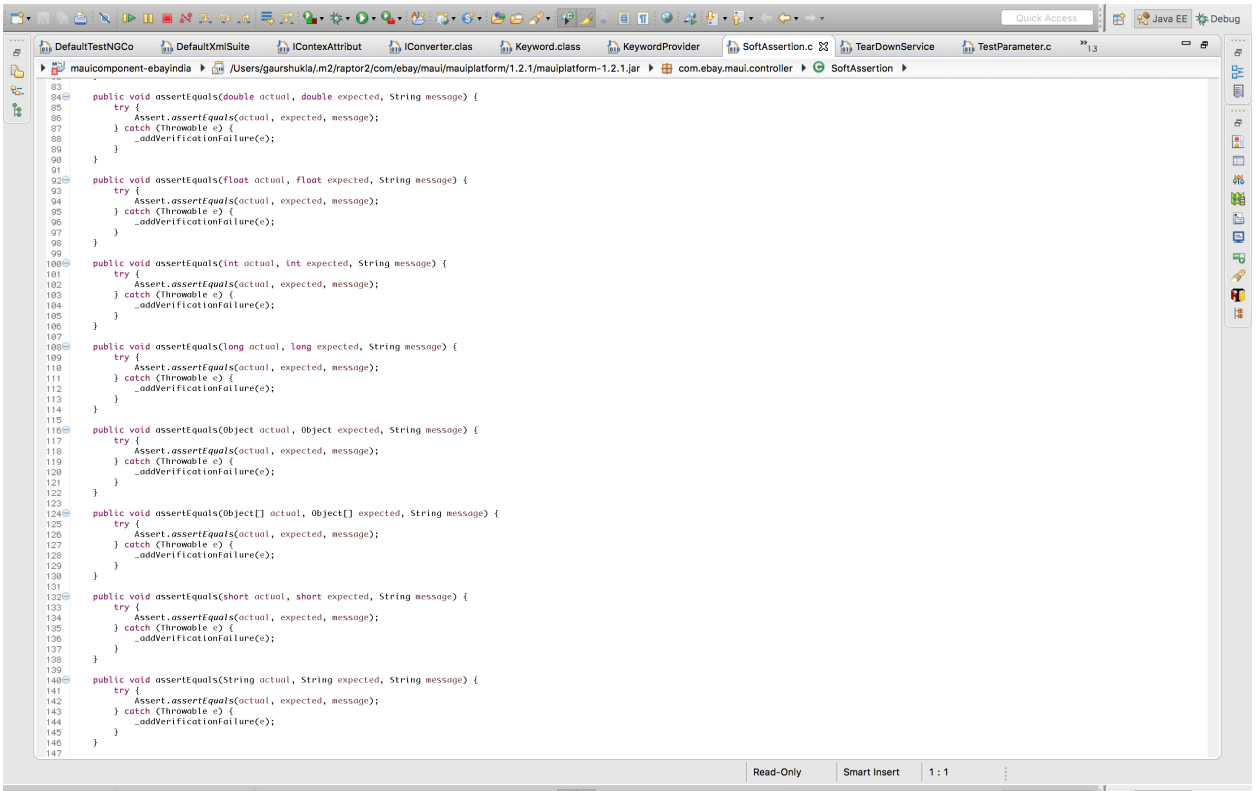
```
        Differences diff = compareGoldenFile(goldenFile, responseXML,
                outputDir, outputfilename, sbMessage, xmlCompareConfig);
        Logging.logAPICall("", getAPIRequestName(reqHeaders, requestXML)
                + " --- " + sbMessage.toString(), false);
        if (diff != null)
            Assertion
                    .assertTrue(
                            (diff.size() == 0),
                            "There're "
                                    + diff.size()
                                    + " differences between output and golden file, please check.");
        return sbMessage.toString();

    }

    private static String getAPIRequestName(Properties reqHeaders,
            String xmlRequest) {
        String xmlRequestName = "";
        try {
            if (reqHeaders != null && !reqHeaders.isEmpty()) {
                if (!(reqHeaders.getProperty("X-EBAY-SOA-OPERATION-NAME") == null))
                    xmlRequestName = reqHeaders
                            .getProperty("X-EBAY-SOA-OPERATION-NAME")
                            + "Request";
                else if (!(reqHeaders.getProperty("X-EBAY-API-CALL-NAME") == null))
                    xmlRequestName = reqHeaders
                            .getProperty("X-EBAY-API-CALL-NAME") + "Request";
            } else {
                xmlRequestName = getXMLTitle(xmlRequest);
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return xmlRequestName;

    }

    protected static String getXMLTitle(String xmlString) {
        String xmlTitle = "";

        String startPattern = "<S:Body>";
        String endPattern = "xmlns=";
        xmlTitle = "";
        if (xmlString.indexOf(startPattern) > 0) {
            int startIndex = xmlString.indexOf(startPattern) + 13;
            int endIndex = xmlString.indexOf(endPattern, startIndex);

            xmlTitle = xmlString.substring(startIndex, endIndex);
            xmlTitle = xmlTitle.replace('<', ' ');
            xmlTitle = xmlTitle.replace('>', ' ');
            xmlTitle = xmlTitle.trim();
        } else {
            Pattern requestType = Pattern.compile("<(\\w+)RequestType>");
            Matcher matcher = requestType.matcher(xmlString);
            if (matcher.find()) {
                xmlTitle = matcher.group().replace("<", "")
                        .replace("Type>", "");
            }
        }
        System.out.println("XML title : " + xmlTitle);
        return xmlTitle;

    }

    public static Differences compareGoldenFile(String goldenFile,
```



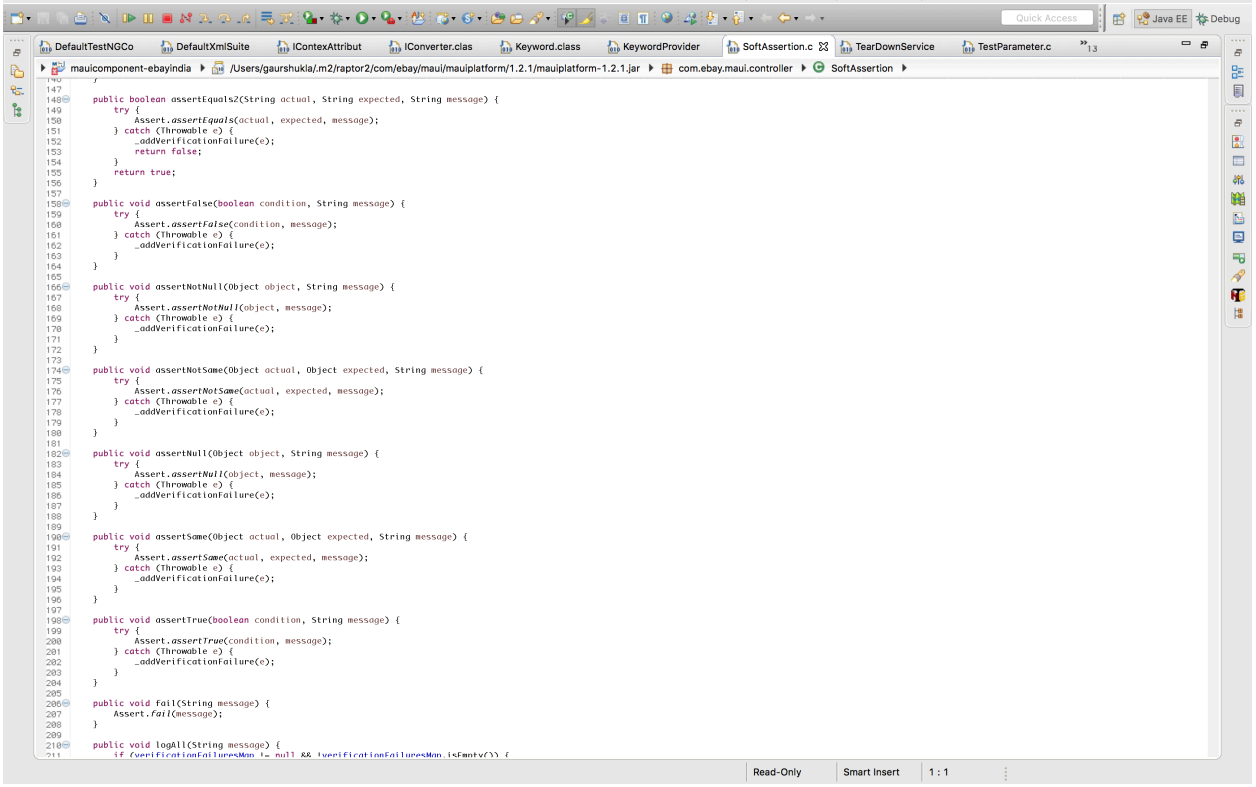
```
        return xmlTitle;

    }

    public static Differences compareGoldenFile(String goldenFile,
            String responseXML, String outputDir, String outputFilename,
            StringBuffer sbMessage, Config xmlCompareConfig) {
        if (goldenFile != null && !goldenFile.isEmpty()) {

            // Save API response to gold fold
            goldenFile = goldenFile.replaceAll("\\\\", "/");

            if (!new File(goldenFile).getParentFile().exists()) {
                new File(goldenFile).getParentFile().mkdirs();
            }

            if (!new File(goldenFile).exists()) {
                try {
                    FileHelper.writeToFile(goldenFile, responseXML);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            String xmlOutputFile = outputDir + "/xmls/" + outputfilename
                    + ".xml";
            xmlOutputFile = xmlOutputFile.replaceAll("\\\\", "/");

            sbMessage.append(" | <a href='file:///" + goldenFile
                    + "' target=golden>golden</a>");
            sbMessage.append(" | <a href=\"javascript:copyFile('"
                    + xmlOutputFile + "','" + goldenFile
                    + "');\">save as golden</a>");

            Differences diff = null;
            XMLDog dog = null;
            try {
                if (xmlCompareConfig != null) {
                    // xmlCompareConfig.setCustomDifference(false);
                    // xmlCompareConfig.setApplyElistToSiblings(true);
                    // xmlCompareConfig.setIgnoringOrder(false);
                    dog = new XMLDog(xmlCompareConfig);
                } else {
                    dog = new XMLDog();
                }
                diff = dog.compare(goldenFile, xmlOutputFile);

            } catch (Exception e) {
                e.printStackTrace();
            }

            sbMessage.append(" | <a href=\"javascript:toggle('"
                    + outputfilename + "');\" class='xmldifflnk'>xml diff ("
                    + ((diff != null) ? diff.getDiffCount() : "UNKNOWN")
                    + ") [+]</a>");
            sbMessage.append(" <div id='" + outputfilename
                    + "' class='xmldiff'> "
                    + (diff == null ? "" : diff.getHTML()) + "</div>");
            return diff;

        } else
            return null;
    }
}
```



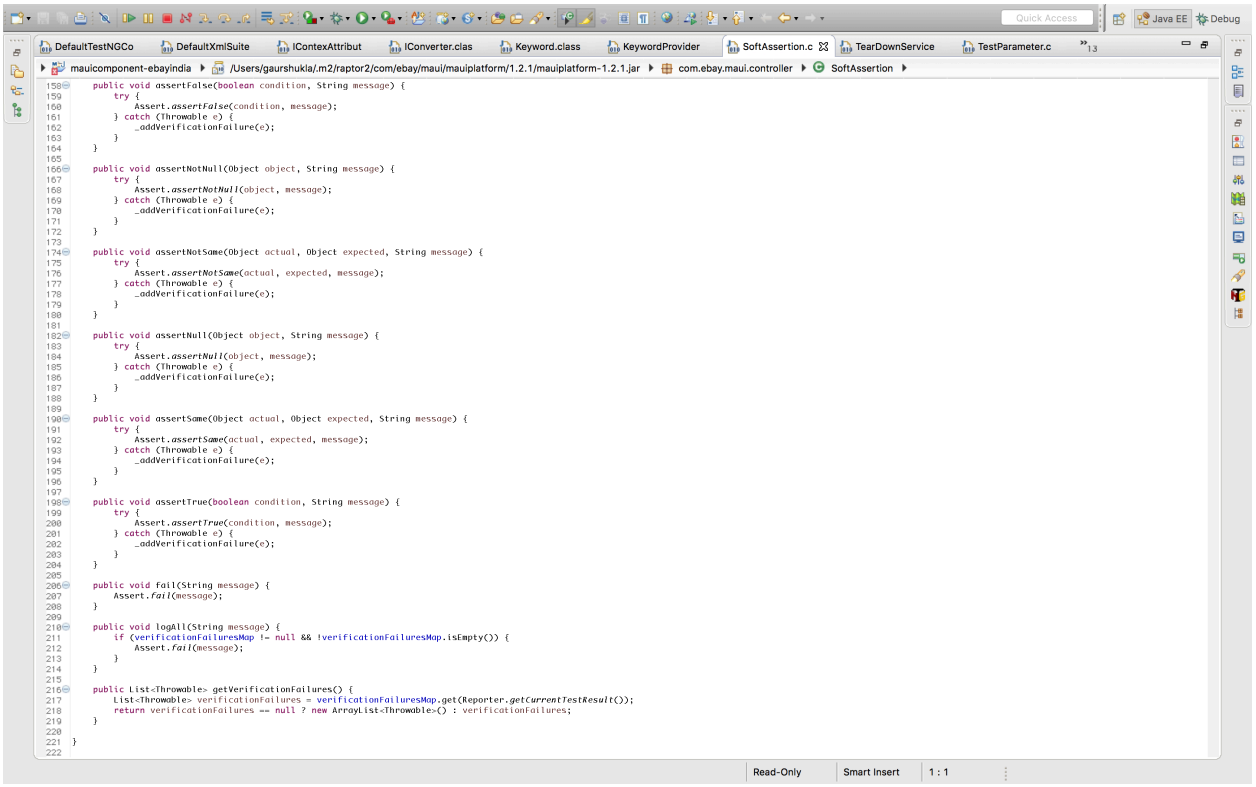
```
public class SoftAssertion {

    private Map<ITestResult, List<Throwable>> verificationFailuresMap = new HashMap<ITestResult, List<Throwable>>();

    public SoftAssertion() {
    }

    public SoftAssertion (Map<ITestResult, List<Throwable>> verificationFailuresMap){
        this.verificationFailuresMap = verificationFailuresMap;
    }

    public Map<ITestResult, List<Throwable>> getVerificationFailuresMap() {
        return verificationFailuresMap;
    }

    protected void _addVerificationFailure(Throwable e) {
        List<Throwable> verificationFailures = getVerificationFailures();
        verificationFailuresMap.put(Reporter.getCurrentTestResult(), verificationFailures);
        verificationFailures.add(e);
        Logging.log(null, "ASSERTION FAILED: " + e.getMessage(), true, true);
    }

    // ////////////// Assertion Methods ////////////////////////////

    public void assertEquals(boolean actual, boolean expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }

    public void assertEquals(byte actual, byte expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }

    public void assertEquals(byte[] actual, byte[] expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }

    public void assertEquals(char actual, char expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }

    @SuppressWarnings("rawtypes")
    public void assertEquals(Collection actual, Collection expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }
```



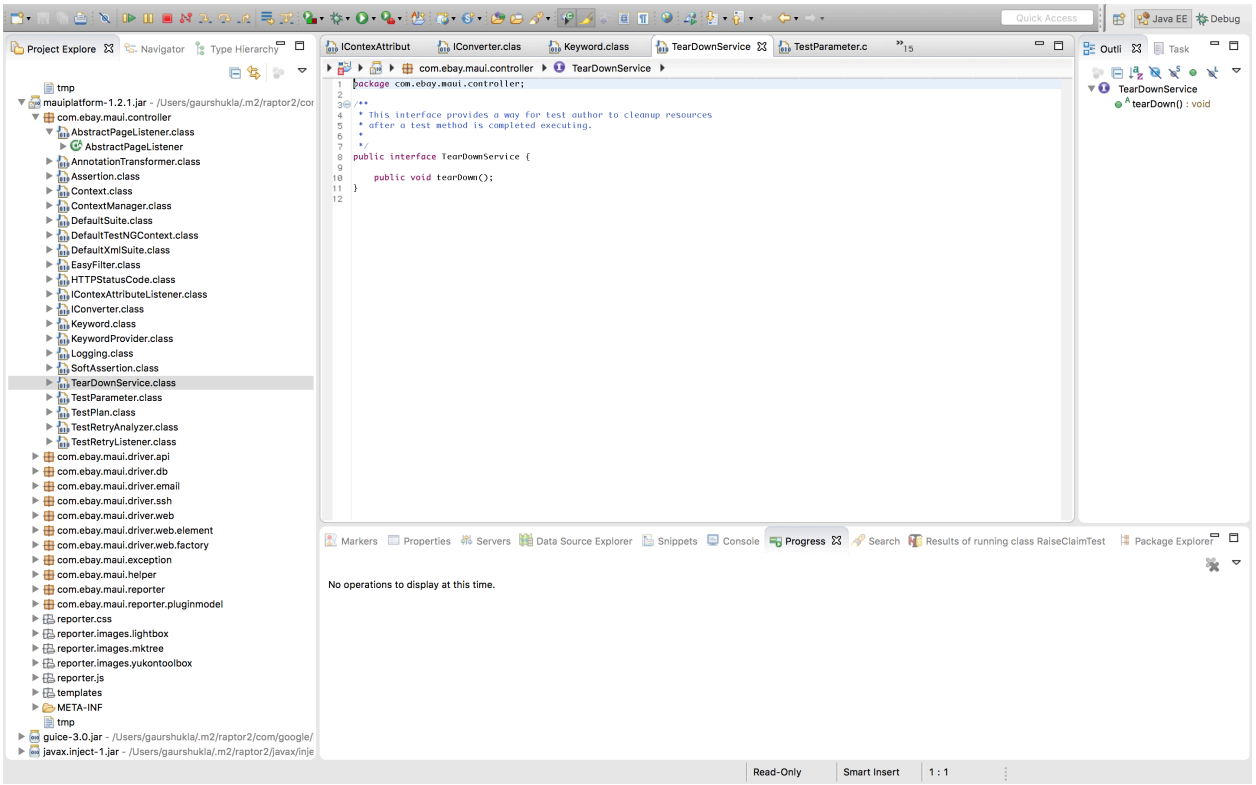
```
    public void assertEquals(double actual, double expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
        }
    }

    public void assertEquals(float actual, float expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
        }
    }

    public void assertEquals(int actual, int expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
        }
    }

    public void assertEquals(long actual, long expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
        }
    }

    public void assertEquals(Object actual, Object expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
        }
    }

    public void assertEquals(Object[] actual, Object[] expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
        }
    }

    public void assertEquals(short actual, short expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
        }
    }

    public void assertEquals(String actual, String expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
        }
    }
```



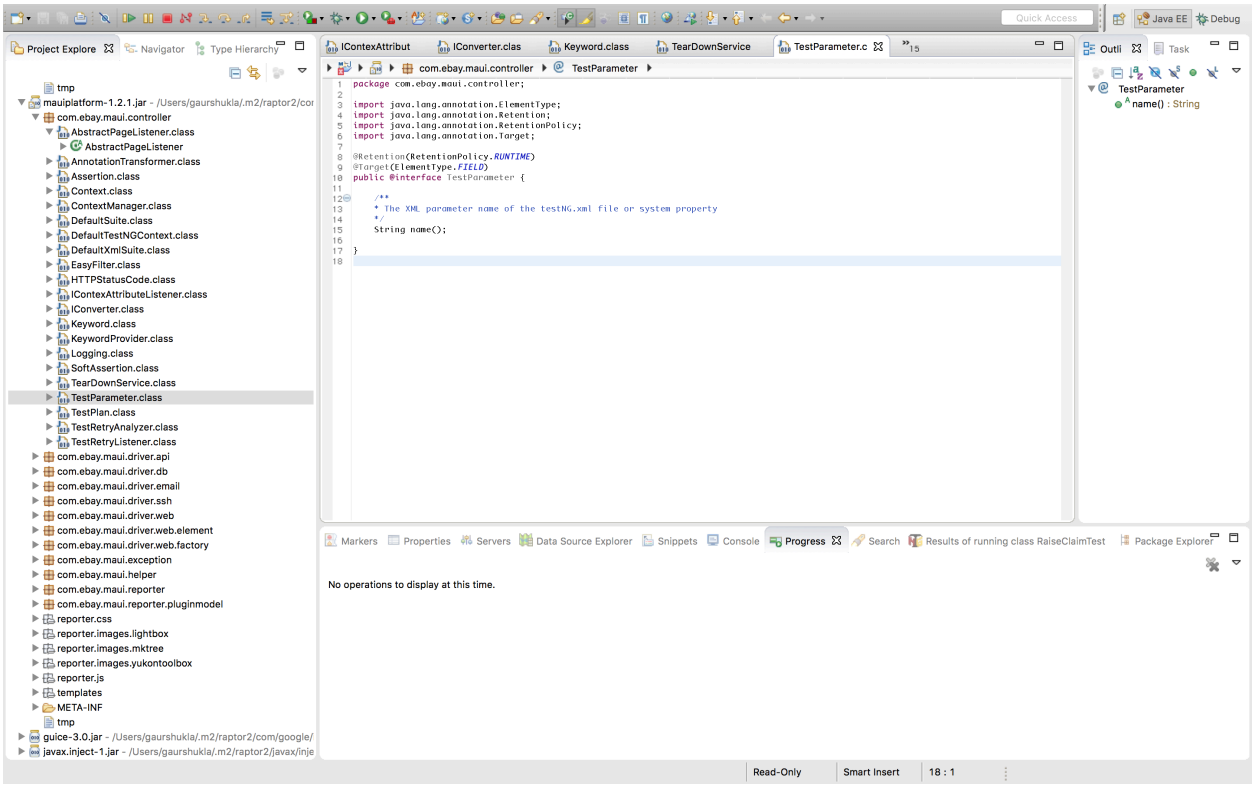
```
    public boolean assertEquals2(String actual, String expected, String message) {
        try {
            Assert.assertEquals(actual, expected, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
            return false;
        }
        return true;
    }

    public void assertFalse(boolean condition, String message) {
        try {
            Assert.assertFalse(condition, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
        }
    }

    public void assertNotNull(Object object, String message) {
        try {
            Assert.assertNotNull(object, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
        }
    }

    public void assertNotSame(Object actual, Object expected, String message) {
        try {
            Assert.assertNotSame(actual, expected, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
        }
    }

    public void assertNull(Object object, String message) {
        try {
            Assert.assertNull(object, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
        }
    }

    public void assertSame(Object actual, Object expected, String message) {
        try {
            Assert.assertSame(actual, expected, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
        }
    }

    public void assertTrue(boolean condition, String message) {
        try {
            Assert.assertTrue(condition, message);
        } catch (Throwable e) {
            addVerificationFailure(e);
        }
    }

    public void fail(String message) {
        Assert.fail(message);
    }

    public void logAll(String message) {
        if (verificationFailuresMap != null && !verificationFailuresMap.isEmpty()) {
```

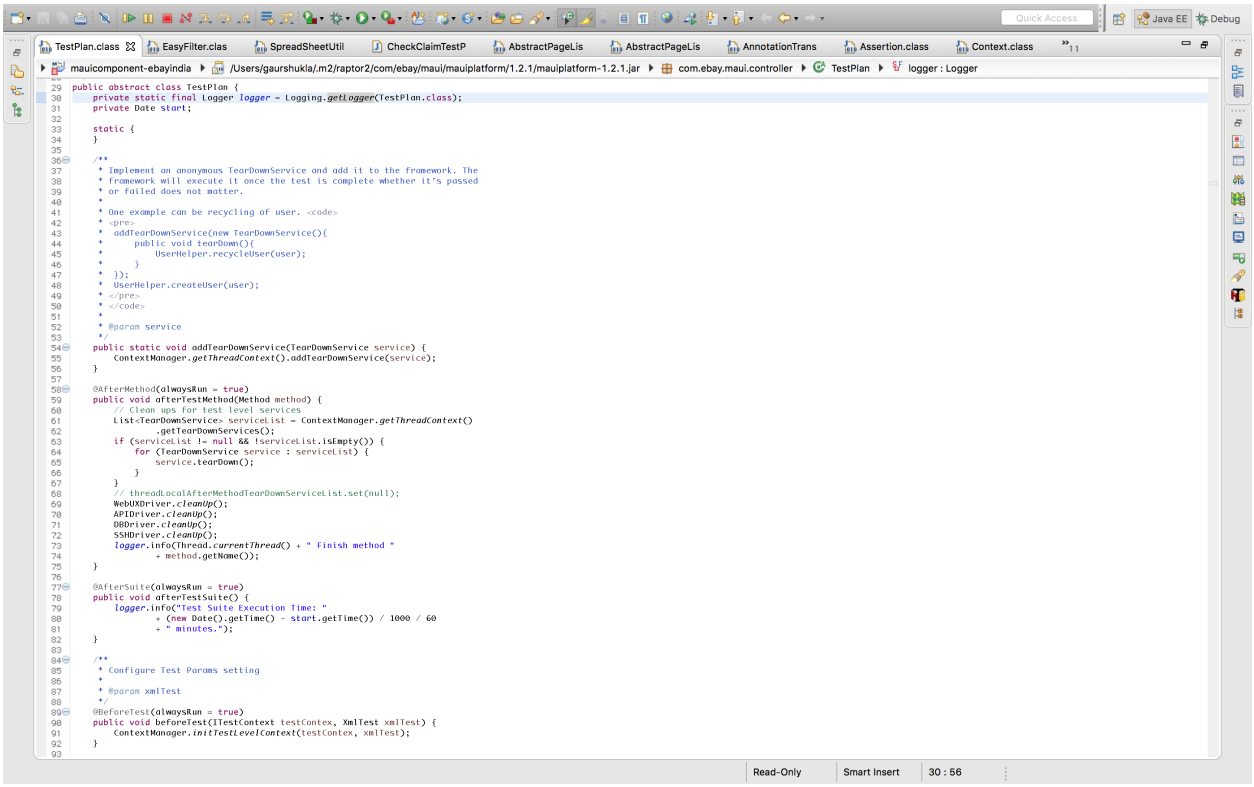


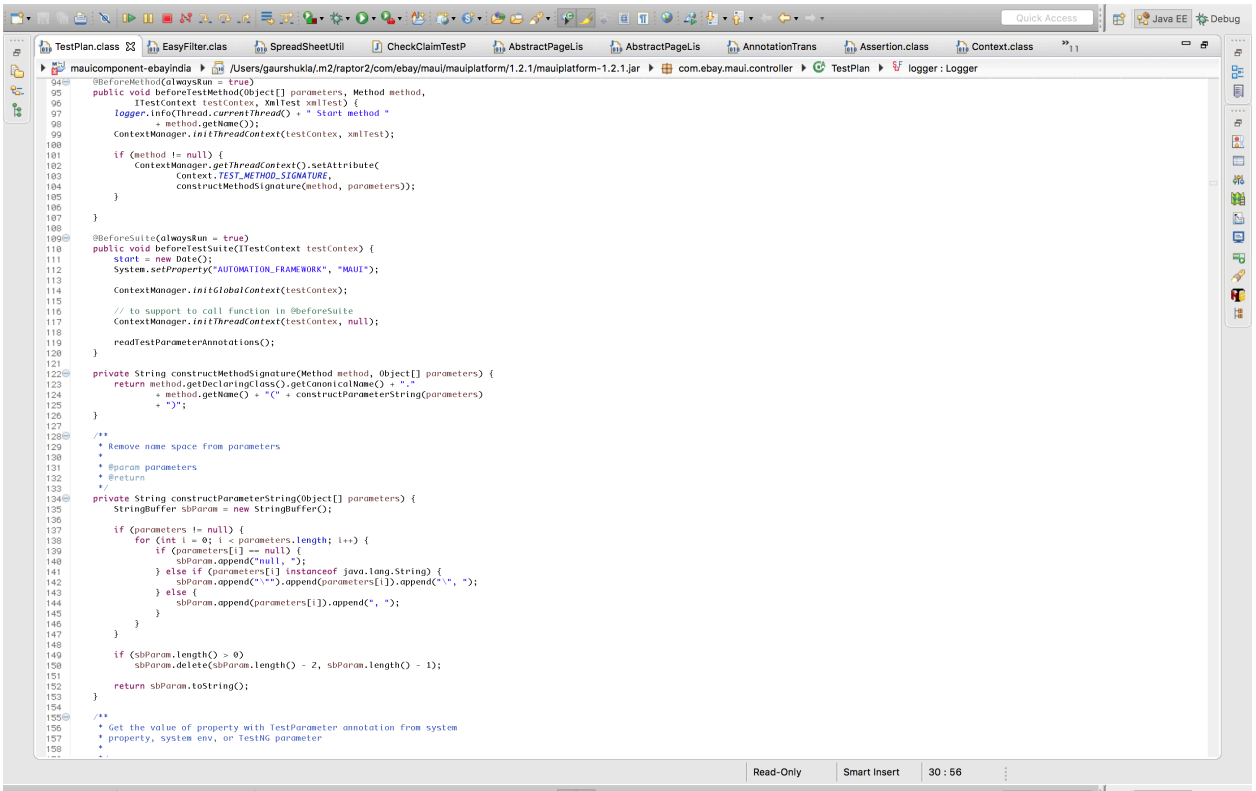
```
    public void assertFalse(boolean condition, String message) {
        try {
            Assert.assertFalse(condition, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }

    public void assertNotNull(Object object, String message) {
        try {
            Assert.assertNotNull(object, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }

    public void assertNotSame(Object actual, Object expected, String message) {
        try {
            Assert.assertNotSame(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }

    public void assertNull(Object object, String message) {
        try {
            Assert.assertNull(object, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }

    public void assertSame(Object actual, Object expected, String message) {
        try {
            Assert.assertSame(actual, expected, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }

    public void assertTrue(boolean condition, String message) {
        try {
            Assert.assertTrue(condition, message);
        } catch (Throwable e) {
            _addVerificationFailure(e);
        }
    }

    public void fail(String message) {
        Assert.fail(message);
    }

    public void logAll(String message) {
        if (verificationFailuresMap != null && !verificationFailuresMap.isEmpty()) {
            Assert.fail(message);
        }
    }

    public List<Throwable> getVerificationFailures() {
        List<Throwable> verificationFailures = verificationFailuresMap.get(Reporter.getCurrentTestResult());
        return verificationFailures == null ? new ArrayList<Throwable>() : verificationFailures;
    }
}
```



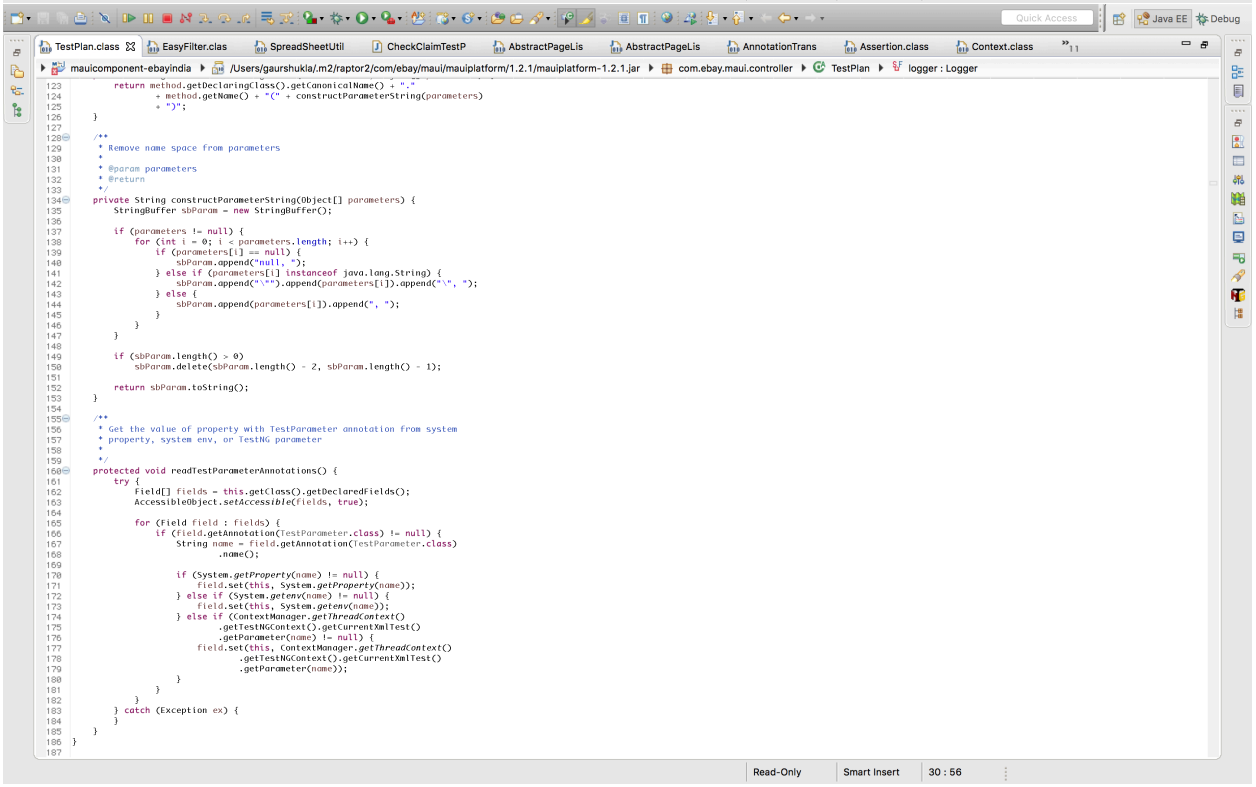
```
package com.ebay.maui.controller;

/**
 * This interface provides a way for test author to cleanup resources
 * after a test method is completed executing.
 *
 */
public interface TearDownService {

    public void tearDown();
}
```


Project Explore
Navigator
Type Hierarchy
IContextAttribut
IConverter.clas
Keyword.class
TearDownService
TestParameter.c
Outli
Task
com.ebay.maui.controller
TestParameter
package com.ebay.maui.controller;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.FIELD)
public @interface TestParameter {
/**
* The XML parameter name of the testNG.xml file or system property
*/
String name();
}
TestParameter
name() : 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.
tmp
mauiplatform-1.2.1.jar - /Users/gaurshukla/.m2/raptor2/cor
com.ebay.maui.controller
AbstractPageListener.class
AbstractPageListener
AnnotationTransformer.class
Assertion.class
Context.class
ContextManager.class
DefaultSuite.class
DefaultTestNGContext.class
DefaultXmlSuite.class
EasyFilter.class
HTTPStatusCode.class
IContextAttributeListener.class
IConverter.class
Keyword.class
KeywordProvider.class
Logging.class
SoftAssertion.class
TearDownService.class
TestParameter.class
TestPlan.class
TestRetryAnalyzer.class
TestRetryListener.class
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
com.ebay.maui.driver.web.factory
com.ebay.maui.exception
com.ebay.maui.helper
com.ebay.maui.reporter
com.ebay.maui.reporter.pluginmodel
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/inje
Read-Only
Smart Insert
18 : 1

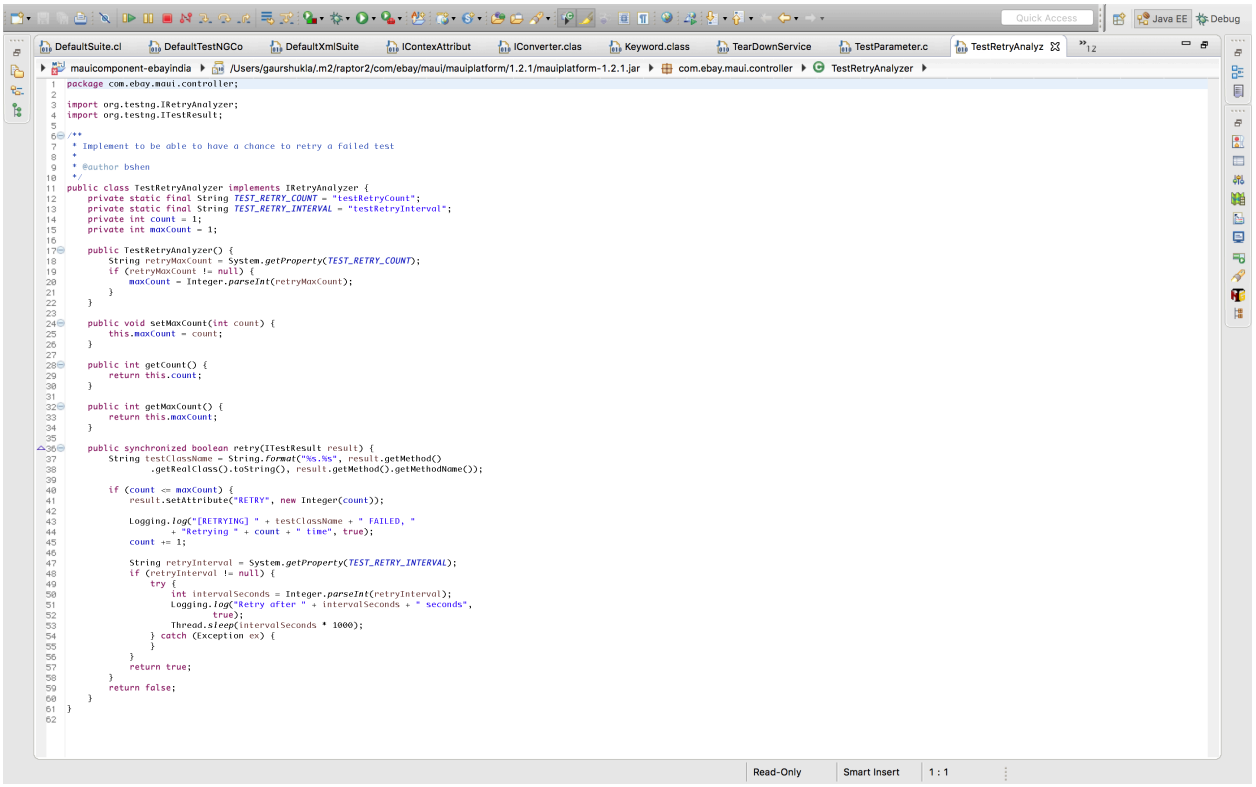

TestPlan.class
EasyFilter.clas
SpreadSheetUtil
CheckClaimTestP
AbstractPageLis
AbstractPageLis
AnnotationTrans
Assertion.class
Context.class
mauicomponent-ebayindia
/Users/gaurshukla/.m2/raptor2/com/ebay/maui/mauiplatform/1.2.1/mauiplatform-1.2.1.jar
com.ebay.maui.controller
TestPlan
logger : Logger
public abstract class TestPlan {
private static final Logger logger = Logging.getLogger(TestPlan.class);
private Date start;
static {
}
/**
* Implement an anonymous TearDownService and add it to the framework. The
* framework will execute it once the test is complete whether it's passed
* or failed does not matter.
*
* One example can be recycling of user. <code>
* <pre>
* addTearDownService(new TearDownService(){
* public void tearDown(){
* UserHelper.recycleUser(user);
* }
* });
* UserHelper.createUser(user);
* </pre>
* </code>
*
* @param service
*/
public static void addTearDownService(TearDownService service) {
ContextManager.getThreadContext().addTearDownService(service);
}
@AfterMethod(alwaysRun = true)
public void afterTestMethod(Method method) {
// Clean ups for test level services
List<TearDownService> serviceList = ContextManager.getThreadContext()
.getTearDownServices();
if (serviceList != null && !serviceList.isEmpty()) {
for (TearDownService service : serviceList) {
service.tearDown();
}
}
// threadLocalAfterMethodTearDownServiceList.set(null);
WebUXDriver.cleanUp();
APIDriver.cleanUp();
DBDriver.cleanUp();
SSHDriver.cleanUp();
logger.info(Thread.currentThread() + " Finish method "
+ method.getName());
}
@AfterSuite(alwaysRun = true)
public void afterTestSuite() {
logger.info("Test Suite Execution Time: "
+ (new Date().getTime() - start.getTime()) / 1000 / 60
+ " minutes.");
}
/**
* Configure Test Params setting
*
* @param xmlTest
*/
@BeforeTest(alwaysRun = true)
public void beforeTest(ITestContext testContex, XmlTest xmlTest) {
ContextManager.initTestLevelContext(testContex, xmlTest);
}
Read-Only
Smart Insert
30 : 56



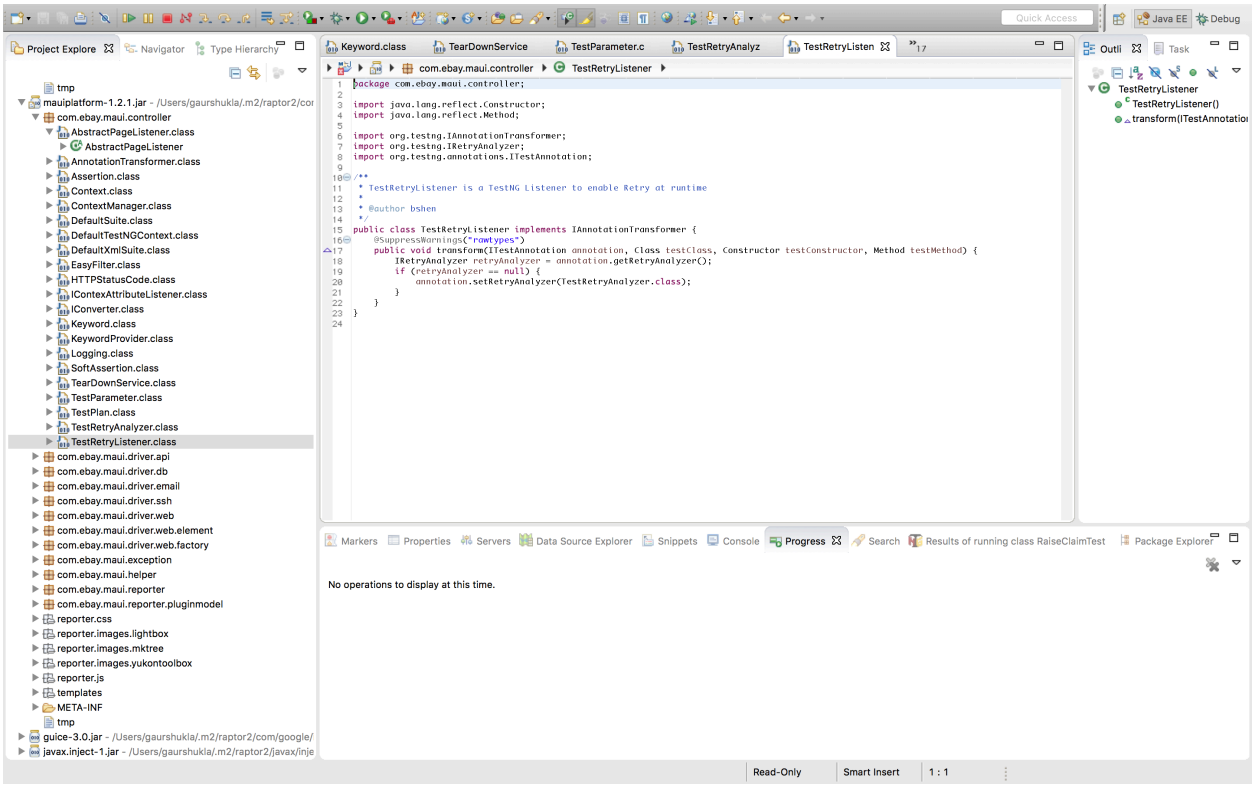
```
@BeforeMethod(alwaysRun = true)
public void beforeTestMethod(Object[] parameters, Method method,
        ITestContext testContex, XmlTest xmlTest) {
    logger.info(Thread.currentThread() + " Start method "
            + method.getName());
    ContextManager.initThreadContext(testContex, xmlTest);

    if (method != null) {
        ContextManager.getThreadContext().setAttribute(
                Context.TEST_METHOD_SIGNATURE,
                constructMethodSignature(method, parameters));
    }

}

@BeforeSuite(alwaysRun = true)
public void beforeTestSuite(ITestContext testContex) {
    start = new Date();
    System.setProperty("AUTOMATION_FRAMEWORK", "MAUI");

    ContextManager.initGlobalContext(testContex);

    // to support to call function in @beforeSuite
    ContextManager.initThreadContext(testContex, null);

    readTestParameterAnnotations();
}

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

/**
 * Remove name space from parameters
 *
 * @param parameters
 * @return
 */
private 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();
}

/**
 * Get the value of property with TestParameter annotation from system
 * property, system env, or TestNG parameter
 *
```



```
        return method.getDeclaringClass().getCanonicalName() + "."
                + method.getName() + "(" + constructParameterString(parameters)
                + ")";
    }

    /**
     * Remove name space from parameters
     *
     * @param parameters
     * @return
     */
    private 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();
    }

    /**
     * Get the value of property with TestParameter annotation from system
     * property, system env, or TestNG parameter
     *
     */
    protected void readTestParameterAnnotations() {
        try {
            Field[] fields = this.getClass().getDeclaredFields();
            AccessibleObject.setAccessible(fields, true);

            for (Field field : fields) {
                if (field.getAnnotation(TestParameter.class) != null) {
                    String name = field.getAnnotation(TestParameter.class)
                            .name();

                    if (System.getProperty(name) != null) {
                        field.set(this, System.getProperty(name));
                    } else if (System.getenv(name) != null) {
                        field.set(this, System.getenv(name));
                    } else if (ContextManager.getThreadContext()
                            .getTestNGContext().getCurrentXmlTest()
                            .getParameter(name) != null) {
                        field.set(this, ContextManager.getThreadContext()
                                .getTestNGContext().getCurrentXmlTest()
                                .getParameter(name));
                    }
                }
            }
        } catch (Exception ex) {
        }
    }
}
```



```
package com.ebay.maui.controller;

import org.testng.IRetryAnalyzer;
import org.testng.ITestResult;

/**
 * Implement to be able to have a chance to retry a failed test
 *
 * @author bshen
 */
public class TestRetryAnalyzer implements IRetryAnalyzer {
    private static final String TEST_RETRY_COUNT = "testRetryCount";
    private static final String TEST_RETRY_INTERVAL = "testRetryInterval";
    private int count = 1;
    private int maxCount = 1;

    public TestRetryAnalyzer() {
        String retryMaxCount = System.getProperty(TEST_RETRY_COUNT);
        if (retryMaxCount != null) {
            maxCount = Integer.parseInt(retryMaxCount);
        }
    }

    public void setMaxCount(int count) {
        this.maxCount = count;
    }

    public int getCount() {
        return this.count;
    }

    public int getMaxCount() {
        return this.maxCount;
    }

    public synchronized boolean retry(ITestResult result) {
        String testClassName = String.format("%s.%s", result.getMethod()
                .getRealClass().toString(), result.getMethod().getMethodName());

        if (count <= maxCount) {
            result.setAttribute("RETRY", new Integer(count));

            Logging.log("[RETRYING] " + testClassName + " FAILED, "
                    + "Retrying " + count + " time", true);
            count += 1;

            String retryInterval = System.getProperty(TEST_RETRY_INTERVAL);
            if (retryInterval != null) {
                try {
                    int intervalSeconds = Integer.parseInt(retryInterval);
                    Logging.log("Retry after " + intervalSeconds + " seconds",
                            true);
                    Thread.sleep(intervalSeconds * 1000);
                } catch (Exception ex) {
                }
            }
            return true;
        }
        return false;
    }
}
```



```
package com.ebay.maui.controller;

import java.lang.reflect.Constructor;
import java.lang.reflect.Method;

import org.testng.IAnnotationTransformer;
import org.testng.IRetryAnalyzer;
import org.testng.annotations.ITestAnnotation;

/**
 * TestRetryListener is a TestNG Listener to enable Retry at runtime
 *
 * @author bshen
 */
public class TestRetryListener implements IAnnotationTransformer {
    @SuppressWarnings("rawtypes")
    public void transform(ITestAnnotation annotation, Class testClass, Constructor testConstructor, Method testMethod) {
        IRetryAnalyzer retryAnalyzer = annotation.getRetryAnalyzer();
        if (retryAnalyzer == null) {
            annotation.setRetryAnalyzer(TestRetryAnalyzer.class);
        }
    }
}
```

No operations to display at this time.