

HELPER:

```

1 package com.ebay.mau.helper;
2
3 import java.util.List;
4
5 import com.ebay.mau.exception.APException;
6 import com.ebay.mau.exception.DataCreationException;
7 import com.ebay.mau.exception.DBException;
8 import com.ebay.mau.exception.EmailException;
9 import com.ebay.mau.exception.MauException;
10 import com.ebay.mau.exception.PageNotCurrentException;
11 import com.ebay.mau.exception.SSHException;
12 import com.ebay.mau.exception.WebResponseException;
13 import com.ebay.mau.exception.WebSessionTerminatedException;
14 import com.ebay.mau.exception.XMLException;
15 import com.ebay.mau.helper.CALHelper;
16 import com.ebay.mau.helper.ClassContextHelper;
17 import com.ebay.mau.helper.EscapeHelper;
18 import com.ebay.mau.helper.OSHelper;
19 import com.ebay.mau.helper.StringHelper;
20 import com.ebay.mau.helper.ThreadHelper;
21 import com.ebay.mau.helper.URLHelper;
22 import com.ebay.mau.helper.WaitHelper;
23 import com.ebay.mau.helper.XMLHelper;
24 import com.ebay.mau.reporter.DetailedLog;
25 import com.ebay.mau.reporter.HTMLReporter;
26 import com.ebay.mau.reporter.MiniTestResult;
27 import com.ebay.mau.reporter.PluginsUtil;
28 import com.ebay.mau.reporter.XMLReporter;
29 import com.ebay.mau.reporter.pluginmodel.Mauplugins;
30 import com.ebay.mau.reporter.pluginmodel.Method;
31 import com.ebay.mau.reporter.pluginmodel.ObjectFactory;
32 import com.ebay.mau.reporter.pluginmodel.Page;
33 import com.ebay.mau.reporter.pluginmodel.Plugin;
34 import com.ebay.mau.reporter.pluginmodel.Test;
35 import com.ebay.mau.reporter.pluginmodel.reporter.css;
36
37 /**
38 * Utility to obtain the <code>Class</code> issuing the call to the current
39 * method of execution
40 */
41 public class ClassContextHelper {
42     private static final Helper HELPER = new Helper();
43
44     /**
45      * Utility to obtain the <code>Class</code> issuing the call to the current
46      * method of execution
47      */
48     public static Class getClassEnclosing() {
49         Class[] classContext = HELPER.getContext();
50         if (classContext[3].isLocalClass())
51             // classContext[3].isLocalClass()
52             return classContext[3].getEnclosingClass();
53         return classContext[3];
54     }
55
56     public static String getCallerMethod() {
57         StackTraceElement[] stackTrace = Thread.currentThread().getStackTrace();
58         return stackTrace[3].getMethodName();
59     }
60
61     public static void main(String[] st) {
62         System.out.println(getCallerMethod()); // KEEPME
63     }
64 }
65

```

```

1 package com.ebay.mau.helper;
2
3 import java.util.Pattern;
4 import java.util.regex.Matcher;
5
6 /**
7  * Escapes all ampersand characters in a URL.
8  * An ampersand character may appear in the query string of a URL. The
9  * ampersand character is indeed valid in a URL.
10 * It is only invalid if it appears on an <a href=> attribute, and
11 * such attributes have the additional constraint that ampersands
12 * must be escaped: <a href=>.
13 *
14 * <br>
15 * The JSTL <curl> tag does indeed perform proper URL encoding of query
16 * parameters, so it does not in general produce text which is valid as
17 * an <a href=> attribute, simply because it does not escape the
18 * ampersand character. This is in contrast with multiple query parameters
19 * appearing in the URL, since it requires a little extra work.
20 */
21 public class EscapeHelper {
22
23     private static final Pattern SCRIPT = Pattern.compile("<SCRIPT>", Pattern.CASE_INSENSITIVE);
24     private static final Pattern SCRIPT_END = Pattern.compile("</SCRIPT>", Pattern.CASE_INSENSITIVE);
25
26     private static void addCharEntity(Integer aldx, StringBuilder obuilder) {
27         String padding = "";
28         if (aldx <= 9) {
29             padding = "00";
30         } else if (aldx <= 99) {
31             padding = "0";
32         } else {
33             // no prefix
34         }
35         String number = padding + aldx.toString();
36         obuilder.append("%&" + number + ";");
37     }
38
39     /**
40      * Escape all ampersand characters in a URL.
41      * An ampersand character may appear in the query string of a URL. The
42      * ampersand character is indeed valid in a URL.
43      * It is only invalid if it appears on an <a href=> attribute, and
44      * such attributes have the additional constraint that ampersands
45      * must be escaped: <a href=>.
46      *
47      * <br>
48      * The JSTL <curl> tag does indeed perform proper URL encoding of query
49      * parameters, so it does not in general produce text which is valid as
50      * an <a href=> attribute, simply because it does not escape the
51      * ampersand character. This is in contrast with multiple query parameters
52      * appearing in the URL, since it requires a little extra work.
53      */
54     public static String formHrefAmpersand(String oURL) {
55         return oURL.replace("&", "&%26");
56     }
57
58     /**
59      * Escapes characters for text appearing in HTML markup.
60      *
61      * <br>
62      * This method exists as a defence against Cross Site Scripting (XSS) hacks.
63      * The idea is to neutralize control characters commonly used by scripts,
64      * such that they will not be executed by the browser. This is done by
65      * replacing the control characters with their escaped equivalents.
66      *
67      * <br>
68      * The following characters are replaced with corresponding HTML character
69      * entities:
70      * - <table border='1' cellpadding='3' cellspacing='0'>
71      * - <tr>
72      * - <th>Character</th>
73      * - <th>Replacement</th>
74      * - <td>
75      * - <td></td>
76      * - <td></td>
77      * - <td>
78      *
79      */
80 }
81

```

```
208
209     * Note that JSTL's <%code coutput> escapes <%> only the first
210     * five '<%>' of the above characters.
211
212     @Override
213     public static String formHTML(String oText) {
214         final StringBuilder result = new StringBuilder();
215         final StringCharacterIterator iterator = new StringCharacterIterator(oText);
216         char character = iterator.current();
217         while (character != CharacterIterator.DONE) {
218             if (character == '<') {
219                 result.append("&lt;");
220             } else if (character == '>') {
221                 result.append("&gt;");
222             } else if (character == '&') {
223                 result.append("&amp;");
224             } else if (character == '#') {
225                 result.append("&#");
226             } else if (character == ' ') {
227                 result.append("&nbsp;");
228             } else if (character == '<!--') {
229                 addCharEntity(x9, result);
230             } else if (character == '-->') {
231                 addCharEntity(x33, result);
232             } else if (character == '<!--') {
233                 addCharEntity(x35, result);
234             } else if (character == '-->') {
235                 addCharEntity(x36, result);
236             } else if (character == '<!--') {
237                 addCharEntity(x37, result);
238             } else if (character == '-->') {
239                 addCharEntity(x39, result);
240             } else if (character == '<!--') {
241                 addCharEntity(x40, result);
242             } else if (character == '-->') {
243                 addCharEntity(x41, result);
244             } else if (character == '<!--') {
245                 addCharEntity(x42, result);
246             } else if (character == '-->') {
247                 addCharEntity(x43, result);
248             } else if (character == '<!--') {
249                 addCharEntity(x44, result);
250             } else if (character == '-->') {
251                 addCharEntity(x45, result);
252             } else if (character == '<!--') {
253                 addCharEntity(x46, result);
254             } else if (character == '-->') {
255                 addCharEntity(x47, result);
256             } else if (character == '<!--') {
257                 addCharEntity(x48, result);
258             } else if (character == '-->') {
259                 addCharEntity(x49, result);
260             } else if (character == '<!--') {
261                 addCharEntity(x51, result);
262             } else if (character == '-->') {
263                 addCharEntity(x53, result);
264             } else if (character == '<!--') {
265                 addCharEntity(x54, result);
266             } else if (character == '-->') {
267                 addCharEntity(x51, result);
268             } else if (character == '<!--') {
269                 addCharEntity(x52, result);
270             } else if (character == '-->') {
271                 addCharEntity(x54, result);
272             }
273         }
274     }
```



The screenshot shows an IDE interface with multiple tabs open. The current tab is 'EscapeHelper.java' from the package 'com.ebay.mau.helper'. The code implements two static methods: `forMaiComponent` and `forRegex`. The `forMaiComponent` method iterates through a character iterator, checking each character against various patterns to build a result string. The `forRegex` method does a similar thing, but it handles regex special characters like ., *, +, ?, ^, \$, |, and \ by escaping them. Both methods use a `StringBuilder` to construct the final result.

```
271     else if (character == '&') {
272         addCharEntity(04, result);
273     } else if (character == '<') {
274         addCharEntity(05, result);
275     } else if (character == '>') {
276         addCharEntity(06, result);
277     } else if (character == '&lt;') {
278         addCharEntity(123, result);
279     } else if (character == '&gt;') {
280         addCharEntity(124, result);
281     } else if (character == '&gt;') {
282         addCharEntity(125, result);
283     } else if (character == '&lt;') {
284         addCharEntity(126, result);
285     } else {
286         // the char is not a special one
287         // add it to the result as is
288         result.append(character);
289     }
290     character = iterator.next();
291 }
292 return result.toString();
293 }
294 /**
295  * Replace characters having special meaning in regular expressions with
296  * their escaped equivalents, preceded by a '\` character.
297  */
298 /**
299  * The escaped characters include :
300  * - \u002a
301  * - \u002b
302  * - \u002c
303  * - \u002d
304  * - \u002e
305  * - \u002f
306  * - \u0028 and \u0029
307  * - \u002b and \u0029
308  * - \u002d and \u0029
309  * - \u002e and \u0029
310  * - \u002f and \u0029
311  */
312 public static String forRegex(String oRegexFragment) {
313     final StringBuilder result = new StringBuilder();
314
315     final StringCharacterIterator iterator = new StringCharacterIterator(oRegexFragment);
316     char character = iterator.current();
317     while (character != CharacterIterator.DONE) {
318
319         /*
320          * All literals need to have backslashes doubled.
321          */
322         if (character == '\u002a') {
323             result.append("\\\\u002a");
324         } else if (character == '\\') {
325             result.append("\\\\\\\\");
326         } else if (character == '?') {
327             result.append("\\\\\\?");
328         } else if (character == '\u002b') {
329             result.append("\\\\u002b");
330         } else if (character == '\u002d') {
331             result.append("\\\\u002d");
332         } else if (character == '\u002e') {
333             result.append("\\\\u002e");
334         } else if (character == '\u002f') {
335             result.append("\\\\u002f");
336         }
337     }
338 }
```

This screenshot shows the Java code for the `EscapeHelper` class in an IDE. The code is annotated with various Javadoc-style comments explaining its functionality.

```
335     result.append("\\\\\"");
336     } else if (character == '\\') {
337         result.append("\\\\\"");
338     } else if (character == '\"') {
339         result.append("\\\\\"");
340     } else if (character == '“') {
341         result.append("\\\\\"\\“");
342     } else if (character == '”') {
343         result.append("\\\\\"\\”");
344     } else if (character == '‘') {
345         result.append("\\\\“\\‘");
346     } else if (character == '’') {
347         result.append("\\\\“\\’");
348     } else if (character == '^') {
349         result.append("\\\\^");
350     } else if (Character.isISOControl(character)) {
351         result.append("\\\\u00" + Integer.toHexString(character));
352     } else if (character >= 0x80) {
353         // the char is not a special one
354         // add it to the result as is
355         result.append(character);
356     }
357     character = iterator.next();
358 }
359 return result.toString();
360 }
361 /**
362 * Escapes <t>'$'</t> and <t>`</t> characters in replacement strings.
363 */
364 private static final StringMatcher quoteReplacement(String text) {
365     // ...
366     // Synonym for <t>Matcher.quoteReplacement(String)</t>.
367     // ...
368     // The following methods use replacement strings which treat <t>'$'</t>
369     // and <t>`</t> as special characters:
370     // ...
371     // <t>String.replaceAll(String, String)</t>;
372     // <t>String.replaceFirst(String, String)</t>;
373     // <t>String.replace(StringBuffer, String)</t>;
374     // ...
375     // ...
376     // ...
377     // If replacement text can contain arbitrary characters, then you will
378     // usually need to escape that text, to ensure special characters are
379     // interpreted literally.
380     // ...
381     public static String forReplacementString(String input) {
382         return Matcher.quoteReplacement(input);
383     }
384 }
385 /**
386 * Disable all <t><SCRIPT></t> tags in <t><dfExt></t>,
387 */
388 /**
389 * <t>In-sensitive to case.
390 */
391 /**
392 * public static String forScriptTagsOnly(String text) {
393     String result = null;
394     Matcher matcher = SCRIPT.matcher(text);
395     result = matcher.replaceAll("<!-->");
396     matcher = SCRIPT_END.matcher(result);
397     result = matcher.replaceAll("<!--><!--");
398     return result;
399 }
400 */

419 /**
420 * public static String forURL(String urlFragment) {
421     String result = null;
422     try {
423         result = URLencoder.encode(urlFragment, "UTF-8");
424     } catch (UnsupportedEncodingException e) {
425         throw new RuntimeException("UTF-8 not supported", e);
426     }
427     return result;
428 }
429 /**
430 * Escape characters for text appearing as XML data, between tags.
431 */
432 /**
433 * The following characters are replaced with corresponding character
434 * entities:
435 * -&lt;table border='1' cellpadding='3' cellspacing='0'&gt;
436 *   &lt;tr&gt;
437 *     &lt;th&gt;Character&lt;/th&gt;
438 *     &lt;th&gt;Encoding&lt;/th&gt;
439 *   &lt;/tr&gt;
440 *   &lt;tr&gt;
441 *     &lt;td&gt;&lt;!--&gt;
442 *     &lt;td&gt;&lt;!--&gt;
443 *     &lt;td&gt;&lt;!--&gt;
444 *   &lt;/tr&gt;
445 *   &lt;tr&gt;
446 *     &lt;td&gt;&lt;!--&gt;
447 *     &lt;td&gt;&amp;nbsp;&lt;/td&gt;
448 *   &lt;/tr&gt;
449 *   &lt;tr&gt;
450 *     &lt;td&gt;&lt;!--&gt;
451 *     &lt;td&gt;&amp;quot;&lt;/td&gt;
452 *   &lt;/tr&gt;
453 *   &lt;tr&gt;
454 *     &lt;td&gt;&lt;!--&gt;
455 *     &lt;td&gt;&amp;#xA0;&lt;/td&gt;
456 *   &lt;/tr&gt;
457 * &lt;/table&gt;
458 *
459 * Note that JSTL's &lt;code&gt;&lt;c:out&gt;&lt;/code&gt; escapes the exact same set of characters
460 * as this method. &lt;code&gt;&lt;span class='highlight'&gt;That is, &lt;code&gt;&lt;c:out&gt;&lt;/code&gt; is good
461 * for escaping to produce valid XML, but not for producing safe
462 * HTML.&lt;/code&gt;
463 */
464 /**
465 * public static String forXML(String text) {
466 *     StringBuilder result = new StringBuilder();
467 *     final StringCharacterIterator iterator = new StringCharacterIterator(text);
468 *     char character = iterator.current();
469 *     while (character != CharacterIterator.DONE) {
470 *         if (character == '&lt;') {
471 *             result.append("&amp;lt;");
472 *         } else if (character == '&gt;') {
473 *             result.append("&amp;gt;");
474 *         } else if (character == '&amp;gt;') {
475 *             result.append("&amp;quot;");
476 *         } else if (character == '&amp;gt;') {
477 *             result.append("&amp;#039;");
478 *         } else if (character == '&amp;gt;') {
479 *             result.append("&amp;#039;");
480 *         } else if (character == '&amp;gt;') {
481 *             result.append("&amp;#039;");
482 *         }
483 *     }
484 *     return result.toString();
485 * }
486 */

487 /**
488 * public static String forHTML(String text) {
489 *     StringBuilder result = new StringBuilder();
490 *     final StringCharacterIterator iterator = new StringCharacterIterator(text);
491 *     char character = iterator.current();
492 *     while (character != CharacterIterator.DONE) {
493 *         if (character == '&lt;') {
494 *             result.append("&amp;lt;");
495 *         } else if (character == '&gt;') {
496 *             result.append("&amp;gt;");
497 *         } else if (character == '&amp;gt;') {
498 *             result.append("&amp;quot;");
499 *         } else if (character == '&amp;gt;') {
500 *             result.append("&amp;#039;");
501 *         } else if (character == '&amp;gt;') {
502 *             result.append("&amp;#039;");
503 *         } else if (character == '&amp;gt;') {
504 *             result.append("&amp;#039;");
505 *         }
506 *     }
507 *     return result.toString();
508 * }
509 */

510 /**
511 * public static String forText(String text) {
512 *     return forHTML(text);
513 * }
514 */

515 /**
516 * public static String forTextWithEscapedAmpersand(String text) {
517 *     return forText(text).replace("&amp;", "&amp;#039;");
518 * }
519 */

520 /**
521 * public static String forTextWithEscapedAmpersandAndApostrophe(String text) {
522 *     return forText(text).replace("&amp;gt;", "&amp;#039;").replace("&amp;gt;", "&amp;quot;");
523 * }
524 */

525 /**
526 * public static String forTextWithEscapedAmpersandAndApostropheAndAmpersand(String text) {
527 *     return forText(text).replace("&amp;gt;", "&amp;#039;").replace("&amp;gt;", "&amp;quot;").replace("&amp;gt;", "&amp;#039;");
528 * }
529 */

530 /**
531 * public static String forTextWithEscapedAmpersandAndApostropheAndAmpersandAndApostrophe(String text) {
532 *     return forText(text).replace("&amp;gt;", "&amp;#039;").replace("&amp;gt;", "&amp;quot;").replace("&amp;gt;", "&amp;#039;").replace("&amp;gt;", "&amp;quot;");
533 * }
534 */

535 /**
536 * public static String forTextWithEscapedAmpersandAndApostropheAndAmpersandAndApostropheAndAmpersand(String text) {
537 *     return forText(text).replace("&amp;gt;", "&amp;#039;").replace("&amp;gt;", "&amp;quot;").replace("&amp;gt;", "&amp;#039;").replace("&amp;gt;", "&amp;quot;").replace("&amp;gt;", "&amp;#039;");
538 * }
539 */

540 /**
541 * public static String forTextWithEscapedAmpersandAndApostropheAndAmpersandAndApostropheAndAmpersandAndApostrophe(String text) {
542 *     return forText(text).replace("&amp;gt;", "&amp;#039;").replace("&amp;gt;", "&amp;quot;").replace("&amp;gt;", "&amp;#039;").replace("&amp;gt;", "&amp;quot;").replace("&amp;gt;", "&amp;#039;").replace("&amp;gt;", "&amp;quot;");
543 * }
544 */

545 /**
546 * public static String forTextWithEscapedAmpersandAndApostropheAndAmpersandAndApostropheAndAmpersandAndApostropheAndAmpersand(String text) {
547 *     return forText(text).replace("&amp;gt;", "&amp;#039;").replace("&amp;gt;", "&amp;quot;").replace("&amp;gt;", "&amp;#039;").replace("&amp;gt;", "&amp;quot;").replace("&amp;gt;", "&amp;#039;").replace("&amp;gt;", "&amp;quot;").replace("&amp;gt;", "&amp;#039;");</pre>
```

This screenshot shows the Java code for the `EscapeHelper` class in an IDE. The code is annotated with various Javadoc-style comments explaining its functionality.

```
419 /**
420 * public static String forURL(String urlFragment) {
421     String result = null;
422     try {
423         result = URLencoder.encode(urlFragment, "UTF-8");
424     } catch (UnsupportedEncodingException e) {
425         throw new RuntimeException("UTF-8 not supported", e);
426     }
427     return result;
428 }
429 /**
430 * Escape characters for text appearing as XML data, between tags.
431 */
432 /**
433 * The following characters are replaced with corresponding character
434 * entities:
435 * -<table border='1' cellpadding='3' cellspacing='0'>
436 *   <tr>
437 *     <th>Character</th>
438 *     <th>Encoding</th>
439 *   </tr>
440 *   <tr>
441 *     <td><!-->
442 *     <td><!-->
443 *     <td><!-->
444 *   </tr>
445 *   <tr>
446 *     <td><!-->
447 *     <td>&nbsp;</td>
448 *   </tr>
449 *   <tr>
450 *     <td><!-->
451 *     <td>&quot;</td>
452 *   </tr>
453 *   <tr>
454 *     <td><!-->
455 *     <td>&#xA0;</td>
456 *   </tr>
457 * </table>
458 *
459 * Note that JSTL's <code><c:out></code> escapes the exact same set of characters
460 * as this method. <code><span class='highlight'>That is, <code><c:out></code> is good
461 * for escaping to produce valid XML, but not for producing safe
462 * HTML.</code>
463 */
464 /**
465 * public static String forXML(String text) {
466 *     StringBuilder result = new StringBuilder();
467 *     final StringCharacterIterator iterator = new StringCharacterIterator(text);
468 *     char character = iterator.current();
469 *     while (character != CharacterIterator.DONE) {
470 *         if (character == '<') {
471 *             result.append("&lt;");
472 *         } else if (character == '>') {
473 *             result.append("&gt;");
474 *         } else if (character == '&gt;') {
475 *             result.append("&quot;");
476 *         } else if (character == '&gt;') {
477 *             result.append("&#039;");
478 *         } else if (character == '&gt;') {
479 *             result.append("&#039;");
480 *         }
481 *     }
482 *     return result.toString();
483 * }
484 *

485 /**
486 * public static String forHTML(String text) {
487 *     StringBuilder result = new StringBuilder();
488 *     final StringCharacterIterator iterator = new StringCharacterIterator(text);
489 *     char character = iterator.current();
490 *     while (character != CharacterIterator.DONE) {
491 *         if (character == '<') {
492 *             result.append("&lt;");
493 *         } else if (character == '>') {
494 *             result.append("&gt;");
495 *         } else if (character == '&gt;') {
496 *             result.append("&quot;");
497 *         } else if (character == '&gt;') {
498 *             result.append("&#039;");
499 *         } else if (character == '&gt;') {
500 *             result.append("&#039;");
501 *         }
502 *     }
503 *     return result.toString();
504 * }
505 *

506 /**
507 * public static String forText(String text) {
508 *     return forHTML(text);
509 * }
510 *

511 /**
512 * public static String forTextWithEscapedAmpersand(String text) {
513 *     return forText(text).replace("&", "&#039;");
514 * }
515 *

516 /**
517 * public static String forTextWithEscapedAmpersandAndApostrophe(String text) {
518 *     return forText(text).replace("&gt;", "&#039;").replace("&gt;", "&quot;");
519 * }
520 *

521 /**
522 * public static String forTextWithEscapedAmpersandAndApostropheAndAmpersand(String text) {
523 *     return forText(text).replace("&gt;", "&#039;").replace("&gt;", "&quot;").replace("&gt;", "&#039;");
524 * }
525 *

526 /**
527 * public static String forTextWithEscapedAmpersandAndApostropheAndAmpersandAndApostrophe(String text) {
528 *     return forText(text).replace("&gt;", "&#039;").replace("&gt;", "&quot;").replace("&gt;", "&#039;").replace("&gt;", "&quot;");
529 * }
530 *

531 /**
532 * public static String forTextWithEscapedAmpersandAndApostropheAndAmpersandAndApostropheAndAmpersand(String text) {
533 *     return forText(text).replace("&gt;", "&#039;").replace("&gt;", "&quot;").replace("&gt;", "&#039;").replace("&gt;", "&quot;").replace("&gt;", "&#039;");
```

The screenshot shows the Eclipse IDE interface with the 'FileHelper' class selected in the left navigation bar. The main editor window displays the 'EscapeHelper.java' code. The code is annotated with numerous Javadoc-style comments explaining the logic for escaping characters in XML. It includes methods for escaping XML attributes and strings, handling special characters like '&', '<', and '>', and dealing with null or empty inputs.

```
458 *      * @param
459 *      *      if this method, ->span class="highlight"><That is, {code <cout>} is good
460 *      *      for escaping to produce valid XML, but not for producing safe
461 *      *      HTML.</span>
462 *
463 *      public static String forXML(String aText) {
464 *          final StringBuilder result = new StringBuilder();
465 *          final StringCharacterIterator iterator = new StringCharacterIterator(aText);
466 *          char character = iterator.current();
467 *          while (character != CharacterIterator.DONE) {
468 *              if (character == '<') {
469 *                  result.append("&lt;");
470 *              } else if (character == '>') {
471 *                  result.append("&gt;");
472 *              } else if (character == '&') {
473 *                  result.append("&amp;");
474 *              } else if (character == '\"') {
475 *                  result.append("&quot;");
476 *              } else if (character == '\'') {
477 *                  result.append("&#039;");
478 *              } else if (character == '#') {
479 *                  result.append("&#");
480 *              } else {
481 *                  // the char is not a special one
482 *                  // add it to the result as is
483 *                  result.append(character);
484 *              }
485 *              character = iterator.next();
486 *          }
487 *          return result.toString();
488 *      }
489 *
490 *      public static void main(String[] args) {
491 *          System.out.println(EscapeHelper.forHTML("cable, satellite"));
492 *      }
493 *
494 *      /**
495 *       * Returns <t>-aText</t> with all <t>'</t> and <t>></t> characters
496 *       * replaced by their escaped equivalents.
497 *       */
498 *      public static String toIsobigLog(String aText) {
499 *          final StringBuilder result = new StringBuilder();
500 *          final StringCharacterIterator iterator = new StringCharacterIterator(aText);
501 *          char character = iterator.current();
502 *          while (character != CharacterIterator.DONE) {
503 *              if (character == '<') {
504 *                  result.append("&lt;");
505 *              } else if (character == '>') {
506 *                  result.append("&gt;");
507 *              } else {
508 *                  // the char is not a special one
509 *                  // add it to the result as is
510 *                  result.append(character);
511 *              }
512 *              character = iterator.next();
513 *          }
514 *          return result.toString();
515 *      }
516 *
517 *      private Escapehelper() {
518 *          // empty - prevent construction
519 *      }
520 *
521 *  }
```

The screenshot shows the Eclipse IDE interface with the 'FileHelper' class selected in the left navigation bar. The main editor window displays the 'FileHelper.java' code. This code handles file operations, specifically extracting jar files and copying files between store locations. It uses Java's ZipInputStream and ZipOutputStream classes to manipulate jar files and creates destination parent directories as needed.

```
37 public class FileHelper {
38     static Logger logger = Logger.getLogger(FileHelper.class);
39     final static int BUFFER = 2048;
40
41     private Map<Integer, FileLock> fileLockMap = new HashMap<Integer, FileLock>();
42
43     public static void extractJar(String storelocation, Class<?> clz)
44         throws IOException {
45         File ffpfile = new File(storelocation);
46         String location = clz.getProtectionDomain().getCodeSource()
47             .getLocation().toURI().getPath();
48         location = FileHelper.decodePath(location);
49         Jarfile jar = new Jarfile(location);
50         System.out.println("extracting " + location);
51         Jarfile destjar = new Jarfile();
52         ffprofile.mkdir();
53         // System.out.println("FirefoxProfile: " + profilePath +
54         // "/UserContent/stylesheet.css");
55         Enumeration<Jarfile> entriesO;
56         while (entriesO.hasMoreElements()) {
57             ZipEntry entry = (ZipEntry) entriesO.nextElement();
58             String currentName = entry.getName();
59             File destfile = new File(storelocation, currentName);
60             File destinationParent = destfile.getParentFile();
61
62             // create the parent directory structure if needed
63             destinationParent.mkdirs();
64             if (Entry.isDirectory()) {
65                 BufferedInputStream is = new BufferedInputStream(
66                     currentName.getInputStream());
67                 int currentByte;
68                 // establish buffer for writing file
69                 byte data[] = new byte[BUFFER];
70
71                 // write the current file to disk
72                 FileOutputStream fos = new FileOutputStream(destfile);
73                 BufferedOutputStream dest = new BufferedOutputStream(fos,
74                     BUFFER);
75
76                 // read and write until last byte is encountered
77                 while ((currentByte = is.read(data, 0, BUFFER)) != -1) {
78                     dest.write(data, 0, currentByte);
79                 }
80                 dest.flush();
81                 dest.close();
82                 is.close();
83             }
84         }
85         FileUtil.deleteDirectory(new File(storelocation + "\\META-INF"));
86         if (osHelper.isWindows()) {
87             new File(storelocation + "\\"
88                 + clz.getCanonicalName().replaceAll("\\\\", "\\\\\\\\")
89                 + ".class").delete();
90         } else {
91             new File(storelocation + "\\"
92                 + clz.getCanonicalName().replaceAll("\\\\", "/") + ".class")
93                 .delete();
94         }
95     }
96
97     public static void copyfile(File srcPath, File dstPath) throws IOException {
98
99         if (srcPath.isDirectory()) {
100            if (!dstPath.exists()) {
101                dstPath.mkdir();
102            }
103        }
104    }
105}
```

FileHelper.java

```
101     }
102
103     String files[] = srcPath.listFiles();
104     for (int i = 0; i < files.length; i++) {
105         copyFile(new File(srcPath), files[i], new File(dstPath),
106                 files[i]);
107     }
108 } else {
109     if (!srcPath.exists()) {
110         throw new IOException("DIR_NOT_FOUND: " + srcPath);
111     } else {
112         InputStream in = null;
113         OutputStream out = null;
114         try {
115             if (dstPath.getParentFile().exists())
116                 dstPath.getParentFile().mkdirs();
117
118             in = new FileInputStream(srcPath);
119             out = new FileOutputStream(dstPath);
120
121             // Transfer bytes from in to out
122             byte[] buf = new byte[1024];
123             int len;
124
125             while ((len = in.read(buf)) > 0) {
126                 out.write(buf, 0, len);
127             }
128         } finally {
129             if (in != null) {
130                 try {
131                     in.close();
132                 } catch (Exception e) {
133                 }
134             }
135             if (out != null) {
136                 try {
137                     out.close();
138                 } catch (Exception e) {
139                 }
140             }
141         }
142     }
143 }
144
145 public static void copyFile(String srcPath, String dstPath)
146     throws IOException {
147     copyFile(new File(srcPath), new File(dstPath));
148 }
149
150 /**
151 * Deletes a Directory with Files
152 * @param path
153 * @return
154 */
155
156 public static boolean deleteDirectory(File path) {
157     if (!path.exists())
158         return true;
159     if (path.listFiles() == null)
160         return true;
161     for (int i = 0; i < path.listFiles().length; i++) {
162         if (path.listFiles()[i].isDirectory()) {
163             deleteDirectory(path.listFiles()[i]);
164         } else {
165             path.listFiles()[i].delete();
166         }
167     }
168     return path.delete();
169 }
170
171 public static boolean deleteDirectory(String dir) {
172     return deleteDirectory(new File(dir));
173 }
174
175 /**
176 * Locks a profile to use
177 * @param profilePath
178 * @param max
179 * @throws IOException
180 */
181
182 public static int performfilelock(String profilePath, int max)
183     throws IOException {
184
185     FileLock lock = null;
186
187     File dir = new File(profilePath);
188     if (!dir.exists())
189         dir.mkdirs();
190
191     RandomAccessFile raf = null;
192     FileChannel filechannel = null;
193
194     for (int i = 1; i <= max; i++) {
195         try {
196             logger.info("Trying to lock profile_" + i);
197
198             raf = new RandomAccessFile(profilePath + "/" + i + ".lock", "rw");
199             filechannel = raf.getChannel();
200             lock = filechannel.tryLock();
201             if (lock != null && lock.isValid() && lock.isShared()) {
202                 File pdir = new File(profilePath + "/profile_" + i);
203                 if (!pd़ir.exists())
204                     pd़ir.mkdirs();
205
206                 fileLockMap.put(Integer.valueOf(i), lock);
207                 logger.info("Successfully locked profile_" + i);
208                 return i;
209             } else {
210                 lock = null;
211                 filechannel.close();
212                 raf.close();
213             } catch (Exception e) {
214                 logger.error("Error while trying to lock profile_" + i);
215             }
216         } catch (Exception ie) {
217             logger.error("Error while trying to lock profile_" + i);
218             ie.printStackTrace();
219         } finally {
220             if (lock != null)
221                 lock.release();
222             if (raf != null)
223                 raf.close();
224             if (filechannel != null)
225                 filechannel.close();
226         }
227     }
228 }
229 }
```

FileHelper.java

```
165     }
166 }
167
168 public static boolean deleteDirectory(String dir) {
169     return deleteDirectory(new File(dir));
170 }
171
172 /**
173 * Locks a profile to use
174 * @param profilePath
175 * @param max
176 * @return
177 * @throws IOException
178 */
179 public static int performfilelock(String profilePath, int max)
180     throws IOException {
181
182     FileLock lock = null;
183
184     File dir = new File(profilePath);
185     if (!dir.exists())
186         dir.mkdirs();
187
188     RandomAccessFile raf = null;
189     FileChannel filechannel = null;
190
191     for (int i = 1; i <= max; i++) {
192         try {
193             logger.info("Trying to lock profile_" + i);
194
195             raf = new RandomAccessFile(profilePath + "/" + i + ".lock", "rw");
196             filechannel = raf.getChannel();
197             lock = filechannel.tryLock();
198             if (lock != null && lock.isValid() && lock.isShared()) {
199                 File pdir = new File(profilePath + "/profile_" + i);
200                 if (!pd़ir.exists())
201                     pd़ir.mkdirs();
202
203                 fileLockMap.put(Integer.valueOf(i), lock);
204                 logger.info("Successfully locked profile_" + i);
205                 return i;
206             } else {
207                 lock = null;
208                 filechannel.close();
209                 raf.close();
210             } catch (Exception e) {
211                 logger.error("Error while trying to lock profile_" + i);
212             }
213         } catch (Exception ie) {
214             logger.error("Error while trying to lock profile_" + i);
215             ie.printStackTrace();
216         } finally {
217             if (lock != null)
218                 lock.release();
219             if (raf != null)
220                 raf.close();
221             if (filechannel != null)
222                 filechannel.close();
223         }
224     }
225 }
```

```

231     throw new IOException("UNABLE_TO_LOCK_PROFILE");
232   }
233 
234   /**
235    * Locking the port
236    *
237    * @param startPort
238    * @param max
239    * @return
240   */
241   public static int performPortLock(int startPort, int max) {
242 
243     for (int i = startPort; i <= startPort + max; i++) {
244       try {
245         // Create a server socket
246         final ServerSocket ss = new ServerSocket(i);
247         Runnable rble = new Runnable() {
248           public void run() {
249             try {
250               ss.accept();
251             } catch (Exception e) {
252               } finally {
253                 try {
254                   logger.info("Releasing lock on port " +
255                               ss.getLocalPort());
256                   ss.close();
257                   logger.info("Done");
258                 } catch (Exception e) { // KEEPME
259                 }
260               }
261             }
262           Thread thr = new Thread(rble);
263           thr.start();
264           return i;
265         } catch (Exception e) {
266         }
267       }
268     }
269     return 0;
270   }
271 
272   /**
273    * Read contents of a file
274    *
275    * @param path
276    * @returns content
277    * @throws IOException
278   */
279   public static String readFromFile(File path) throws IOException {
280     FileInputStream fis = null;
281     try {
282       fis = new FileInputStream(path);
283       return readFromFile(fis);
284     } finally {
285       if (fis != null)
286         fis.close();
287     }
288   }
289 
290   /**
291    * Read contents From Stream
292    *
293    * @param path
294    * @returns content
295    * @throws IOException
296   */
297   public static String readFromFile(InputStream path) throws IOException {
298     InputStreamReader fr = null;
299     BufferedReader br = null;
300     StringBuffer sbContent = new StringBuffer();
301 
302     try {
303       fr = new InputStreamReader(path);
304       br = new BufferedReader(fr);
305 
306       String line = null;
307       while ((line = br.readLine()) != null)
308         sbContent.append(line).append("\n");
309     } finally {
310       if (br != null) {
311         try {
312           br.close();
313         } catch (IOException e) {
314         }
315       }
316       if (fr != null) {
317         try {
318           fr.close();
319         } catch (IOException e) {
320         }
321       }
322     }
323     return sbContent.toString();
324   }
325 
326   /**
327    * Releases filelock given a profile number
328    *
329    * @param i
330   */
331   public static void releaseFilelock(int i) {
332     Filelock lock = filelockMap.get(Integer.valueOf(i));
333     logger.info("Lock for profile_" + i + " released");
334     if (lock != null) {
335       try {
336         FileChannel channel = lock.channel();
337         lock.release();
338         channel.close();
339       } catch (IOException e) {
340         logger.error("Failed to release lock for profile_" + i);
341       }
342     }
343   }
344 
345   /**
346    * Release port lock
347    */
348   public static void releasePortLock(int port) {
349     Socket socket = null;
350     OutputStream os = null;
351     try {
352       socket = new Socket("127.0.0.1", port);
353       os = socket.getOutputStream();
354     } catch (Exception e) {
355       logger.warn("Ex", e);
356     } finally {
357       try {
358         os.close();
359       } catch (Exception e) {
360       }
361     }
362   }

```

The screenshot shows the Java code for `FileHelper.java` in an IDE. The code includes methods for writing images from byte arrays and reading binary files.

```
362     }
363     */
364     /* Creates Image From bytes and stores it
365      * @param path
366      * @param screenShot
367      */
368     public static synchronized void writeImage(String path, byte[] byteArray) {
369         if (byteArray.length == 0)
370             return;
371         System.gc(); // KEEPME
372         InputStream in = null;
373         FileOutputStream fos = null;
374         File parentDir = new File(path).getParentFile();
375         if (!parentDir.exists())
376             parentDir.mkdirs();
377         byte[] decodedBytes = Base64.decodeBase64(byteArray);
378         ByteArrayInputStream img = ImageIO.read(new ByteArrayInputStream(decodedBytes));
379         BufferedImage imgObj = ImageIO.read(img);
380         fos = new FileOutputStream(path);
381         ImageIO.write(imgObj, "png", fos);
382         img = null;
383     } catch (Exception e) {
384         logger.warn(e.getMessage());
385     } finally {
386         if (in != null) {
387             try {
388                 in.close();
389             } catch (Exception e) {
390             }
391         }
392         if (fos != null) {
393             try {
394                 fos.close();
395             } catch (Exception e) {
396             }
397         }
398     }
399 }
400
401 public static byte[] readBinaryFile(String aInputFileName) {
402     logger.info("Reading in binary file named :" + aInputFileName);
403     File file = new File(aInputFileName);
404     logger.info("file size : " + file.length());
405     byte[] result = new byte[(int) file.length()];
406     try {
407         InputStream input = null;
408         try {
409             int totalBytesRead = 0;
410             input = new BufferedInputStream(new FileInputStream(file));
411             while ((totalBytesRead < result.length) && (input.read(result) != -1)) {
412                 totalBytesRead = result.length - totalBytesRead;
413                 // input.read() returns -1, 0, or more ;
414                 int bytesRead = input.read(result, totalBytesRead,
415                     bytesRemaining);
416                 if (bytesRead > 0) {
417                     totalBytesRead = totalBytesRead + bytesRead;
418                 }
419             }
420         } catch (FileNotFoundException ex) {
421             logger.info("File not found.");
422         } catch (IOException ex) {
423             logger.info(ex);
424         }
425         return result;
426     } finally {
427         /*
428          * Write to file
429          * @param path
430          * @param is
431          * @throws IOException
432          */
433         public static void writeToFile(String path, InputStream is) throws IOException {
434             System.gc(); // KEEPME
435             BufferedInputStream bis = new BufferedInputStream(is);
436             BufferedOutputStream bos = new BufferedOutputStream(
437                 new FileOutputStream(path));
438             try {
439                 File parentDir = new File(path).getParentFile();
440                 if (!parentDir.exists())
441                     parentDir.mkdirs();
442                 byte[] bArr = new byte[4096];
443                 int mRead = 0;
444                 while ((mRead = bis.read(bArr)) != -1) {
445                     bos.write(bArr, 0, mRead);
446                 }
447             } finally {
448                 try {
449                     bis.close();
450                 } catch (Exception e) {
451                 }
452                 try {
453                     is.close();
454                 } catch (Exception e) {
455                 }
456             }
457         }
458     }
459     /*
460      * Saves HTML Source
461      * @param path
462      * @param selenium
463      * @throws Exception
464      */
465 }
```

The screenshot shows the Java code for `FileHelper.java` in an IDE. The code includes methods for writing images from byte arrays and reading binary files.

```
433     }
434     */
435     public static void writeToFile(String path, InputStream is) throws IOException {
436         System.gc(); // KEEPME
437         BufferedInputStream bis = new BufferedInputStream(is);
438         BufferedOutputStream bos = new BufferedOutputStream(
439             new FileOutputStream(path));
440         try {
441             File parentDir = new File(path).getParentFile();
442             if (!parentDir.exists())
443                 parentDir.mkdirs();
444             byte[] bArr = new byte[4096];
445             int mRead = 0;
446             while ((mRead = bis.read(bArr)) != -1) {
447                 bos.write(bArr, 0, mRead);
448             }
449         } finally {
450             try {
451                 bis.close();
452             } catch (Exception e) {
453             }
454             try {
455                 is.close();
456             } catch (Exception e) {
457             }
458         }
459     }
460     /*
461      * Saves HTML Source
462      * @param path
463      * @param selenium
464      * @throws Exception
465      */
466 }
```

The screenshot shows the Java code for `FileHelper.java` in an IDE. The code includes methods for writing to files, getting the latest file in a folder, and decoding URLs. It uses `OutputStreamWriter`, `BufferedWriter`, and `FileInputStream` classes.

```
498     public static void writeToFile(String path, String content) throws IOException {
499         System.gc(); // KEEPME
500         FileOutputStream fos = null;
501         OutputStreamWriter osw = null;
502         BufferedWriter bw = null;
503         try {
504             File parentDir = new File(path).getParentFile();
505             if (!parentDir.exists())
506                 parentDir.mkdirs();
507             catch (Exception ignore) {
508             }
509             fos = new FileOutputStream(path);
510             osw = new OutputStreamWriter(fos, "UTF-8");
511             bw = new BufferedWriter(osw);
512             bw.write(content);
513         } finally {
514             if (bw != null) {
515                 try {
516                     bw.close();
517                 } catch (Exception e) {
518                     } // KEEPME
519             }
520             if (osw != null) {
521                 try {
522                     osw.close();
523                 } catch (Exception e) {
524                     } // KEEPME
525             }
526             if (fos != null) {
527                 try {
528                     fos.close();
529                 } catch (Exception e) {
530                     } // KEEPME
531             }
532             if (fos != null) {
533                 try {
534                     fos.close();
535                 } catch (Exception e) {
536                     } // KEEPME
537         }
538     }
539     public static String getLatestFile(String folder) {
540         String file = null;
541         File folderFile = new File(folder);
542         if (!folderFile.exists() && !folderFile.isDirectory()) {
543             file[] files = folderFile.listFiles();
544             long date = 0;
545             for (int i = 0; i < files.length; i++) {
546                 if (files[i].lastModified() > date) {
547                     date = files[i].lastModified();
548                     file = files[i].getAbsolutePath();
549                 }
550             }
551         }
552         return file;
553     }
554     public static String decodePath(String path)
555         throws UnsupportedEncodingException {
556         return URLDecoder.decode(path, "UTF-8");
557     }
558 }
```

The screenshot shows the Java code for `OSHelper.java` in an IDE. The code includes methods for determining the operating system type, executing commands, and reading registry values. It uses `Runtime`, `Process`, and `BufferedReader` classes.

```
8     public class OSHelper {
9         public static String getOSName(){
10             return System.getProperty("os.name");
11         }
12         public static boolean isMac(){
13             return getOSName().startsWith("Mac");
14         }
15         public static boolean isWindows(){
16             return getOSName().startsWith("Win");
17         }
18         public static String getOSBits(){
19             return System.getProperty("os.arch");
20         }
21         public static boolean is32(){
22             return getOSBits().equals("x86");
23         }
24         public static boolean is64(){
25             if (isWindows())
26                 {
27                     return System.getenv("ProgramW6432") != null;
28                 }
29             else
30                 return !getOSBits().equals("x86");
31         }
32         private static List<String> executeCommand(String cmd){
33             List<String> output = new ArrayList<String>();
34             Process p;
35             try {
36                 p = Runtime.getRuntime().exec(cmd);
37             } catch (IOException e) {
38                 return output;
39             }
40             BufferedReader stdInput = new BufferedReader(new InputStreamReader(p.getInputStream()),8*1024);
41             String s = null;
42             try {
43                 while ((s = stdInput.readLine()) != null)
44                     output.add(s);
45             } catch (IOException e) {
46                 return output;
47             }
48             return output;
49         }
50         public static int getVersion(){
51             List<String> output = new ArrayList<String>();
52             output = executeCommand("reg query \"HKEY\Software\Microsoft\Internet Explorer\" /v svcVersion");
53             if(output.size()>0)
54             {
55                 output = executeCommand("reg query \"HKEY\Software\Microsoft\Internet Explorer\" /v Version");
56             }
57             String internet_explorer_value = output.get(2);
58             String version = internet_explorer_value.trim().split("\\.")[0];
59             version = version.trim().split("\\.")[0];
60             return Integer.parseInt(version);
61         }
62         public static String getSlash(){
63             if(isWindows())
64                 return "\\";
65         }
66     }
```

The screenshot shows the Eclipse IDE interface with the OSHelper.java file open in the editor. The code implements various static methods to determine operating system details and execute system commands.

```
19 public static boolean isWindows(){
20     return getOSName().startsWith("win");
21 }
22
23 public static String getOSBits(){
24     return System.getProperty("os.arch");
25 }
26
27 public static boolean is32(){
28     return getOSBits().equals("x86");
29 }
30
31 public static boolean is64(){
32     if (isWindows())
33     {
34         return (System.getenv("ProgramW6432") != null);
35     }
36     else
37     {
38         return !getOSBits().equals("x86");
39     }
40 }
41
42 private static List<String> executeCommand(String cmd){
43     List<String> output = new ArrayList<String>();
44     Process p;
45     try {
46         p = Runtime.getRuntime().exec(cmd);
47         catch (IOException e) {
48             return output;
49         }
50         BufferedReader stdInput = new BufferedReader(new InputStreamReader(p.getInputStream()),8*1024);
51         String s = null;
52         try {
53             while ((s = stdInput.readLine()) != null)
54                 output.add(s);
55         } catch (IOException e) {
56             return output;
57         }
58     }
59     public static int getIEVersion(){
60         List<String> output;
61         output = executeCommand("reg query \\"HKEY\Software\\Microsoft\\Internet Explorer\" /v svcVersion");
62         if(output.size()>0)
63         {
64             output = executeCommand("reg query \\"HKEY\Software\\Microsoft\\Internet Explorer\" /v Version");
65             String internet_explorer_value = output.get(2);
66             String version = internet_explorer_value.trim().split("\\\\")[-1];
67             version = version.trim().split("\\\\")[-1];
68             return Integer.parseInt(version);
69         }
70
71         public static String getSlash(){
72             if(!isWindows())
73                 return "\\\\";
74             else
75                 return "/";
76     }
77     public static void main(String[] args){
78         System.out.println(getIEVersion());
79     }
80 }
81 }
```

The screenshot shows the Eclipse IDE interface with the StringHelper.java file open in the editor. The code contains utility methods for constructing method signatures and generating random hash codes.

```
1 package com.ebay.maul.helper;
2
3 import java.lang.reflect.Method;
4 import java.math.BigInteger;
5 import java.security.MessageDigest;
6 import java.util.UUID;
7
8 import com.ebay.maul.controller.ContextManager;
9
10 public class StringHelper {
11
12     public static String constructMethodSignature(Method method, Object[] parameters) {
13         return method.getDeclaringClass().getName() + "." + method.getName() + "(" + constructParameterString(parameters) + ")";
14     }
15
16     public static String constructParameterString(Object[] parameters) {
17         StringBuilder sbParam = new StringBuilder();
18
19         if (parameters.length > 0) {
20             for (int i = 0; i < parameters.length; i++) {
21                 if (parameters[i] == null) {
22                     sbParam.append("null,");
23                 } else if (parameters[i] instanceof java.lang.String) {
24                     sbParam.append("\"").append(parameters[i]).append("\",");
25                 } else {
26                     sbParam.append(parameters[i]).append(", ");
27                 }
28             }
29         }
30         if (sbParam.length() > 0)
31             sbParam.delete(sbParam.length() - 2, sbParam.length() - 1);
32
33         return sbParam.toString();
34     }
35
36     public static String getRandomHashCode(String seed) {
37         byte[] data = ContextManager.getThreadContext().getTestMethodSignature() + UUID.randomUUID().getLeastSignificantBits() + seed).getBytes();
38         try {
39             MessageDigest digest = MessageDigest.getInstance("MD5");
40             return new BigInteger(1, digest.digest(data)).toString(16);
41         } catch (Exception e2) {
42             //OK, we won't digest
43             return new String(data);
44         }
45     }
46 }
47 }
```

Screenshot of Eclipse IDE showing the Java EE perspective. The current file is `RaiseClaimTestPlan.java`. The code editor displays the `ThreadHelper` class:

```

1 package com.ebay.maul.helper;
2
3 public class ThreadHelper {
4     /**
5      * Wait For seconds
6      * @param seconds
7      */
8     public static void waitForSeconds(int seconds) {
9         /*
10          * long timeout = 1000 * 60 * 3;
11          * if(ContextManager.getThreadContext()>null) timeout =
12          * ContextManager.getThreadContext().getWebSessionTimeout();
13          * if(seconds > timeout) throw new RuntimeException("Can not wait for " +
14          * seconds + " seconds. Because that is longer than web session timeout of " +
15          * timeout / 1000 +
16          * " seconds which is defined in the testing configuration file.");
17          */
18         try {
19             Thread.sleep(seconds * 1000); // KEEPME
20         } catch (InterruptedException ignore) {
21             ignore.printStackTrace();
22         }
23     }
24 }

```

The Project Explorer shows several packages and their contents. The `ThreadHelper` class is highlighted in the code editor.

Screenshot of Eclipse IDE showing the Java EE perspective. The current file is `URLHelper.java`. The code editor displays the `URLHelper` class:

```

1 package com.ebay.maul.helper;
2
3 public class URLHelper {
4     private static final String HTTPS_PROTOCOL = "https";
5     private static final int HTTPS_PORT = 443;
6
7     public static String convert(String url) {
8         String urlString = ContextManager.getThreadContext()
9             .getUri().convertToClass();
10        if (url != null) {
11            try {
12                Class<?> converterClass = Class.forName(urlConvertClass);
13                Object converter = converterClass.newInstance();
14                Method convertMethod = converterClass.getMethod("convertURL",
15                    String.class);
16                return (String) convertMethod.invoke(converter, url);
17            } catch (Exception e) {
18                logger.warn("Convert URL failed", e);
19            }
20        }
21        return url;
22    }
23
24    public static String getRandomHashcode(String seed) {
25        String signature;
26        if (ContextManager.getThreadContext() != null)
27            signature = ContextManager.getThreadContext()
28                .getTestMethodSignature();
29        else
30            signature = "";
31        byte[] data = (signature + Uuid.randomUUID().getLeastSignificantBits() + seed)
32            .getBytes();
33        try {
34            MessageDigest digest = MessageDigest.getInstance("MD5");
35            return new BigInteger(1, digest.digest(data)).toString(16);
36        } catch (Exception e2) {
37            // OK, we won't digest
38        }
39        return new String(data);
40    }
41
42    public static void main(String[] args) throws Exception {
43        //open("http://besbulki.qa.ebay.com/admin/v3console?ConsumerType=SGFConsumerForBDX&eventTypes=SGF.BBX,JOB_NEW&eventTypes=SGF.BDX,RECURRING,JOB&eventTypes=SGF.BDX,SUBJOB,NEW&reportType=ConsumerHostReport");
44        //String urlString="http://www.wmss.stratus.qa.ebay.com/wmssvc/EBAU/users/1192676159?paymentInstrumentRefId=7409604308&creditCardType=MASTER&paymentInstrumentType=CreditCard&client=EBP&lostFour=0424&expiration";
45    }
46
47    public static String encode(String urlString) throws MalformedURLException, UnsupportedEncodingException {
48        URL url = new URL(urlString);
49        if(url.getQuery() == null) urlString;
50        String[] params = url.getQuery().split("&");
51        String encodedQueryString = "";
52        for(int i=0;i<params.length;i++)
53        {
54            if(params[i].contains("="))
55            {
56                String key = params[i].split("=")[0];
57                String value = params[i].substring(key.length()+1,params[i].length());
58            }
59        }
60    }

```

The Project Explorer shows the `maulcomponent-ebayindia` project with its files. The `URLHelper` class is highlighted in the code editor.

Screenshot of an IDE (likely Eclipse) showing the Java code for the `URLHelper` class. The code handles URL encoding and decoding, opening URLs, and reading their content. It includes annotations like `@param`, `@return`, and `@throws`.

```
184     String value = params[1].substring(key.length()>1,params[1].length());
185     if(value.length() > 0)
186         encodedQueryString = encodedQueryString + URLEncoder.encode(key,"UTF-8")+"="+URLEncoder.encode(value,"UTF-8") +"&";
187     }
188 }
189 if(encodedQueryString.endsWith("&"))
190     encodedQueryString = encodedQueryString.substring(0,encodedQueryString.length()-1);
191 String encodedUrl = urlString.replace(url.getQuery(), encodedQueryString);
192
193 //System.out.println(encodedUrl);
194
195 }
196
197 /**
198 * Open URL using URLConnection
199 *
200 * @param url
201 * @return content of the page
202 * @throws Exception
203 */
204 public static String open(String url) throws Exception {
205     return open(url, false);
206 }
207
208 /**
209 * Open URL, validate : default: false
210 * @param url
211 * @param validate : true - will use webProxyAddress in Context, default:true
212 * @return
213 * @throws Exception
214 */
215 public static String open(String url, boolean validate) throws Exception {
216     return open(url, validate, 30 * 1000,true);
217 }
218
219 /**
220 * Open URL using URLConnection
221 * @param url
222 * @return
223 * @throws Exception
224 */
225 public static String open(String url, boolean validate, boolean useProxy) throws Exception {
226     return open(url, validate, 30 * 1000,useProxy);
227 }
228
229 /**
230 * Open URL using URLConnection
231 * @param url
232 * @param validate : default: false
233 * @param useProxy : true - will use webProxyAddress in Context, default:true
234 * @return
235 * @throws Exception
236 */
237 public static String open(String url, boolean validate, int timeout,boolean useProxy)
238     throws Exception {
239
240     InputStream is = null;
241     BufferedReader br = null;
242     InputStreamReader isr = null;
243
244     try {
245         is = openInputStream(url, validate, timeout,useProxy);
246         isr = new InputStreamReader(is,"UTF-8");
247         br = new BufferedReader(is);
248         StringBuffer sb = new StringBuffer();
249         String line = null;
250         while ((line = br.readLine()) != null) {
251             sb.append(line).append("\n");
252         }
253     } catch (IOException e) {
254         throw new MaulException(e.getMessage(), e);
255     } finally {
256         if (br != null) {
257             try {
258                 br.close();
259             } catch (IOException e) {
260                 e.printStackTrace();
261             }
262         }
263         if (isr != null) {
264             try {
265                 isr.close();
266             } catch (IOException e) {
267                 e.printStackTrace();
268             }
269         }
270     }
271 }
272
273 /**
274 * Open the URL, remember close the input stream after usage
275 * @param url
276 * @return InputStream
277 * @throws Exception
278 */
279 public static InputStream openAsStream(String url) throws Exception {
280     return openAsStream(url, false, 30 * 1000,true);
281 }
282
283 /**
284 * Open the URL, remember close the input stream after usage
285 * @param url
286 * @param convert
287 * @return InputStream
288 * @throws Exception
289 */
290 public static InputStream openAsStream(String url, boolean convert, boolean useProxy)
291     throws Exception {
292     return openAsStream(url, convert, 30 * 1000, useProxy);
293 }
294
295 /**
296 * private static InputStream openAsStream(String url, boolean convert,
297 * @param url
298 * @param convert
299 * @return
300 */
301 private static InputStream openAsStream(String url, boolean convert,
302     int timeout,boolean useProxy) throws Exception {
303     URLIdentifier url;
304     if (convert)
305         convert(url);
306
307     if (url.startsWith("HTTPS_PROTOCOL"))
308         return openHttpsURLConnection(url,useProxy);
309
310     logger.info("opening URL: " + url);
311     InputStream is = null;
312     URLConnection connection = null;
313 }
```

Screenshot of an IDE (likely Eclipse) showing the Java code for the `URLHelper` class. This version of the code includes additional methods for opening HTTPS URLs and handling SSL certificates.

```
169
170     return sb.toString();
171 } catch (Throwable e) {
172     throw new MaulException(e.getMessage(), e);
173 } finally {
174
175     if (br != null) {
176         try {
177             br.close();
178         } catch (IOException e) {
179             e.printStackTrace();
180         }
181     }
182     if (isr != null) {
183         try {
184             isr.close();
185         } catch (IOException e) {
186             e.printStackTrace();
187         }
188     }
189     if (is != null) {
190         try {
191             is.close();
192         } catch (IOException e) {
193             e.printStackTrace();
194         }
195     }
196 }
197
198 /**
199 * Open the URL, remember close the input stream after usage
200 * @param url
201 * @param convert
202 * @return InputStream
203 * @throws Exception
204 */
205 public static InputStream openAsStream(String url) throws Exception {
206     return openAsStream(url, false, 30 * 1000,true);
207 }
208
209 /**
210 * Open the URL, remember close the input stream after usage
211 * @param url
212 * @param convert
213 * @param convert : false - won't convert the url with urlconverter
214 * @return InputStream
215 * @throws Exception
216 */
217 public static InputStream openAsStream(String url, boolean convert, boolean useProxy)
218     throws Exception {
219     return openAsStream(url, convert, 30 * 1000, useProxy);
220 }
221
222 /**
223 * private static InputStream openAsStream(String url, boolean convert,
224 * @param url
225 * @param convert
226 * @return
227 */
228 private static InputStream openAsStream(String url, boolean convert,
229     int timeout,boolean useProxy) throws Exception {
230     URLIdentifier url;
231     if (convert)
232         convert(url);
233
234     if (url.startsWith("HTTPS_PROTOCOL"))
235         return openHttpsURLConnection(url,useProxy);
236
237     logger.info("opening URL: " + url);
238     InputStream is = null;
239     URLConnection connection = null;
240 }
```

The screenshot shows the Java code for URLHelper.java in an IDE. The code handles URL connections, including proxy handling and SSL configuration. It uses Context, URL, and HttpURLConnection classes, along with various exception handling blocks. The code is annotated with suppression warnings for deprecated methods.

```
232     URLConnection connection = null;
233     try {
234         Context context = ContextManager.getThreadContext();
235         if (useProxy && context.getWebProxyAddress() != null) {
236             String host = context.getWebProxyAddress().split(":")[0];
237             int port = Integer.parseInt(context.getWebProxyAddress().split(":")[1]);
238             Proxy proxy = new Proxy(Proxy.Type.HTTP, new InetSocketAddress(host, port));
239             connection = new URL(url).openConnection(proxy);
240         } else {
241             connection = new URL(url).openConnection();
242         }
243         connection.setConnectTimeout(timeout);
244         connection.setReadTimeout(timeout);
245         is = connection.getInputStream();
246         return is;
247     } catch (Throwable e) {
248         e.printStackTrace();
249         throw new MauiException(e.getMessage(), e);
250     }
251 }
252
253 /**
254  * @suppressWarnings("deprecation")
255  */
256 public static String openHttpsUrl(String url) throws HttpException,
257     IOException {
258     URL needUrl = new URL(url);
259     Protocol myhttps = new Protocol("https",
260         new EasySSLProtocolSocketFactory(), needUrl.getPort());
261     Protocol.registerProtocol("https", myhttps);
262
263     HttpClient httpclient = new HttpClient();
264     String response = null;
265     GetMethod httpget = new GetMethod(url);
266     try {
267         // httpclient.getPostConfiguration().setHost(needUrl.getHost(),
268         // needUrl.getPort(), newProtocol("https", needUrl));
269         // EasySSLProtocolSocketFactory(needUrl.getPort());
270         httpclient.executeMethod(httpget);
271         response = httpget.getResponseBodyAsString();
272     } finally {
273         httpget.releaseConnection();
274     }
275     return response;
276 }
277
278 /**
279  * @suppressWarnings("deprecation")
280  */
281 public static InputStream openHttpsAsStream(String url, boolean useProxy)
282     throws HttpException, IOException {
283     URL needUrl = new URL(url);
284     Protocol myhttps = new Protocol("https",
285         new EasySSLProtocolSocketFactory(), needUrl.getDefaultPort());
286     Protocol.registerProtocol("https", myhttps);
287
288     HttpClient httpclient = new HttpClient();
289     Context context = ContextManager.getThreadContext();
290     if (useProxy && context.getWebProxyAddress() != null) {
291         String host = context.getWebProxyAddress().split(":")[0];
292         int port = Integer.parseInt(context.getWebProxyAddress().split(":")[1]);
293         httpclient.getHostConfiguration().setProxy(host, port);
294     }
295     InputStream response = null;
296     GetMethod httpget = new GetMethod(url);
297     httpclient.executeMethod(httpget);
298     response = httpget.getResponseBodyAsStream();
299 }
300 }
```

The screenshot shows the same Java code for URLHelper.java in the IDE, but with some additional code at the bottom. This additional code is identical to the code shown in the first screenshot, starting from line 232.

```
232     URLConnection connection = null;
233     try {
234         Context context = ContextManager.getThreadContext();
235         if (useProxy && context.getWebProxyAddress() != null) {
236             String host = context.getWebProxyAddress().split(":")[0];
237             int port = Integer.parseInt(context.getWebProxyAddress().split(":")[1]);
238             Proxy proxy = new Proxy(Proxy.Type.HTTP, new InetSocketAddress(host, port));
239             connection = new URL(url).openConnection(proxy);
240         } else {
241             connection = new URL(url).openConnection();
242         }
243         connection.setConnectTimeout(timeout);
244         connection.setReadTimeout(timeout);
245         is = connection.getInputStream();
246         return is;
247     } catch (Throwable e) {
248         e.printStackTrace();
249         throw new MauiException(e.getMessage(), e);
250     }
251 }
252
253 /**
254  * @suppressWarnings("deprecation")
255  */
256 public static String openHttpsUrl(String url) throws HttpException,
257     IOException {
258     URL needUrl = new URL(url);
259     Protocol myhttps = new Protocol("https",
260         new EasySSLProtocolSocketFactory(), needUrl.getPort());
261     Protocol.registerProtocol("https", myhttps);
262
263     HttpClient httpclient = new HttpClient();
264     String response = null;
265     GetMethod httpget = new GetMethod(url);
266     try {
267         // httpclient.getPostConfiguration().setHost(needUrl.getHost(),
268         // needUrl.getPort(), newProtocol("https", needUrl));
269         // EasySSLProtocolSocketFactory(needUrl.getPort());
270         httpclient.executeMethod(httpget);
271         response = httpget.getResponseBodyAsString();
272     } finally {
273         httpget.releaseConnection();
274     }
275     return response;
276 }
277
278 /**
279  * @suppressWarnings("deprecation")
280  */
281 public static InputStream openHttpsAsStream(String url, boolean useProxy)
282     throws HttpException, IOException {
283     URL needUrl = new URL(url);
284     Protocol myhttps = new Protocol("https",
285         new EasySSLProtocolSocketFactory(), needUrl.getDefaultPort());
286     Protocol.registerProtocol("https", myhttps);
287
288     HttpClient httpclient = new HttpClient();
289     Context context = ContextManager.getThreadContext();
290     if (useProxy && context.getWebProxyAddress() != null) {
291         String host = context.getWebProxyAddress().split(":")[0];
292         int port = Integer.parseInt(context.getWebProxyAddress().split(":")[1]);
293         httpclient.getHostConfiguration().setProxy(host, port);
294     }
295     InputStream response = null;
296     GetMethod httpget = new GetMethod(url);
297     httpclient.executeMethod(httpget);
298     response = httpget.getResponseBodyAsStream();
299 }
300 }
```



The screenshot shows the Eclipse IDE interface with the Java EE perspective selected. The top bar includes 'Quick Access' and 'Java EE' buttons. The left sidebar shows project navigation with 'RaiseClaimTest.java' as the current file. The main editor area displays the Java code for the 'WaitHelper' class. The code includes imports for 'com.mau.component.ebayindia', 'com.ebay.mau.helper', and 'com.openqa.selenium'. It contains several static methods: 'presenceOfElementLocated', 'waitForElementToBeVisible', 'waitForElementAndReturnElement', and 'waitForElementToBeVisible'. Each method takes a WebDriver and a locator as parameters. The 'presenceOfElementLocated' method returns a boolean indicating if the element is present. The 'waitForElementToBeVisible' methods use a WebDriverWait with a timeout of 30 seconds. The 'waitForElementAndReturnElement' method returns the found element. The 'waitForElementToBeVisible' methods catch a RuntimeException and rethrow it with a descriptive message including the locator and current URL.

```
public class WaitHelper {  
    static long TIMEOUT_S = 60;  
    static int INT_TIMEOUT_S = Integer.parseInt(toString(TIMEOUT_S));  
    static long WAIT_TIMEOUT = 30L;  
    protected static final Logger logger = Logging.getLogger(HtmlElement.class);  
  
    protected static Function<WebDriver, WebElement> presenceOfElementLocated(  
        final By locator) {  
        return new Function<WebDriver, WebElement>() {  
            public WebElement apply(WebDriver driver) {  
                return driver.findElement(locator);  
            }  
        };  
    }  
  
    /**  
     * Waits for the element to be visible until a timeout of 30 secs.  
     * @param driver  
     * @param locator  
     */  
    public static void waitForElementToBeVisible(final WebDriver driver,  
        final By locator) throws RuntimeException {  
        Wait<WebDriver> wait = new WebDriverWait(driver, WAIT_TIMEOUT);  
        try {  
            wait.until(new ExpectedCondition<WebElement>() {  
                public WebElement apply(WebDriver driver) {  
                    // driver.switchTo().defaultContent();  
                    WebElement element = driver.findElement(locator);  
                    if (element.isDisplayed()) {  
                        return element;  
                    }  
                    return null;  
                }  
            });  
        } catch (Exception e) {  
            throw new RuntimeException("Exception while waiting for " + locator  
                + ". Exception:" + e + " on " + driver.getCurrentUrl());  
        }  
    }  
  
    public static WebElement waitForElementAndReturnElement(  
        final WebDriver driver, final By locator) throws RuntimeException {  
        waitForElementToBeVisible(driver, locator);  
        return driver.findElement(locator);  
    }  
  
    /**  
     * Waits for the element to be visible until the specified timeout.  
     * @param driver  
     * @param locator  
     * @param timeout  
     */  
    public static void waitForElementToBeVisible(final WebDriver driver,  
        final By locator, long timeout) throws RuntimeException {  
        Wait<WebDriver> wait = new WebDriverWait(driver, timeout);  
        try {  
            wait.until(new ExpectedCondition<WebElement>() {  
                public WebElement apply(WebDriver driver) {  
                    return null;  
                }  
            });  
        } catch (Exception e) {  
            throw new RuntimeException("Exception while waiting for " + locator  
                + ". Exception:" + e + " on " + driver.getCurrentUrl());  
        }  
    }  
}
```

The screenshot shows the Eclipse IDE interface with several tabs open at the top: 'RaiseClaimTestPlan.java', 'WaitHelper.class', 'ThreadHelper.class', and 'URLHelper.class'. The main editor area displays Java code for a class named 'WaitHelper'. The code includes methods for waiting for elements to appear or disappear from the page. The code uses annotations like '@Test' and '@BeforeClass' to mark test methods and setup logic. It also imports various Selenium and Java libraries.

```
public void apply(WebDriver driver) {
    // driver.switchTo().defaultContent();
    WebElement element = driver.findElement(locator);
    if (element.isDisplayed()) {
        return element;
    }
    return null;
}
} catch (Exception e) {
    throw new RuntimeException("Exception while waiting for " + locator
        + ". Exception:" + e);
}
}

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

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

```

The screenshot shows an IDE interface with the following details:

- Title Bar:** Quick Access, Java EE, Debug.
- Project Explorer:** Shows `RaiseClaimTestPlan.java`, `WaitHelper.class`, `ThreadHelper.class`, and `URLHelper.class`.
- Code Editor:** Displays the `WaitHelper.class` file with approximately 215 lines of Java code. The code implements several static methods for waiting on web driver conditions, such as `waitForTitleStartingWithString`, `waitForTitleContainingString`, `waitForCurrentUrlToContainString`, and `waitForCurrentUrltoMatchString`. It uses the `ExpectedCondition` interface and `WebDriverWait` to handle timeouts and exceptions.
- Status Bar:** Read-Only, Smart Insert, 224 : 57.

The screenshot shows an IDE interface with the following details:

- Title Bar:** Quick Access, Java EE, Debug.
- Project Explorer:** Shows `RaiseClaimTestPlan.java`, `WaitHelper.class`, `ThreadHelper.class`, and `URLHelper.class`.
- Code Editor:** Displays the `WaitHelper.class` file with approximately 246 lines of Java code. This version includes additional methods and logic compared to the first screenshot, particularly around URL matching and explicit waits. It also includes a `holdUntil` method and a `waitForSeconds` method.
- Status Bar:** Read-Only, Smart Insert, 224 : 57.

Screenshot of an IDE (likely Eclipse) showing the Java code for `XMLHelper`.

Project Explorer:

- com.ebay.mau.helper
 - IEDriverFactory.class
 - IWebDriverFactory.class
 - OpenCapabilitiesFactory.class
 - OpenDriverFactory.class
 - PhantomJSCapabilitiesFactory.class
 - RemoteDriverFactory.class
 - SafarCapabilitiesFactory.class
 - SafarDriverFactory.class
 - com.ebay.mau.exception
 - APIException.class
 - DataCreationException.class
 - DBException.class
 - DBFinderException.class
 - EmailException.class
 - MauException.class
 - PageNotCurrentException.class
 - SSHException.class
 - WebResponseException.class
 - WebSessionTerminatedException.class
 - XMLException.class
 - CALHelper.class
 - ClassContextHelper.class
 - EscapeHelper.class
 - FileHelper.class
 - OSHelper.class
 - StringHelper.class
 - ThreadHelper.class
 - URLHelper.class
 - WaitHelper.class
 - XMLHelper.class
- com.ebay.mau.reporter
 - DetailedLog.class
 - HTMLReporter.class
 - MiniTestResult.class
 - PluginsUtil.class
 - XMLReporter.class
- com.ebay.mau.reporter.pluginmodel
 - Mauplugins.class
 - Method.class
 - ObjectFactory.class
 - Page.class
 - Plugin.class
 - Test.class
 - reporter.cs

Code Editor:

```

package com.ebay.mau.helper;

import java.io.File;
import java.io.StringWriter;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
public class XMLHelper {
    public static NodeList getXMLNodes(String xmlFileName, String tagName) {
        NodeList nlist = null;
        try {
            File xmlFile = new File(xmlFileName);
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);
            doc.getDocumentElement().normalize();
            nlist = doc.getElementsByTagName(tagName);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return nlist;
    }
    public static <T> String getXml(T response) {
        try {
            JAXBContext jaxbContext = JAXBContext.newInstance(response.getClass());
            Marshaller jaxb = jaxbContext.createMarshaller();
            // output pretty printed
            jaxb.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
            StringWriter writer = new StringWriter();
            jaxb.marshal(response, writer);
            return writer.toString();
        } catch (JAXBException e) {
            e.printStackTrace();
        }
        return "";
    }
}

```

Call Hierarchy:

Members calling constructors of "WebPageSection" - in workspace

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

REPORTER:

The screenshot shows the Java code for the XMLReporter class. The code is annotated with numerous red question marks, indicating potential issues or areas of concern. The code itself is a detailed log implementation.

```
1 package com.ebay.maui.reporter;
2
3 public class DetailedLog {
4     private String type;
5     private String msg;
6     private String screen;
7     private String src;
8     private String location;
9     private String href;
10    private String root;
11
12    public DetailedLog() {
13    }
14
15    public DetailedLog(String s, String root) {
16        this.root = root;
17        if (s == null) {
18            return;
19        }
20        String[] parts = s.split("\\\\\\\\");
21        for (int i = 0; i < parts.length; i++) {
22            parse(parts[i]);
23        }
24    }
25
26    public String getHref() {
27        return href;
28    }
29
30    public String getLocation() {
31        return location;
32    }
33
34    public String getMsg() {
35        return msg;
36    }
37
38    public String getRoot() {
39        return root;
40    }
41
42    public String getScreen() {
43        return screen;
44    }
45
46    public String getScreenURL() {
47        if (screen != null) {
48            return screen.replaceAll("\\\\\\\\", "/");
49        }
50        return "no screen";
51    }
52
53    public String getSource() {
54        return src;
55    }
56
57    public String getType() {
58        return type;
59    }
60
61    private void parse(String part) {
62        if (part.startsWith("TYPE=")) {
63            type = part.replace("TYPE=", "");
64        }
65    }
66
67    private void parse(String part) {
68        if (part.startsWith("MSG=")) {
69            msg = part.replace("MSG=", "");
70        }
71        else if (part.startsWith("SCREEN=")) {
72            screen = part.replace("SCREEN=", "");
73        }
74        else if (part.startsWith("SRC=")) {
75            src = part.replace("SRC=", "");
76        }
77        else if (part.startsWith("LOCATION=")) {
78            location = part.replace("LOCATION=", "");
79        }
80        else if (part.startsWith("HREF=")) {
81            href = part.replace("HREF=", "");
82        }
83    }
84
85    public void setHref(String href) {
86        this.href = href;
87    }
88
89    public void setLocation(String location) {
90        this.location = location;
91    }
92
93    public void setMsg(String msg) {
94        this.msg = msg;
95    }
96
97    public void setScreen(String screen) {
98        this.screen = screen;
99    }
100
101    public void setSrc(String src) {
102        this.src = src;
103    }
104
105    public void setType(String type) {
106        this.type = type;
107
108    }
109
110    public String toString() {
111        StringBuffer buff = new StringBuffer();
112        buff.append("<LOG>");
113        if (type != null) {
114            buff.append(type);
115        }
116        buff.append("<MSG>");
117        if (msg != null) {
118            buff.append(msg);
119        }
120        buff.append("<SCREEN>");
121        if (screen != null) {
122            buff.append(screen);
123        }
124        buff.append("<SRC>");
125        if (src != null) {
126            buff.append(src);
127        }
128        buff.append("<LOCATION>");
129        if (location != null) {
130            buff.append(location);
131        }
132        buff.append("<HREF>");
133    }
134}
```

The screenshot shows the Java code for the XMLReporter class. The code is annotated with numerous red question marks, indicating potential issues or areas of concern. The code itself is a detailed log implementation.

```
1 package com.ebay.maui.reporter;
2
3 public class DetailedLog {
4     private String type;
5     private String msg;
6     private String screen;
7     private String src;
8     private String location;
9     private String href;
10    private String root;
11
12    public DetailedLog() {
13    }
14
15    public DetailedLog(String s, String root) {
16        this.root = root;
17        if (s == null) {
18            return;
19        }
20        String[] parts = s.split("\\\\\\\\");
21        for (int i = 0; i < parts.length; i++) {
22            parse(parts[i]);
23        }
24    }
25
26    public String getHref() {
27        return href;
28    }
29
30    public String getLocation() {
31        return location;
32    }
33
34    public String getMsg() {
35        return msg;
36    }
37
38    public String getRoot() {
39        return root;
40    }
41
42    public String getScreen() {
43        return screen;
44    }
45
46    public String getScreenURL() {
47        if (screen != null) {
48            return screen.replaceAll("\\\\\\\\", "/");
49        }
50        return "no screen";
51    }
52
53    public String getSource() {
54        return src;
55    }
56
57    public String getType() {
58        return type;
59    }
60
61    private void parse(String part) {
62        if (part.startsWith("TYPE=")) {
63            type = part.replace("TYPE=", "");
64        }
65    }
66
67    private void parse(String part) {
68        if (part.startsWith("MSG=")) {
69            msg = part.replace("MSG=", "");
70        }
71        else if (part.startsWith("SCREEN=")) {
72            screen = part.replace("SCREEN=", "");
73        }
74        else if (part.startsWith("SRC=")) {
75            src = part.replace("SRC=", "");
76        }
77        else if (part.startsWith("LOCATION=")) {
78            location = part.replace("LOCATION=", "");
79        }
80        else if (part.startsWith("HREF=")) {
81            href = part.replace("HREF=", "");
82        }
83        else {
84            msg = part;
85        }
86    }
87
88    public void setHref(String href) {
89        this.href = href;
90    }
91
92    public void setLocation(String location) {
93        this.location = location;
94    }
95
96    public void setMsg(String msg) {
97        this.msg = msg;
98    }
99
100   public void setScreen(String screen) {
101        this.screen = screen;
102    }
103
104   public void setSrc(String src) {
105        this.src = src;
106    }
107
108   public void setType(String type) {
109        this.type = type;
110
111    }
112
113    public String toString() {
114        StringBuffer buff = new StringBuffer();
115        buff.append("<LOG>");
116        if (type != null) {
117            buff.append(type);
118        }
119        buff.append("<MSG>");
120        if (msg != null) {
121            buff.append(msg);
122        }
123        buff.append("<SCREEN>");
124        if (screen != null) {
125            buff.append(screen);
126        }
127        buff.append("<SRC>");
128        if (src != null) {
129            buff.append(src);
130        }
131        buff.append("<LOCATION>");
132        if (location != null) {
133            buff.append(location);
134        }
135        buff.append("<HREF>");
136    }
137}
```

The screenshot shows the Java code for the XMLReporter class. The code is responsible for generating a string representation of test results based on various parameters like href, location, msg, screen, type, and src. It uses a StringBuffer to build the final string by appending different parts based on their presence and type.

```
71     src = part.replace("SRC=", "");
72     } else if (part.startsWith("LOCATION=")) {
73         location = part.substring("LOCATION=".length());
74     } else if (part.startsWith("HREF=")) {
75         href = part.replace("HREF=", "");
76     } else {
77         msg = part;
78     }
79 }
80
81 public void setHref(String href) {
82     this.href = href;
83 }
84
85 public void setLocation(String location) {
86     this.location = location;
87 }
88
89 public void setMsg(String msg) {
90     this.msg = msg;
91 }
92
93 public void setScreen(String screen) {
94     this.screen = screen;
95 }
96
97 public void setSrc(String src) {
98     this.src = src;
99 }
100
101 public void setType(String type) {
102     this.type = type;
103 }
104
105
106 public String testing() {
107     StringBuffer buff = new StringBuffer();
108     buff.append("<TYPE>");
109     if (type != null) {
110         buff.append(type);
111     }
112     buff.append("<MSG>");
113     if (msg != null) {
114         buff.append(msg);
115     }
116     buff.append("<SCREEN>");
117     if (screen != null) {
118         buff.append(screen);
119     }
120     buff.append("<SRC>");
121     if (src != null) {
122         buff.append(src);
123     }
124     buff.append("<LOCATION>");
125     if (location != null) {
126         buff.append(location);
127     }
128     buff.append("<HREF>");
129     if (href != null) {
130         buff.append(href);
131     }
132 }
133
134 }
135 }
```

The screenshot shows the Java code for the MiniTestResult class. This class represents a single test result with properties for name, id, total methods, instances passed, failed, and skipped. It provides methods to get and set these values, as well as to calculate the percentage of tests passed.

```
3 public class MiniTestResult {
4
5     private String name;
6     private String id;
7
8     private int totalMethod;
9
10    private int instancesPassed;
11
12    private int instancesFailed;
13
14    private int instancesSkipped;
15
16    public MiniTestResult(String name) {
17        this.name = name;
18        this.id = name.toLowerCase().replaceAll(" ", "_");
19    }
20
21    public String getId() {
22        return id;
23    }
24
25    public int getInstancesFailed() {
26        return instancesFailed;
27    }
28
29    public int getInstancesPassed() {
30        return instancesPassed;
31    }
32
33    public int getInstancesSkipped() {
34        return instancesSkipped;
35    }
36
37    public String getName() {
38        return name;
39    }
40
41    public int getTotalMethod() {
42        return totalMethod;
43    }
44
45
46    public void setId(String id) {
47        this.id = id;
48    }
49
50    public void setInstancesFailed(int instancesFailed) {
51        this.instancesFailed = instancesFailed;
52    }
53
54    public void setInstancesPassed(int instancesPassed) {
55        this.instancesPassed = instancesPassed;
56    }
57
58    public void setInstancesSkipped(int instancesSkipped) {
59        this.instancesSkipped = instancesSkipped;
60    }
61
62    public void setName(String name) {
63        this.name = name;
64    }
65
66    public void setTotalMethod(int totalMethod) {
67        this.totalMethod = totalMethod;
68    }
69 }
```

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

- Project Bar:** Shows the current project is "RaiseClaimTestPlan" and the active file is "RaiseClaimTestPlan.java".
- Toolbars:** Standard Eclipse toolbars for file operations, search, and navigation.
- Left Margin:** Shows the line numbers from 66 to 132.
- Code Editor:** Displays the Java code for the `RaiseClaimTestPlan.java` file. The code includes imports for `java.util`, `java.io`, and `com.cognizant.tps`. It defines a class `HTMLReporter` that implements `IReporter` and `IInvokedMethodListener`. The class contains several protected inner classes: `TestMethodSorter`, `TestResultSorter`, and `StringHelper`. The `TestMethodSorter` class uses a comparator to sort methods by class name and method name. The `TestResultSorter` class uses a comparator to sort results by class name and method name. The `StringHelper` class provides utility methods for escaping strings. The `main` method reads the class name from the command line and writes the resource to a file.
- Right Margin:** Shows the status bar with "Read-Only", "Smart Insert", and "1:1".

A screenshot of an IDE interface showing a Java code editor. The code is part of a class named `HTMLReporter`. The code handles failure messages, merges them, and then copies resources from a specified directory into the output directory. It uses `File.separator` to separate directory and file paths. The code includes imports for `java.util.List`, `java.util.ArrayList`, `java.io.File`, and `java.io.IOException`.

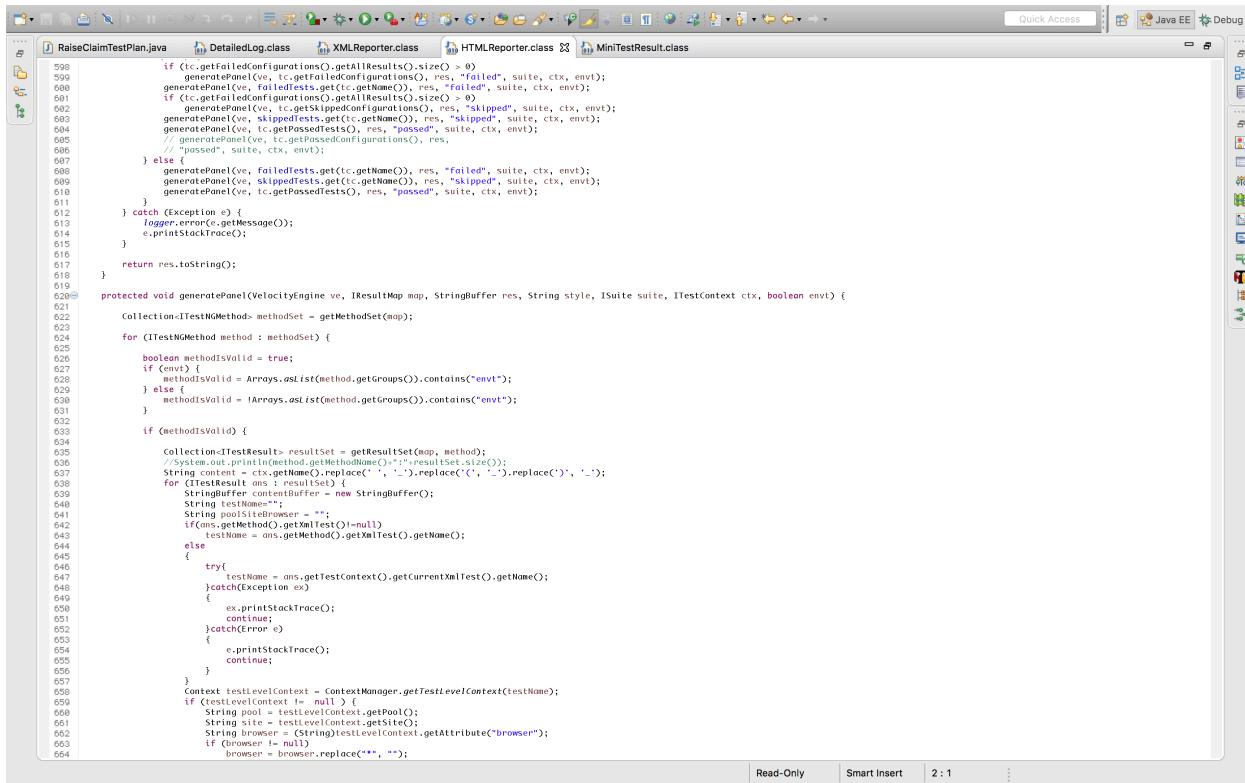
```
199     for (int i = 0; i < size - 1; i++) {
200         FailureMessage.append("Failure ").append(i).append(" of ").append(size).append("\n");
201         Throwable t = verificationFailures.get(i);
202         String fullStackTrace = Utils.stackTrace(t, false)[1];
203         FailureMessage.append(fullStackTrace).append("\n");
204     }
205
206     //Final failure
207     Throwable last = verificationFailures.get(size - 1);
208     FailureMessage.append("Failure ").append(size).append(" of ").append(size).append("\n");
209     FailureMessage.append(last.toString());
210
211     //set merged throwable
212     Throwable merged = new Throwable(FailureMessage.toString());
213     merged.setStackTrace(last.getStackTrace());
214
215     result.setThrowable(merged);
216 }
217 }
218
219 public void beforeInvocation(IInvokedMethod arg0, ITestResult arg1) {
220 }
221
222 protected void copyResources() throws Exception {
223     new File(outputDirectory + File.separator + "resources").mkdir();
224     new File(outputDirectory + File.separator + "resources" + File.separator + "css").mkdir();
225     new File(outputDirectory + File.separator + "resources" + File.separator + "images").mkdir();
226     new File(outputDirectory + File.separator + "resources" + File.separator + "images" + File.separator + "lightbox").mkdir();
227     new File(outputDirectory + File.separator + "resources" + File.separator + "images" + File.separator + "images" + File.separator + "mktree").mkdir();
228     new File(outputDirectory + File.separator + "resources" + File.separator + "images" + File.separator + "yukontoolbox").mkdir();
229     new File(outputDirectory + File.separator + "resources" + File.separator + "js").mkdir();
230
231     List<String> list = new ArrayList<String>();
232     resources.add("reporter" + File.separator + "reporter.css");
233     resources.add("reporter" + File.separator + "css" + File.separator + "jquery.lightbox-0.5.css");
234     resources.add("reporter" + File.separator + "css" + File.separator + "mktree.css");
235     resources.add("reporter" + File.separator + "images" + File.separator + "lightbox" + File.separator + "lightbox-blank.gif");
236     resources.add("reporter" + File.separator + "images" + File.separator + "lightbox" + File.separator + "lightbox-btn-close.gif");
237     resources.add("reporter" + File.separator + "images" + File.separator + "lightbox" + File.separator + "lightbox-btn-next.gif");
238     resources.add("reporter" + File.separator + "images" + File.separator + "lightbox" + File.separator + "lightbox-btn-prev.gif");
239     resources.add("reporter" + File.separator + "images" + File.separator + "lightbox" + File.separator + "lightbox-ico-loading.gif");
240     resources.add("reporter" + File.separator + "images" + File.separator + "mktree" + File.separator + "minus.gif");
241     resources.add("reporter" + File.separator + "images" + File.separator + "mktree" + File.separator + "plus.gif");
242     resources.add("reporter" + File.separator + "images" + File.separator + "mktree" + File.separator + "text1.gif");
243     resources.add("reporter" + File.separator + "images" + File.separator + "mktree" + File.separator + "text2.gif");
244     resources.add("reporter" + File.separator + "images" + File.separator + "mktree" + File.separator + "text3.gif");
245     resources.add("reporter" + File.separator + "images" + File.separator + "yukontoolbox" + File.separator + "yukontoolbox-goldprod.gif");
246     resources.add("reporter" + File.separator + "images" + File.separator + "yukontoolbox" + File.separator + "yukontoolbox-mid.gif");
247     resources.add("reporter" + File.separator + "images" + File.separator + "yukontoolbox" + File.separator + "yukontoolbox" + File.separator + "grey-bl.gif");
248     resources.add("reporter" + File.separator + "images" + File.separator + "yukontoolbox" + File.separator + "yukontoolbox" + File.separator + "yellow-tl.gif");
249     resources.add("reporter" + File.separator + "images" + File.separator + "yukontoolbox" + File.separator + "yukontoolbox" + File.separator + "yellow-tr.gif");
250     resources.add("reporter" + File.separator + "js" + File.separator + "jquery-1.3.min.js");
251     resources.add("reporter" + File.separator + "js" + File.separator + "jquery.lightbox-0.5.min.js");
252     resources.add("reporter" + File.separator + "js" + File.separator + "mktree.js");
253     resources.add("reporter" + File.separator + "js" + File.separator + "report.js");
254
255     writeResourceToFile(resources, resourcename, HTMLReporter.class);
256 }
257 }
```

A screenshot of an IDE interface showing a Java code editor. The code is part of a class named `HTMLReporter`. It handles resource copying, creates a writer for generating HTML reports, and executes command strings for different operating systems. It also includes a method for generating an error report. The code includes imports for `java.util.List`, `java.util.ArrayList`, `java.io.File`, `java.io.IOException`, and `java.lang.Runtime`.

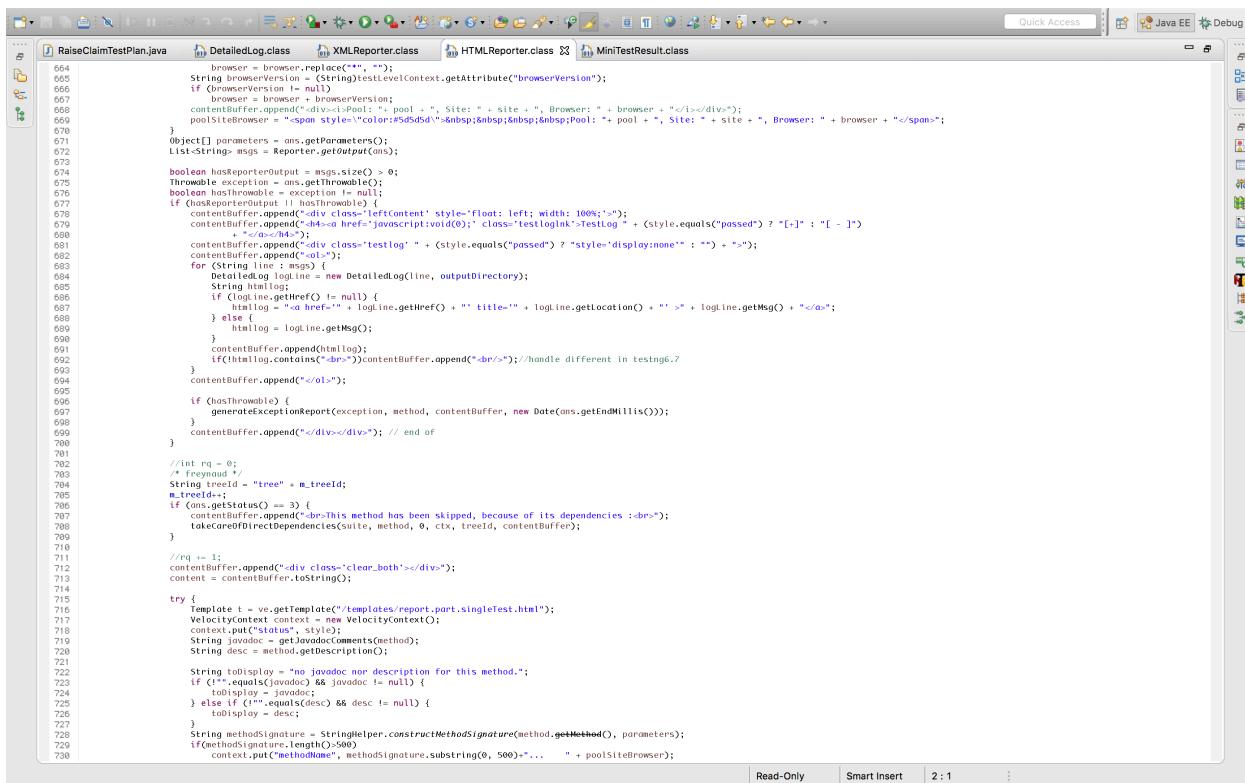
```
268     for (String resourcename : resources) {
269         File f = new File(outputDirectory, resourcename.replace("reporter", "resources"));
270         resourcename = resourcename.replaceAll("\\\\", "/");
271         logger.debug("about to write resource " + resourcename + " to the file " + f.getAbsolutePath());
272         writeResourceToFile(f, resourcename, HTMLReporter.class);
273     }
274
275     protected PrintWriter createWriter(String outdir) throws IOException {
276         System.setProperty("file.encoding", "UTF-8");
277         File f = new File(outdir, "mouireport" + uid + ".html";
278         // ConfigLoader.getInstance().setProperty("report",
279         // f.getAbsolutePath());
280         logger.info("generating report " + f.getAbsolutePath());
281
282         report = f;
283         PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter(f)));
284         // handle quoted code issue
285         OutputStream out = new FileOutputStream();
286         Writer writer = new BufferedWriter(new OutputStreamWriter(out, "utf-8"));
287         return new PrintWriter(writer);
288     }
289
290     /** Finishes HTML stream */
291     protected void endhtml(PrintWriter out) {
292         out.println("</body></html>");
293     }
294
295     public void executeCmdString(browserPath, String theurl) {
296         String cmdline = null;
297         String osname = System.getProperty("os.name");
298
299         if (osname.startsWith("Windows")) {
300             cmdline = "start " + theurl;
301             // you need to start cmd.exe because start is not
302             // an external command but internal, you need to start the
303             // command interpreter
304             // cmdline = "cmd.exe < " + cmdline;
305             cmdline = "runDLL C:\Windows\ShellExec_RunDLL " + browserPath + " " + theUrl;
306         } else if (osname.startsWith("Mac")) {
307             cmdline = "open " + theurl;
308         } else {
309             // Linux
310             cmdline = "open " + browserPath + " " + theUrl;
311         }
312         try {
313             Runtime.getRuntime().exec(cmdline);
314         } catch (Exception e) {
315             logger.info(e);
316         }
317     }
318
319 }
320
321 protected void generateCalErrorReport(String exception, StringBuffer contentBuffer) {
322     contentBuffer.append(<div class="stContainer"><a href="javascript:void(0); class="exceptionLink">Detail</a></div>);
323     contentBuffer.append(<div class="exception" style="display:none">);
324     contentBuffer.append(exception);
325     contentBuffer.append(</div></div>);
326 }
327
328 /**
329 * protected String generateAllErrorHTML(ITestContext tc, ISuite suite,
330 * StringBuffer calcount) {
331 * }
```

The screenshot shows the Eclipse IDE interface with several tabs open. The active tab is 'RaisClaimTestPlan.java'. The code in the editor is as follows:

```
532     StringBuffer contentBuffer = new StringBuffer();
533     contentBuffer.append("<table class='ex' width='99%>"); //thead><tr><th>TestMethod</th><th>Errors</th></tr></thead><tbody>");
534     Map<String, List<String>> errorMap = pageEntry.getValue();
535
536     boolean found = false;
537     for (ITestResult testResult : testResults) {
538         Method method = testResult.getMethod();
539         String methodSignature = StringHelper.constructMethodSignature(method, testResult.getParameters());
540         if (errorMap.containsKey(methodSignature)) {
541             found = true;
542             contentBuffer.append("<tr>"); //td>" + methodSignature + "</td><td>");
543             for (String message : errorMap.get(methodSignature)) {
544                 contentBuffer.append("<br>"); //td>message + "<br>"); //td>
545                 contentBuffer.append("<br>"); //td>
546             }
547             contentBuffer.append("</td><tr>"); //tbody>
548         }
549     }
550
551     if (found) {
552         contentBuffer.append("</tbody></table>");
553     }
554
555     try {
556         Template t = ve.getTemplate("templates/report_part.singlePageError.html");
557         VelocityContext context = new VelocityContext();
558         context.put("status", style);
559         context.put("pagefactory", pagefactory.getKey());
560         context.put("content", content);
561         StringWriter writer = new StringWriter();
562         t.merge(context, writer);
563         res.append(writer.toString());
564     } catch (Exception e) {
565         logger.error("Error creating a singlePageError." + e.getMessage());
566     }
567     pageCount++;
568 }
569 }
570 sbColCount.append(pageCount);
571
572 public void generateGroupsAndCollections(Collection<ITestNGMethod> methods) {
573     Set<String> allGroups = new HashSet<String>();
574     for (ITestNGMethod method : methods) {
575         for (int j = 0; j < method.getGroups().length; j++) {
576             allGroups.add(method.getGroups()[j]);
577         }
578     }
579     m_out.print("<p>Tags :<br/>");
580     for (String group : allGroups) {
581         m_out.print("<input type='checkbox' value=" + group.replace(' ', '_').replace('(', '_').replace(')', '_') + " checked='checked'> " + group
582             + "<br/>");
```



```
598     if (tc.getFailedConfigurations().getResults().size() > 0)
599         generatePanel(vc, tc.getFailedConfigurations(), res, "failed", suite, ctx, envt);
600     generatePanel(vc, failedTests.get(tc.getName()), res, "failed", suite, ctx, envt);
601     if (!tc.getFailedConfigurations().getResults().size() > 0)
602         generatePanel(vc, tc.getSkippedConfigurations(), res, "skipped", suite, ctx, envt);
603     generatePanel(vc, tc.getSkippedTests().get(tc.getName()), res, "skipped", suite, ctx, envt);
604     generatePanel(vc, tc.getPassedTests(), res, "passed", suite, ctx, envt);
605     // generatePanel(vc, tc.getPassedConfigurations(), res,
606     // "passed", suite, ctx, envt);
607 } else {
608     generatePanel(vc, failedTests.get(tc.getName()), res, "failed", suite, ctx, envt);
609     generatePanel(vc, skippedTests.get(tc.getName()), res, "skipped", suite, ctx, envt);
610     generatePanel(vc, tc.getPassedTests(), res, "passed", suite, ctx, envt);
611 }
612 } catch (Exception e) {
613     logger.error(e.getMessage());
614     e.printStackTrace();
615 }
616
617 return res.toString();
}
618
619 protected void generatePanel(VelocityEngine ve, ITestMap map, StringBuffer res, String style, ISuite suite, ITestCase ctx, boolean envt) {
620     Collection<ITestNGMethod> methodSet = getMethodSet(map);
621
622     for (ITestNGMethod method : methodSet) {
623         boolean methodsValid = true;
624         if (envt) {
625             methodsValid = Arrays.asList(method.getGroups()).contains("envt");
626         } else {
627             methodsValid = !Arrays.asList(method.getGroups()).contains("envt");
628         }
629
630         if (methodsValid) {
631
632             Collection<ITestResult> resultSet = getResultSet(map, method);
633             System.out.println(resultSet.size() + " resultset.size()");
634             String contextName = tc.getName().replace(' ', '_').replace('(', '_').replace(')', '_');
635             for (ITestResult rms : resultSet) {
636                 StringBuffer contentBuffer = new StringBuffer();
637                 String testName = "";
638                 String browserVersion = "";
639                 if (rms.getMethod() != null)
640                     testName = rms.getMethod().getXmlTest().getName();
641                 else
642                     testName = rms.getMethod().getXmlTest().getClassName();
643
644                 try {
645                     testName = rms.getTestContext().getCurrentXmlTest().getName();
646                 } catch (Exception ex) {
647                     ex.printStackTrace();
648                     continue;
649                 } catch (Error e) {
650                     e.printStackTrace();
651                     continue;
652                 }
653             }
654
655             Context testlevelContext = ContextManager.getTestLevelContext(testName);
656             if (testlevelContext != null) {
657                 String pool = testlevelContext.getAttribute("pool");
658                 String site = testlevelContext.getAttribute("site");
659                 String browser = (String)testlevelContext.getAttribute("browser");
660                 if (browser != null)
661                     browser = browser.replace(" ", "_");
662
663                 browser = browser.replace(" ", "_");
664                 String browserVersion = (String)testlevelContext.getAttribute("browserVersion");
665                 if (browserVersion != null)
666                     browser += browserVersion;
667                 contentBuffer.append("<div><i>Pool:</i> " + pool + "<i>Site:</i> " + site + "<i>Browser:</i> " + browser + "</span>");
668                 contentBuffer.append(" " + pool + " " + site + " " + browser);
669             }
670
671             Object[] parameters = rms.getParameters();
672             List<String> args = Reporter.getOutput(rms);
673
674             boolean hasReportedOutput = args.size() > 0;
675             Throwable exception = rms.getThrowable();
676             boolean hasException = exception != null;
677             if (hasReportedOutput || hasException) {
678                 contentBuffer.append("<div class='leftContent' style='float:left; width: 100%;>");
679                 contentBuffer.append("<div class='testlog' style='border: 1px solid black; padding: 5px; margin-bottom: 10px; float: left; width: 100%;>"); // style="display:none" ? "" : "[ - ]"
680                 contentBuffer.append("<div class='testlog' style='border: 1px solid black; padding: 5px; margin-bottom: 10px; float: left; width: 100%;>"); // style="display:none" ? "" : "[ - ]"
681                 contentBuffer.append("<ol>"); // style="list-style-type: none; padding-left: 0; margin: 0;">
682                 for (String logLine : rms.getOutputLines()) {
683                     DetailedLogLine outputDirectory;
684                     String htmlLog;
685                     if (logLine.getHref() == null) {
686                         htmlLog = "<a href=" + logLine.getHref() + " title=" + logLine.getLocation() + ">" + logLine.getMsg() + "</a>";
687                     } else {
688                         htmlLog = logLine.getMsg();
689                     }
690                     contentBuffer.append(htmlLog);
691                     if (htmlLog.contains("<br>")) contentBuffer.append("<br>"); // handle different in testing6.7
692                 }
693                 contentBuffer.append("</ol>"); // style="list-style-type: none; padding-left: 0; margin: 0;">
694
695                 if (hasException) {
696                     generateExceptionReport(exception, method, contentBuffer, new Date(ms.getEndMillis()));
697                 }
698             }
699             contentBuffer.append("</div></div>"); // end of div
700
701             //int rq = 0;
702             //if (rq > 0) {
703             String treeId = "tree" + m_treeId;
704             m_treeId++;
705             if (cons.getStatus() == 3) {
706                 contentBuffer.append("<br>This method has been skipped, because of its dependencies :<br>");
707                 takeCareOfDirectDependencies(suite, method, 0, ctx, treeId, contentBuffer);
708             }
709         }
710
711         contentBuffer.append("<div class='clear_both'></div>");
712         content = contentBuffer.toString();
713
714     } try {
715         Template t = ve.getTemplate("/templates/report_port.singleTest.html");
716         VelocityContext context = new VelocityContext();
717         context.put("status", style);
718         String desc = method.getDescription();
719         String desc = method.getDescription();
720
721         String tooltip = "no javadoc nor description for this method.";
722         if (!"".equals(javadoc) && javadoc != null) {
723             tooltip = javadoc;
724         } else if (!"".equals(desc) && desc != null) {
725             tooltip = desc;
726         }
727         String methodSignature = StringHelper.constructMethodSignature(method.getMethod(), parameters);
728         if (methodSignature.length() > 500)
729             context.put("methodName", methodSignature.substring(0, 500) + "... " + poolSiteBrowser);
730
731     } catch (Exception e) {
732         e.printStackTrace();
733     }
734 }
```



```
735     Template t = ve.getTemplate("/templates/report_port.singleTest.html");
736     VelocityContext context = new VelocityContext();
737     context.put("status", style);
738     String desc = method.getDescription();
739     String desc = method.getDescription();
740
741     String tooltip = "no javadoc nor description for this method.";
742     if (!"".equals(javadoc) && javadoc != null) {
743         tooltip = javadoc;
744     } else if (!"".equals(desc) && desc != null) {
745         tooltip = desc;
746     }
747     String methodSignature = StringHelper.constructMethodSignature(method.getMethod(), parameters);
748     if (methodSignature.length() > 500)
749         context.put("methodName", methodSignature.substring(0, 500) + "... " + poolSiteBrowser);
750
751 }
```

```

731         else
732             context.put("methodSignature", methodSignature + " " + poolSiteBrowser);
733             context.put("desc", toDisplay.replaceAll("\n\n", "\n").replaceAll("\n", "\n"));
734             context.put("content", content);
735             context.put("time", ((long) (oms.getEndMillis() - oms.getStartTimeMillis()) / 1000) + "sec.");
736             StringWriter writer = new StringWriter();
737             t.merge(context, writer);
738             res.append(writer.toString());
739         } catch (Exception e) {
740             logger.error("Error creating a singleTest." + e.getMessage());
741             e.printStackTrace();
742         }
743     }
744 }
745 }
746 }
747 }
748 }
749
750 public void generateReport(List<XmlSuite> xml, List<ISuite> suites, String outdir) {
751     ITestContext testctx = ContextManager.getGlobalContext().getTestNGContext();
752     if (testctx == null)
753         logger.error("Please check if your class extends from TestPlan!");
754     return;
755 }
756 File f = new File(ContextManager.getGlobalContext().getOutputDirectory());
757 setOutputDirectory(f.getParentFile().getAbsolutePath());
758 setResources(getOutputDirectory() + "\\resources");
759 try {
760     m_out = createWriter(getOutputDirectory());
761     startReport(testctx, m_out);
762     generateReportSummaryReport(suites, xml.get(0).getName());
763     generateReportSection(suites);
764     endHTML(m_out);
765     m_out.flush();
766     m_out.close();
767     copyResources();
768     logger.info("Report generated.");
769     String browserPath = testctx.getParameter(Context.OPEN_REPORT_IN_BROWSER);
770     String reportPath = (String) ContextManager.getGlobalContext().getAttribute(Context.OPEN_REPORT_IN_BROWSER);
771     if (browserPath != null && browserPath.trim().length() > 0)
772         executeCmd(browserPath, getReportLocation().getAbsolutePath());
773     } catch (Exception e) {
774         logger.error("Output file", e);
775         return;
776     }
777 }
778 }
779 }
780
781 protected void generateReportDetailsContainer(String name, int envtp, int envtf, int envts, int testip, int testsf, int tests, String envthml,
782     String testhtml) {
783     try {
784         VelocityEngine ve = new VelocityEngine();
785         ve.setProperty("resource.loader", "class");
786         ve.setProperty("class.resource.loader.class", "org.apache.velocity.runtime.resource.loader.ClasspathResourceLoader");
787         ve.init();
788
789         Template t = ve.getTemplate("/templates/report/part.testDetail.html");
790         VelocityContext context = new VelocityContext();
791         context.put("testid", name.toLowerCase().replace(' ', '_').replace('.', '_').replace('-', '_'));
792         context.put("testname", name);
793         context.put("envtp", envtp);
794         context.put("envts", envts);
795         context.put("testip", testip);
796         context.put("testsf", testsf);
797     }
798 }
799
800 protected void generateReportDetailsContainer(String name, int envtp, int envtf, int envts, int testip, int testsf, int tests, String envthml,
801     String testhtml, StringBuffer colCount, String globalErrorTabs, String globalErrorHtml) {
802     try {
803         VelocityEngine ve = new VelocityEngine();
804         ve.setProperty("resource.loader", "class");
805         ve.setProperty("class.resource.loader.class", "org.apache.velocity.runtime.resource.loader.ClasspathResourceLoader");
806         ve.init();
807
808         Template t = ve.getTemplate("/templates/report/part.testDetail.html");
809         VelocityContext context = new VelocityContext();
810         context.put("testid", name.toLowerCase().replace(' ', '_').replace('.', '_').replace('-', '_'));
811         context.put("testname", name);
812         context.put("envtp", envtp);
813         context.put("envts", envts);
814         context.put("testip", testip);
815         context.put("testsf", testsf);
816         context.put("tests", tests);
817         context.put("envthml", envthml);
818         context.put("testhtml", testhtml);
819         context.put("colcount", colCount.toString());
820         context.put("globalerrortabs", globalErrorTabs);
821         context.put("globalerrorhtmls", globalErrorHtml);
822
823         StringWriter writer = new StringWriter();
824         t.merge(context, writer);
825         m_out.write(writer.toString());
826     }
827 }
828
829 protected void generateReportsSection(List<ISuite> suites) {
830     m_out.println("<div id='reports'>");
831     for (ISuite suite : suites) {
832         Map<String, ISuiteResult> r = suite.getResults();
833         for (ISuiteResult r2 : r.values()) {
834             ITestContext tc = r2.getTestContext();
835
836             int envtp = getNbInstanceForGroup(true, tc.getPassedTests());
837             int envts = getNbInstanceForGroup(true, tc.getFailedTests().get(tc.getName()));
838             int envvs = getNbInstanceForGroup(true, tc.getSkippedTests().get(tc.getName()));
839             // envtp += getNbInstanceForGroup(true, tc.getPassedConfigurations());
840             envtp += getNbInstanceForGroup(true, tc.getFailedConfigurations());
841             envts += getNbInstanceForGroup(true, tc.getSkippedConfigurations());
842
843             int testp = getNbInstanceForGroup(false, tc.getPassedTests());
844

```

```

845             int testsf = getNbInstanceForGroup(false, tc.getFailedTests());
846             int tests = getNbInstanceForGroup(false, tc.getSkippedTests());
847             m_out.println("        <div class='suite'> " + suite.getName() + " </div> ");
848             m_out.println("            <table border='1'> " + suite.getResults().size() + " rows </table> ");
849             for (ISuiteResult r2 : r.values()) {
850                 ITestContext tc = r2.getTestContext();
851
852                 int envtp = getNbInstanceForGroup(true, tc.getPassedTests());
853                 int envts = getNbInstanceForGroup(true, tc.getFailedTests().get(tc.getName()));
854                 int envvs = getNbInstanceForGroup(true, tc.getSkippedTests().get(tc.getName()));
855                 // envtp += getNbInstanceForGroup(true, tc.getPassedConfigurations());
856                 envtp += getNbInstanceForGroup(true, tc.getFailedConfigurations());
857                 envts += getNbInstanceForGroup(true, tc.getSkippedConfigurations());
858
859                 int testp = getNbInstanceForGroup(false, tc.getPassedTests());
860                 int testsf = getNbInstanceForGroup(false, tc.getFailedTests());
861                 int tests = getNbInstanceForGroup(false, tc.getSkippedTests());
862             }
863         }
864     }
865     m_out.println("</div> ");
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
999 }
```

This screenshot shows a Java code editor within an IDE. The code is part of a class named `RaiseClaimTestPlan.java`. The code is responsible for generating suite summary reports. It includes logic for calculating various metrics such as total methods, passed tests, failed tests, and skipped tests. It also handles global error counts and generates HTML reports. The code uses `StringBuffer` objects to build up the report content.

```
862     int testp = getN0InstanceForGroup(false, tc.getPassedTests());
863     int testf = getN0InstanceForGroup(false, failedTests.get(tc.getName()));
864     int tests = getN0InstanceForGroup(false, skippedTests.get(tc.getName()));
865 
866     String envhtml = generateHTML(tc, true, suite, tc);
867     String testhtml = generateHTML(tc, false, suite, tc);
868     StringBuffer colcount = new StringBuffer();
869     // String colhtml = generateErrorHTML(tc, suite, colcount);
870 
871     StringBuffer globalErrorTabs = new StringBuffer();
872     StringBuffer globalErrorHTMLs = new StringBuffer();
873 
874     generateGlobalErrorHTML(tc, suite, globalErrorTabs, globalErrorHTMLs);
875 
876     generateReportDetailsContainer(tc.getName(), envip, envtf, envts, testp, testf, tests, envhtml, testhtml, colcount,
877         globalErrorTabs.toString(), globalErrorHTMLs.toString());
878 
879     }
880   }
881   }
882   }
883   }
884   }
885   }
886   }
887   public void generateSuiteSummaryReport(List<ISuite> suites, String suiteName) {
888     NumberFormat nfComma = new DecimalFormat("#,###.0");
889     int qty_tests = 0;
890     int qty_method = 0;
891     //int qty_pass_m = 0;
892     int qty_pass_s = 0;
893     int qty_skip = 0;
894     int qty_fail = 0;
895     long time_start = Long.MAX_VALUE;
896     long time_end = Long.MAX_VALUE;
897 
898     List<MiniTestResult> tests2 = new ArrayList<MiniTestResult>();
899     for (ISuite suite : suites) {
900       Map<String, ISuiteResults> tests = suite.getResults();
901       for (ISuiteResult r : tests.values()) {
902         if (r instanceof TestContext) {
903           ITestContext overview = r.getTestContext();
904           MiniTestResult mini = new MiniTestResult(overview.getName().replace(' ', '_').replace('<', '_').replace('>', '_'));
905           int q = getMethodSet(overview.getPassedTests()).size();
906           //qty_pass_m += q;
907           q = overview.getTotalMethod();
908           q += getMethodSet(overview.getFailedTests()).size();
909           q += getMethodSet(overview.getSkippedTests()).size();
910           q += overview.getPassedTests().size();
911           q += overview.getFailedTests().size();
912           q += overview.getSkippedTests().size();
913           mini.setTotalMethod(q);
914           mini.setInstancesPassed(q);
915           q = skippedTests.get(overview.getName()).size(); //getMethodSet(skippedTests.get(overview.getName())).size();
916           qty_skip += q;
917           q = getMethodSet(overview.getSkippedTests());
918           if (!isRetryableNeeded(overview.getName()))
919             q = failedTests.get(overview.getName()).size();
920           else
921             q = failedTests.get(overview.getName()).size() + getN0InstanceForGroup(true, overview.getFailedConfigurations());
922           qty_fail += q;
923           mini.setInstancesFailed(q);
924           time_start = Math.max(overview.getStartTime(), time_start);
925           time_end = Math.max(overview.getEndDate(), getEndTime(), time_end);
926           tests2.add(mini);
927         }
928       }
929     }
930     MiniTestResult total = new MiniTestResult("total");
931     total.setTotalMethod(qty_method);
932     total.setInstancesPassed(qty_pass_m);
933     total.setInstancesFailed(qty_fail);
934     total.setInstancesSkipped(qty_skip);
935 
936     try {
937       // BufferedWriter out = new BufferedWriter(new FileWriter(path));
938       // first, get and initialize an engine
939       VelocityEngine ve = new VelocityEngine();
940       ve.setProperty("resource.loader", "class");
941       ve.setProperty("class.resource.loader.class", "org.apache.velocity.runtime.resource.loader.ClasspathResourceLoader");
942       ve.init();
943 
944       Template t = ve.getTemplate("/templates/report_port.summary.html");
945       VelocityContext context = new VelocityContext();
946       context.put("suiteName", suiteName);
947       context.put("time_start", formatter.format((time_end - time_start) / 1000.) + " sec");
948       context.put("tests2", tests2);
949       context.put("total", total);
950 
951       StringWriter writer = new StringWriter();
952       t.merge(context, writer);
953       m_out.write(writer.toString());
954 
955     } catch (Exception e) {
956       logger.error(e.getMessage());
957     }
958   }
959 
960   private String getDate(Date date) {
961     SimpleDateFormat df = new SimpleDateFormat("HH:mm:ss,SSS");
962     return df.format(date);
963   }
964 
965   protected void generateTheStackTrace(Throwable exception, ITestNGMethod method, String title, StringBuffer contentBuffer, Date date) {
966     contentBuffer.append("<div class='stContainer'>" + exception.getClassName() + ":" + escape(title)
967     + "<a href='javascript:void(0);' class='exceptionLink'>stacktrace</a>" );
968 
969     contentBuffer.append("<div class='exception' style='display:none;'>" + getDate(date) + "&ampnbsp" );
970 
971     StackTraceElement[] sl = exception.getStackTrace();
972     Throwable t2 = exception.getCause();
973     if (t2 == exception) {
974       t2 = null;
975     }
976 
977     for (int x = 0; x < sl.length; x++) {
978       contentBuffer.append("<br/>" + escape(sl[x].toString()));
979     }
980 
981     if (t2 != null) {
982       generateTheStackTrace(t2, method, "Caused by " + t2.getLocalizedMessage(), contentBuffer, date);
983     }
984     contentBuffer.append("</div></div>");
985   }
986 
987   protected Collection<ITestNGMethod> getAllMethods(ISuite suite) {
988     Set<ITestNGMethod> all = new LinkedHashSet<ITestNGMethod>();
989     Map<String, Collection<ITestNGMethod>> methods = suite.getMethodsByGroups();
990     for (Entry<String, Collection<ITestNGMethod>> group : methods.entrySet()) {
991       all.addAll(methods.get(group.getKey()));
992     }
993   }
994 }
```

This screenshot shows another Java code editor within the same IDE. This code is part of the `RaiseClaimTestPlan.java` class. It contains a method for generating stack traces for exceptions. The code uses `StringBuffer` to build the stack trace output. It handles both the main exception and its cause, ensuring the stack trace is correctly formatted and displayed.

```
862     int testp = getN0InstanceForGroup(false, tc.getPassedTests());
863     int testf = getN0InstanceForGroup(false, failedTests.get(tc.getName()));
864     int tests = getN0InstanceForGroup(false, skippedTests.get(tc.getName()));
865 
866     String envhtml = generateHTML(tc, true, suite, tc);
867     String testhtml = generateHTML(tc, false, suite, tc);
868     StringBuffer colcount = new StringBuffer();
869     // String colhtml = generateErrorHTML(tc, suite, colcount);
870 
871     StringBuffer globalErrorTabs = new StringBuffer();
872     StringBuffer globalErrorHTMLs = new StringBuffer();
873 
874     generateGlobalErrorHTML(tc, suite, globalErrorTabs, globalErrorHTMLs);
875 
876     generateReportDetailsContainer(tc.getName(), envip, envtf, envts, testp, testf, tests, envhtml, testhtml, colcount,
877         globalErrorTabs.toString(), globalErrorHTMLs.toString());
878 
879     }
880   }
881   }
882   }
883   }
884   }
885   }
886   }
887   public void generateSuiteSummaryReport(List<ISuite> suites, String suiteName) {
888     NumberFormat nfComma = new DecimalFormat("#,###.0");
889     int qty_tests = 0;
890     int qty_method = 0;
891     //int qty_pass_m = 0;
892     int qty_pass_s = 0;
893     int qty_skip = 0;
894     int qty_fail = 0;
895     long time_start = Long.MAX_VALUE;
896     long time_end = Long.MAX_VALUE;
897 
898     List<MiniTestResult> tests2 = new ArrayList<MiniTestResult>();
899     for (ISuite suite : suites) {
900       Map<String, ISuiteResults> tests = suite.getResults();
901       for (ISuiteResult r : tests.values()) {
902         if (r instanceof TestContext) {
903           ITestContext overview = r.getTestContext();
904           MiniTestResult mini = new MiniTestResult(overview.getName().replace(' ', '_').replace('<', '_').replace('>', '_'));
905           int q = getMethodSet(overview.getPassedTests()).size();
906           //qty_pass_m += q;
907           q = overview.getTotalMethod();
908           q += getMethodSet(overview.getFailedTests()).size();
909           q += getMethodSet(overview.getSkippedTests()).size();
910           q += overview.getPassedTests().size();
911           q += overview.getFailedTests().size();
912           q += overview.getSkippedTests().size();
913           mini.setTotalMethod(q);
914           mini.setInstancesPassed(q);
915           q = skippedTests.get(overview.getName()).size(); //getMethodSet(skippedTests.get(overview.getName())).size();
916           qty_skip += q;
917           q = getMethodSet(overview.getSkippedTests());
918           if (!isRetryableNeeded(overview.getName()))
919             q = failedTests.get(overview.getName()).size();
920           else
921             q = failedTests.get(overview.getName()).size() + getN0InstanceForGroup(true, overview.getFailedConfigurations());
922           qty_fail += q;
923           mini.setInstancesFailed(q);
924           time_start = Math.max(overview.getStartTime(), time_start);
925           time_end = Math.max(overview.getEndDate(), getEndTime(), time_end);
926           tests2.add(mini);
927         }
928       }
929     }
930     MiniTestResult total = new MiniTestResult("total");
931     total.setTotalMethod(qty_method);
932     total.setInstancesPassed(qty_pass_m);
933     total.setInstancesFailed(qty_fail);
934     total.setInstancesSkipped(qty_skip);
935 
936     try {
937       // BufferedWriter out = new BufferedWriter(new FileWriter(path));
938       // first, get and initialize an engine
939       VelocityEngine ve = new VelocityEngine();
940       ve.setProperty("resource.loader", "class");
941       ve.setProperty("class.resource.loader.class", "org.apache.velocity.runtime.resource.loader.ClasspathResourceLoader");
942       ve.init();
943 
944       Template t = ve.getTemplate("/templates/report_port.summary.html");
945       VelocityContext context = new VelocityContext();
946       context.put("suiteName", suiteName);
947       context.put("time_start", formatter.format((time_end - time_start) / 1000.) + " sec");
948       context.put("tests2", tests2);
949       context.put("total", total);
950 
951       StringWriter writer = new StringWriter();
952       t.merge(context, writer);
953       m_out.write(writer.toString());
954 
955     } catch (Exception e) {
956       logger.error(e.getMessage());
957     }
958   }
959 
960   private String getDate(Date date) {
961     SimpleDateFormat df = new SimpleDateFormat("HH:mm:ss,SSS");
962     return df.format(date);
963   }
964 
965   protected void generateTheStackTrace(Throwable exception, ITestNGMethod method, String title, StringBuffer contentBuffer, Date date) {
966     contentBuffer.append("<div class='stContainer'>" + exception.getClassName() + ":" + escape(title)
967     + "<a href='javascript:void(0);' class='exceptionLink'>stacktrace</a>" );
968 
969     contentBuffer.append("<div class='exception' style='display:none;'>" + getDate(date) + "&ampnbsp" );
970 
971     StackTraceElement[] sl = exception.getStackTrace();
972     Throwable t2 = exception.getCause();
973     if (t2 == exception) {
974       t2 = null;
975     }
976 
977     for (int x = 0; x < sl.length; x++) {
978       contentBuffer.append("<br/>" + escape(sl[x].toString()));
979     }
980 
981     if (t2 != null) {
982       generateTheStackTrace(t2, method, "Caused by " + t2.getLocalizedMessage(), contentBuffer, date);
983     }
984     contentBuffer.append("</div></div>");
985   }
986 
987   protected Collection<ITestNGMethod> getAllMethods(ISuite suite) {
988     Set<ITestNGMethod> all = new LinkedHashSet<ITestNGMethod>();
989     Map<String, Collection<ITestNGMethod>> methods = suite.getMethodsByGroups();
990     for (Entry<String, Collection<ITestNGMethod>> group : methods.entrySet()) {
991       all.addAll(methods.get(group.getKey()));
992     }
993   }
994 }
```

A screenshot of an IDE interface showing a Java code editor. The code is part of a class named `RaiseClaimTestPlan.java`. The editor shows several methods and annotations, including `@Override`, `@TestNGMethod`, and `@SuppressWarnings`. The code is related to handling test results and generating JavaDoc comments. The IDE has a standard toolbar at the top and a sidebar on the right.

```
995     protected int getDim(Class<?> cls) {
996         int dim = 0;
997         while (!cls.isArray()) {
998             dim++;
999             cls = cls.getComponentType();
1000         }
1001         return dim;
1002     }
1003
1004     public int getEnvConfigTestsCount(IResultMap map) {
1005         int count = 0;
1006         for (ITestNGMethod method : map.getAllMethods()) {
1007             String[] groups = method.getGroups();
1008             if (groups != null) {
1009                 for (int i = 0; i < groups.length; i++) {
1010                     if ("envt".equalsIgnoreCase(groups[i])) {
1011                         count++;
1012                         break;
1013                     }
1014                 }
1015             }
1016         }
1017         return count;
1018     }
1019
1020     protected ITestResult getFailedSkippedResults(ITestContext ctx, ITestNGMethod method) {
1021         List res = new ArrayList();
1022         res.addAll(method.getFailedTests());
1023         if (res.size() != 0) {
1024             return res.get(0);
1025         }
1026         res.addAll(ctx.getPassedTests().getResults(method));
1027         if (res.size() != 0) {
1028             return res.get(0);
1029         }
1030         res.addAll(skippedTests.get(ctx.getName()).getResults(method));
1031         if (res.size() != 0) {
1032             return res.get(0);
1033         }
1034     }
1035
1036     /**
1037      * @SupressWarnings("CrownTypes")
1038     protected JavaDocBuilder getJavaDocBuilder(Class clz) throws URISyntaxException {
1039         String packagePath = "com/ebay/nail/register";
1040         String docPath = new File(clz).getAbsolutePath();
1041         String packagePath = clz.getName().replaceAll("\\\\", "/");
1042         if (builder == null) {
1043             builder = new JavaDocBuilder();
1044
1045             URL resource = Thread.currentThread().getContextClassLoader().getResource(packagePath);
1046             File src = new File(resource.toURI());
1047             builder.addSourceTree(src);
1048
1049             // project source folder
1050             File realFolder = new File(projectPath + "/src/main/java/" + packagePath);
1051             if (realFolder.exists())
1052                 builder.addSourceTree(realFolder);
1053         }
1054         return builder;
1055     }
1056
1057     /**
1058     */
1059 }
```

A screenshot of an IDE interface showing a Java code editor. The code is part of the same `RaiseClaimTestPlan.java` class. It includes methods for getting JavaDoc comments, a collection of test methods, and an instance for a group. The code uses annotations like `@Override`, `@TestNGMethod`, and `@Param`. The IDE interface is similar to the one above, with a toolbar and a sidebar.

```
1065     protected String getJavadocComments(ITestNGMethod method) {
1066
1067         try {
1068             Method m = method.getMethod();
1069             String javaclass = m.getDeclaringClass().getName();
1070             String jmethod = m.getName();
1071             JavaClass jc = getJavaDocBuilder(javaclass).getClassByName(javaclass);
1072             Class<?>[] types = method.getMethod().getParameterTypes();
1073             Type[] qdoxTypes = new Type[types.length];
1074             for (int i = 0; i < types.length; i++) {
1075                 String type = types[i].getCanonicalName();
1076                 String type = getJavadocType(type);
1077                 int dim = getDim(types[i]);
1078                 // System.out.println(" " + type + " " + dim);
1079                 qdoxTypes[i] = new Type(type, dim);
1080             }
1081             jc.getMethodBySignature(jmethod, qdoxTypes);
1082             return jc.getComments();
1083         } catch (Throwable e) {
1084             logger.error("error loading the javadoc comments for :" + method.getMethodName() + e);
1085         }
1086         return null;
1087     }
1088
1089     /**
1090      * @param tests
1091      * @return
1092     */
1093     protected Collection<ITestNGMethod> getMethodSet(IResultMap tests) {
1094         Set<ITestNGMethod> r = new TreeSet<ITestNGMethod>(<new TestMethodSorter<ITestNGMethod>>());
1095         r.addAll(tests.getAllMethods());
1096         return r;
1097     }
1098
1099     /**
1100      * @param boolean forGroup
1101      * @return
1102     */
1103     protected int getNBInstanceForGroup(boolean envt, IResultMap tests) {
1104         int res = 0;
1105
1106         for (ITestResult result : tests.getAllResults()) {
1107             boolean resultIsAnEnvRes = Arrays.asList(result.getMethod().getGroups()).contains("envt");
1108
1109             if (resultIsAnEnvRes) {
1110                 if (envt) {
1111                     res++;
1112                 } else {
1113                     if (!envt) {
1114                         res++;
1115                     }
1116                 }
1117             }
1118         }
1119
1120         public String getOutputDirectory() {
1121             return outputDirectory;
1122         }
1123
1124         public File getReportLocation() {
1125             return report;
1126         }
1127
1128         public String getResources() {
1129             return resources;
1130         }
1131     }
1132 }
```

```

1130    protected Collection<ITestResult> getTestResultSet(ITestNGMethod method) {
1131        Set<ITestResult> r = new TreeSet<ITestResult>(new TestResultSorter());
1132        for (ITestResult result : tests.getAllResults()) {
1133            if (result.getMethod().getMethodName().equals(method.getMethodName())) {
1134                r.add(result);
1135            }
1136        }
1137        return r;
1138    }
1139
1140    protected ITestNGMethod getTestMethod(ITestContext ctx, String method) {
1141        Collection<ITestNGMethods> methods = new HashSet<ITestNGMethod>();
1142
1143        //filter
1144        //TESTING 4.3. Skip method does not start with package name. So strip off the package name before comparing.
1145        int index = method.substring(0, method.lastIndexOf('.')).lastIndexOf('.');
1146        String localMethod = method.substring(index+1);
1147
1148        ITestNGMethod[] all = ctx.getAllTestMethods();
1149        for (int i = 0; i < all.length; i++) {
1150            methods.add(all[i]);
1151        }
1152        for (ITestNGMethod m : methods) {
1153            if (m.toString().startsWith(localMethod + "O")) {
1154                return m;
1155            }
1156        }
1157        throw new RuntimeException("method " + method + " not found. " + "Should not happen. Suite " + ctx.getName());
1158    }
1159
1160    protected String getType(Class<?> cls) {
1161        while (cls.isArray()) {
1162            cls = cls.getComponentType();
1163        }
1164        return cls.getName();
1165    }
1166
1167    protected boolean hasDependencies(ITestNGMethod method) {
1168        return (method.getGroupsDependedUpon().length + method.getMethodsDependedUpon().length) != 0;
1169    }
1170
1171    protected Map<String, ITestResult> initMethodsByGroup() {
1172        Map<String, ITestResult> methodsByGroup = new HashMap<String, ITestResult>;
1173        return null;
1174    }
1175
1176    public void onfinish(ITestContext arg0) {
1177        // runEnd = Calendar.getInstance().getTime();
1178        // ...
1179        // Thread soothThread = new Thread()@Override
1180        // public void run() {
1181        //     if (context.isCalCollectionEnabled()) {
1182        //         soothReporter.onFinish(arg0);
1183        //     }
1184        // }
1185        // ...
1186        // soothThread.start();
1187        // ...
1188        // if (Context.isCalCollectionEnabled()) {
1189        //     logger.info("Collecting CAL Events...");
1190        // }
1191        // Calendar counter = Calendar.getInstance();
1192    }

```

```

1240    // ...
1241    // ...
1242    if (isRetryHandleNeeded.getArg0().getName()) {
1243        removeIncorrectlySkippedTests(arg0, failedTests.getArg0.getName());
1244        removeFailedTestsInTestNG(arg0);
1245    } else {
1246        failedTests.put(arg0.getName(), arg0.getFailedTests());
1247        skippedTests.put(arg0.getName(), arg0.getSkippedTests());
1248    }
1249    System.out.println("----- MUI Reports -----");
1250    if (failedCoses.size() == 0) {
1251        System.out.println("Suite Passed!");
1252    } else {
1253        System.out.println("Suite Failed!");
1254    }
1255    System.out.println("Passed:" + arg0.getPassedTests().size() + ", Skipped:" + skippedCoses.size() + ", Failed:" + failedCoses.size());
1256    System.out.println("-----");
1257
1258
1259}
1260
1261    public void onstart(ITestContext arg0) {
1262        // runBegin = Calendar.getInstance().getTime();
1263        // if (context.isCalCollectionEnabled()) {
1264        //     soothReporter.onStart(arg0);
1265        // }
1266        isRetryHandleNeeded.putArg0.getName(), false);
1267        failedTests.putArg0.getName(), new ResultMap());
1268        skippedTests.putArg0.getName(), new ResultMap());
1269    }
1270
1271    public void ontestFailedButWithinSuccessPercentage(ITestResult arg0) {
1272        // if (Context.isCalCollectionEnabled()) {
1273        //     soothReporter.onTestFailedButWithinSuccessPercentage(arg0);
1274        // }
1275    }
1276
1277    public synchronized void ontestFailure(ITestResult arg0) {
1278        if (arg0.getMethod().getRetryAnalyzer() != null) {
1279            TestRetryAnalyzer testRetryAnalyzer = (TestRetryAnalyzer) arg0.getMethod().getRetryAnalyzer();
1280            if (testRetryAnalyzer.getCount() == testRetryAnalyzer.getMaxCount()) {
1281                arg0.setStatus(ITestResult.SKIP);
1282                Reporter.setCurrentTestResult(null);
1283            } else {
1284                IResultMap rMap = failedTests.get(arg0.getTestContext().getName());
1285                if (rMap == null) {
1286                    rMap = new ResultMap();
1287                    failedTests.put(arg0.getTestContext().getName(), rMap);
1288                }
1289                rMap.addResult(arg0, arg0.getMethod());
1290                System.out.println(arg0.getMethod() + " Failed in " + testRetryAnalyzer.getCount() + " times");
1291            }
1292        }
1293        if (isRetryHandleNeeded.getArg0().getName(), true);
1294        soothReporter.onTestFailure(arg0);
1295    }
1296
1297    public void ontestSkipped(ITestResult arg0) {
1298        // if (Context.isCalCollectionEnabled()) {
1299        //     soothReporter.onTestSkipped(arg0);
1300        // }
1301    }
1302
1303}
1304
1305    public void ontestStart(ITestResult arg0) {
1306        // if (Context.isCalCollectionEnabled())
1307    }

```

```
1305     // If Context is CalCollection (enabled)
1306     // so Reporter.onTestStart (arg0);
1307     //
1308 }
1309
1310 /**
1311  * public void onTestSuccess(ITestResult arg0) {
1312  *     // If Context is CalCollection (enabled)
1313  *     // so Reporter.onTestSuccess (arg0);
1314  *     //
1315  *     if (arg0.getMethod().getRetryAnalyzer() != null) {
1316  *         TestRetryAnalyzer testRetryAnalyzer = (TestRetryAnalyzer) arg0.getMethod().getRetryAnalyzer();
1317  *
1318  *         System.out.println(arg0.getMethod()) + " Passed in " + testRetryAnalyzer.getCount() + " times";
1319  *
1320     }
1321     isRetryHandleNeeded = true;
1322 }
1323
1324
1325 /**
1326 * Remote failed test cases in TestNG
1327 * @param tc
1328 * @return
1329 */
1330 private void removeFailedTestsInTestNG(ITestContext tc) {
1331     IResultMap returnValue = tc.getAllResults();
1332
1333     IResultMap removeMap = new ResultMap();
1334
1335     for (ITestResult result : returnValue.getAllResults()) {
1336         boolean isFailed = false;
1337         for (ITestResult resultToCheck : failedTests.get(tc.getName()).getAllResults()) {
1338             if (result.getMethod().equals(resultToCheck.getMethod()) && result.getEndMillis() == resultToCheck.getEndMillis())
1339             {
1340                 //logger.info("Keep Failed cases:" + result.getMethod().getMethodName());
1341                 isFailed = true;
1342                 break;
1343             }
1344         }
1345     }
1346     if (!isFailed) {
1347         //logger.info("Removed failed cases:" + result.getMethod().getMethodName());
1348         System.out.println("Removed failed cases:" + result.getMethod().getMethodName());
1349         //test.getFailedTests().getAllResults().remove(result);
1350         removeMap.addResult(result, result.getMethod());
1351         //test.getFailedTests().remove(result.getResultMethod());
1352     }
1353 }
1354
1355 for (ITestResult result : removeMap.getAllResults()) {
1356     ITestResult removeResult = null;
1357     for (ITestResult resultToCheck : returnValue.getAllResults()) {
1358         if (result.getMethod().equals(resultToCheck.getMethod()) && result.getEndMillis() == resultToCheck.getEndMillis())
1359         {
1360             removeResult = resultToCheck;
1361             break;
1362         }
1363     }
1364     if (removeResult != null) returnValue.getAllResults().remove(removeResult);
1365 }
1366
1367 /**
1368 * Remove retrying failed test cases from skipped test cases
1369 */
1370
1371 /**
1372 * Remove retrying failed test cases from skipped test cases
1373 */
1374
```

```
1375     private void removeIncorrectlySkippedTests(ITestContext tc, IResultMap map)
1376     {
1377         List<ITestNGMethod> failsToRemove = new ArrayList<ITestNGMethod>();
1378         IResultMap returnValue = tc.getSkippedTests();
1379
1380         for (ITestResult result : returnValue.getAllResults())
1381         {
1382             for (ITestResult resultToCheck : map.getAllResults())
1383             {
1384                 if (resultToCheck.getMethod().equals(result.getMethod()))
1385                 {
1386                     failsToRemove.add(resultToCheck.getMethod());
1387                     break;
1388                 }
1389             }
1390         }
1391
1392         for (ITestResult result : returnValue.getAllResults())
1393         {
1394             for (ITestResult resultToCheck : tc.getPossessedTests().getAllResults())
1395             {
1396                 if (resultToCheck.getMethod().equals(result.getMethod()))
1397                 {
1398                     failsToRemove.add(resultToCheck.getMethod());
1399                     break;
1400                 }
1401             }
1402         }
1403
1404         for (ITestNGMethod method : failsToRemove)
1405         {
1406             returnValue.removeResult(method);
1407         }
1408
1409         skippedTests.put(tc.getName(), tc.getSkippedTests());
1410     }
1411
1412
1413     public void setOutputDirectory(String outTimestamped) {
1414         this.outputDirectory = outTimestamped;
1415     }
1416
1417     public void setReportId(String uuid) {
1418         this.uuid = uuid;
1419     }
1420
1421     public void setResources(String resources) {
1422         this.resources = resources;
1423     }
1424
1425     /** Starts HTML stream */
1426     protected void startHtml(ITestContext ctx, PrintWriter out) {
1427         try {
1428             // BufferedWriter out = new BufferedWriter(new FileWriter(path));
1429
1430             // First, get and initialize an engine
1431             VelocityEngine ve = new VelocityEngine();
1432             ve.setProperty("resource.loader", "class");
1433             ve.setProperty("class.resource.loader.class", "org.apache.velocity.runtime.resource.loader.ClasspathResourceLoader");
1434             ve.init();
1435
1436             Template t = ve.getTemplate("/templates/report.part.header.html");
1437             VelocityContext context = new VelocityContext();
1438
1439             String original = System.getProperty("user.name");
1440
1441             // String poolInfo = "a target=_blank href=" +
1442             // WebHelper.convertURL("http://signin.fpi24.qa.ebay.com/admin/v3console/ValidateInternal");
1443             // + ">";
1444
1445         } catch (Exception e) {
1446             e.printStackTrace();
1447         }
1448     }
1449
1450 }
```

```

1455     // ContextManager.getContext().getAttribute(Context.REPORT_TITLE);
1456     context.put("title", String.valueOf(context.getAttribute(Context.REPORT_TITLE)));
1457     context.put("userName", userName);
1458     context.put("currentDate", new Date().toString());
1459     // ContextManager.getContext().getAttribute(Context.RUN_ID);
1460     // ContextManager.getContext().getAttribute(Context.POOL);
1461     context.put("pool", poolInfo);
1462     context.put("poolName", String.valueOf(context.getAttribute(Context.API_POOL)));
1463     // ContextManager.getContext().getAttribute(Context.API_POOL);
1464     // ContextManager.getContext().getWebInfo();
1465     String hubUrl = ContextManager.getGlobalContext().getWebServerUrl();
1466     //context.put("gridhub", "<a href='"+ hubUrl + "' target='huburl'" + (null == huburl? null : new URL(hubUrl).getHost()) + "></a>");
1467     context.put("gridhub", "<a href='"+ hubUrl + "' target='huburl'" + hubUrl + "></a>");  

1468     context.put("node", node);
1469
1470     context.put("groups", sbGroups);
1471
1472     StringBuffer sbGroups = new StringBuffer();
1473     Set<AbstractPageListener> pgolistenerSet = PluginsUtil.getInstance().getPgolisteners();
1474     if (pgolistenerSet != null && !pgolistenerSet.isEmpty()) {
1475         for (AbstractPageListener abstractPgolistener : pgolistenerSet) {
1476             sbGroups.append(abstractPgolistener.getClass().getSimpleName());
1477         }
1478     }
1479
1480     context.put("groups", sbGroups.toString());
1481
1482     StringWriter writer = new StringWriter();
1483     t.merge(context, writer);
1484     out.write(writer.toString());
1485
1486 } catch (Exception e) {
1487     logger.error(e.getMessage());
1488 }
1489
1490 }
1491
1492 protected void takeCareOfDirectDependencies( ISuite suite, ITestNGMethod method, int indent, ITestContext ctx, String treeid, StringBuffer res) {
1493
1494     if (indent == 0) {
1495         res.append("<a href='"+ onclick+"expandTree(" + treeid + "'); return false;'>Expand All</a>&ampnbsp");
1496         res.append("<a href='"+ onclick+"collapseTree(" + treeid + "'); return false;'>Collapse All</a>");
1497     }
1498
1499
1500     String[] methStr = method.getMethodsDependedUpon();
1501     if (methStr.length != 0) {
1502         Set<ITestNGMethod> methSet = new LinkedHashSet<ITestNGMethod>();
1503         for (int i = 0; i < methStr.length; i++) {
1504             ITestNGMethod m = suite.getTestNGMethod(ctx, methStr[i]);
1505             String intendstr = "<ul>";
1506             for (int j = 0; j < indent; j++) {
1507                 intendstr += "<li>";
1508             }
1509             String img = "<img src='";
1510             img += m.getRoot() + "/test" + getFailedOrSkippedResult(ctx, m).getStatus() + ".gif";
1511             img += "'/>";
1512
1513             res.append(intendstr + "<li>" + img + m);
1514             if (hasDependencies(m)) {
1515                 res.append(intendstr + "<ul>");
1516                 res.append("<li>" + m.getRoot() + "/test" + getFailedOrSkippedResult(ctx, m).getStatus() + ".gif");
1517                 takeCareOfDirectDependencies(suite, m, indent + 1, ctx, treeid, res);
1518                 res.append(intendstr + "</ul>");
1519             }
1520             res.append("</li>");
1521     }
1522
1523 }
1524
1525 for (int i = 0; i < method.getGroupsDependedUpon().length; i++) {
1526     if (methodsbGroup == null) {
1527         // Collection<ITestNGMethod> all = suite.getInvokedMethods();
1528         methodsbGroup = suite.getMethodsByGroups();
1529         // System.out.println(all);
1530     }
1531     String dependentGroup = method.getGroupsDependedUpon()[i];
1532
1533     Set<ITestNGMethod> methods = new LinkedHashSet<ITestNGMethod>();
1534     Collection<ITestNGMethod> c = suite.getMethodsByGroups().get(dependentGroup);
1535
1536     if (c != null) {
1537         methods.addAll(c);
1538         res.append("<li>" + dependentGroup + "</li>");
1539         res.append("<ul>");
1540
1541         for (ITestNGMethod m : methods) {
1542             String intendstr = "<ul>";
1543             for (int j = 0; j < indent; j++) {
1544                 intendstr += "<li>";
1545             }
1546             String img = "<img src='";
1547             img += m.getRoot() + "/test" + getFailedOrSkippedResult(ctx, m).getStatus() + ".gif";
1548             img += "'/>";
1549             res.append(intendstr + "<li>" + img + m);
1550             if (hasDependencies(m)) {
1551                 res.append(intendstr + "<ul>");
1552                 res.append("<li>" + m.getRoot() + "/test" + getFailedOrSkippedResult(ctx, m).getStatus() + ".gif");
1553                 takeCareOfDirectDependencies(suite, m, indent + 1, ctx, treeid, res);
1554                 res.append(intendstr + "</ul>");
1555             }
1556             res.append("</li>");
1557         }
1558         res.append("</ul>");
1559         res.append("</li>");
1560     }
1561
1562     if (indent == 0) {
1563         res.append("</ul>");
1564     }
1565 }
1566 }

```

```

1560
1561     String[] methStr = method.getMethodsDependedUpon();
1562     if (methStr.length != 0) {
1563         Set<ITestNGMethod> methSet = new LinkedHashSet<ITestNGMethod>();
1564         for (int i = 0; i < methStr.length; i++) {
1565             ITestNGMethod m = suite.getTestNGMethod(ctx, methStr[i]);
1566             String intendstr = "<ul>";
1567             for (int j = 0; j < indent; j++) {
1568                 intendstr += "<li>";
1569             }
1570             String img = "<img src='";
1571             img += m.getRoot() + "/test" + getFailedOrSkippedResult(ctx, m).getStatus() + ".gif";
1572             img += "'/>";
1573
1574             res.append(intendstr + "<li>" + img + m);
1575             if (hasDependencies(m)) {
1576                 res.append(intendstr + "<ul>");
1577                 takeCareOfDirectDependencies(suite, m, indent + 1, ctx, treeid, res);
1578                 res.append(intendstr + "</ul>");
1579             }
1580             res.append("</li>");
1581     }
1582
1583
1584     for (int i = 0; i < method.getGroupsDependedUpon().length; i++) {
1585         if (methodsbGroup == null) {
1586             // Collection<ITestNGMethod> all = suite.getInvokedMethods();
1587             methodsbGroup = suite.getMethodsByGroups();
1588             // System.out.println(all);
1589         }
1590         String dependentGroup = method.getGroupsDependedUpon()[i];
1591
1592         Set<ITestNGMethod> methods = new LinkedHashSet<ITestNGMethod>();
1593         Collection<ITestNGMethod> c = suite.getMethodsByGroups().get(dependentGroup);
1594
1595         if (c != null) {
1596             methods.addAll(c);
1597             res.append("<li>" + dependentGroup + "</li>");
1598             res.append("<ul>");
1599
1600             for (ITestNGMethod m : methods) {
1601                 String intendstr = "<ul>";
1602                 for (int j = 0; j < indent; j++) {
1603                     intendstr += "<li>";
1604                 }
1605                 String img = "<img src='";
1606                 img += m.getRoot() + "/test" + getFailedOrSkippedResult(ctx, m).getStatus() + ".gif";
1607                 img += "'/>";
1608                 res.append(intendstr + "<li>" + img + m);
1609                 if (hasDependencies(m)) {
1610                     res.append(intendstr + "<ul>");
1611                     res.append("<li>" + m.getRoot() + "/test" + getFailedOrSkippedResult(ctx, m).getStatus() + ".gif");
1612                     takeCareOfDirectDependencies(suite, m, indent + 1, ctx, treeid, res);
1613                     res.append(intendstr + "</ul>");
1614                 }
1615                 res.append("</li>");
1616             }
1617             res.append("</ul>");
1618             res.append("</li>");
1619         }
1620
1621     if (indent == 0) {
1622         res.append("</ul>");
1623     }
1624 }
1625
1626 }

```

A screenshot of an IDE interface showing the XMLReporter.java file. The code implements the IReporter interface, handling XML reporting for suites. It includes methods for generating reports, setting output directories, and managing stack trace output. The code uses XMLStringBuffer to build XML documents and Logging.getLogger for logging.

```
32
33     /**
34      * public class XMLReporter implements IReporter {
35      *     private static Logger logger = Logging.getLogger(XMLReporter.class);
36      *
37      *     private XMLReporterConfig config = new XMLReporterConfig();
38      *     private XMLStringBuffer rootBuffer;
39
40     public void generateReport(List<ISuite> xmlSuites, List<ISuite> suites, String outputDirectory) {
41         if (utils.isEmpty(config.getOutputDirectory())) {
42             config.setOutputDirectory(outputDirectory);
43         }
44
45         rootBuffer = new XMLStringBuffer("<*>");
46         rootBuffer.push(XMLReporterConfig.TAG_TESTING_RESULTS);
47
48         writeReportToOutput(rootBuffer);
49         for (int i = 0; i < suites.size(); i++) {
50             writeSuite(xmlSuites.get(i), suites.get(i));
51         }
52         rootBuffer.pop();
53         utils.writeFile(config.getOutputDirectory(),
54                         // "testing-results-ebay.xml", rootBuffer.toXML());
55         try {
56             utils.writeFile(ExecutionContextManager.getGlobalContext().getOutputDirectory(), "testing-results-ebay.xml", rootBuffer.toXML());
57         } catch (Throwable e) {
58             logger.warn("Ex", e);
59         }
60
61         // TODO: This is not the shortest way to implement the config
62         public int getFileFragmentationLevel() {
63             return config.getFileFragmentationLevel();
64         }
65
66         public String getOutputDirectory() {
67             return config.getOutputDirectory();
68         }
69
70         public int getStackTraceOutputMethod() {
71             return config.getStackTraceOutputMethod();
72         }
73
74         private Properties getSuiteAttributes(ISuite suite) {
75             Properties props = new Properties();
76             props.setProperty(XMLReporterConfig.ATTR_NAME, suite.getName());
77             return props;
78         }
79
80         public String getTimestampFormat() {
81             return XMLReporterConfig.getTimestampFormat();
82         }
83
84         private Set<ITestNGMethod> getUniqueMethodSet(Collection<ITestNGMethod> methods) {
85             Set<ITestNGMethod> result = new LinkedHashSet<ITestNGMethod>();
86             for (ITestNGMethod method : methods) {
87                 result.add(method);
88             }
89             return result;
90         }
91
92         public boolean isGenerateDependsOnGroups() {
93             return config.isGenerateDependsOnGroups();
94         }
95
96         public boolean isGenerateDependsOnMethods() {
97             return config.isGenerateDependsOnMethods();
98         }
99
100    }
```

A screenshot of an IDE interface showing the XMLReporter.java file. The code implements the IReporter interface, handling XML reporting for suites. It includes methods for generating reports, setting output directories, and managing stack trace output. The code uses XMLStringBuffer to build XML documents and Logging.getLogger for logging.

```
101     public boolean isGenerateGroupsAttribute() {
102         return config.isGenerateGroupsAttribute();
103     }
104
105     public boolean isSplitClassAndPackageName() {
106         return config.isSplitClassAndPackageName();
107     }
108
109     private File referenceSuite(XMLStringBuffer xmlBuffer, ISuite suite) {
110         String relativePath = suite.getName() + File.separatorChar + "testing-results.xml";
111         File suiteFile = new File(config.getOutputDirectory(), relativePath);
112         Properties attrs = new Properties();
113         attrs.setProperty(XMLReporterConfig.ATTR_URL, relativePath);
114         xmlBuffer.addElement(XMLReporterConfig.TAG_SUITE, attrs);
115         return suiteFile;
116     }
117
118     public void setFileFragmentationLevel(int fileFragmentationLevel) {
119         config.setFileFragmentationLevel(fileFragmentationLevel);
120     }
121
122     public void setGenerateDependsOnGroups(boolean generateDependsOnGroups) {
123         config.setGenerateDependsOnGroups(generateDependsOnGroups);
124     }
125
126     public void setGenerateDependsOnMethods(boolean generateDependsOnMethods) {
127         config.setGenerateDependsOnMethods(generateDependsOnMethods);
128     }
129
130     public void setGenerateGroupsAttribute(boolean generateGroupsAttribute) {
131         config.setGenerateGroupsAttribute(generateGroupsAttribute);
132     }
133
134     public void setOutputDirectory(String outputDirectory) {
135         config.setOutputDirectory(outputDirectory);
136     }
137
138     public void setSplitClassAndPackageName(boolean splitClassAndPackageName) {
139         config.setSplitClassAndPackageName(splitClassAndPackageName);
140     }
141
142     public void setStackTraceOutputMethod(int stackTraceOutputMethod) {
143         config.setStackTraceOutputMethod(stackTraceOutputMethod);
144     }
145
146     public void setTimestampFormat(String timestampFormat) {
147         config.setTimestampFormat(timestampFormat);
148     }
149
150     private void writePopBuildInfoForSuite(ISuite suite, XMLStringBuffer xmlBuffer) {
151         Properties prop = new Properties();
152         prop.put("user", System.getProperty("user.name"));
153         try {
154             prop.put("host", InetAddress.getLocalHost().getCanonicalHostName());
155         } catch (UnknownHostException e) {
156         }
157
158         // List<BugForTestMethodIgnore> ignoreList =
159         // AbstractTest.getMethodIgnoreList();
160
161         // Set<ITestResult> failedTests = testResult.getFailedTests().getAllResults();
162         // for (ITestResult testResult : failedTests) {
163         // Method method = testResult.getMethod();
164         // for (BugForTestMethodIgnore ignoreMethod : ignoreList) {
165         // if (ignoreMethod.getMethodSignature().equals(method.getMethodSignature()) ||
166         // ignoreMethod.getMethodName().equals(method.getMethodName()))
167         //
```

```

173     // prop.put("build_id", ContextManager.getGlobalContext().getAttribute(Context.SELENIUM_HUB));
174     prop.put("pool", ContextManager.getGlobalContext().getPool());
175     // prop.put("run_id", String.valueOf(ContextManager.getGlobalContext().getRunId()));
176     // ContextManager.getGlobalContext().getAttribute(Context.RUN_ID);
177     // prop.put("translate", "true");
178     // prop.put("build_id", (String) ContextManager.getGlobalContext().getAttribute(Context.BUILD_ID));
179     // prop.put("hub", ContextManager.getGlobalContext().getAttribute(Context.BUILD_TAG));
180     // prop.put("pool", ContextManager.getGlobalContext().getPool());
181     // prop.put("run_id", String.valueOf(ContextManager.getGlobalContext().getRunId()));
182     xmlBuffer.push("bugfor-run-info", prop);
183     xmlBuffer.push("config-spec");
184     // xmlBuffer.addData("String");
185     // ContextManager.getGlobalContext().getAttribute(Context.BUILD_CONFIG_SPEC);
186     // ContextManager.getGlobalContext().pop();
187     xmlBuffer.pop();
188 }
189 }
190 }
191 }
192 }
193 /**
194 * private void writeReportOutput(XMLStringBuffer xmlBuffer) {
195 * // TODO: Consider - maybe a <line> element isn't indicated for each line.
196 * // Also escaping might be considered
197 * xmlBuffer.push(XMLReporterConfig.TAG_REPORTER_OUTPUT);
198 * List<String> output = reporterConfig.getOutput();
199 * for (String line : output) {
200 *     xmlBuffer.push(XMLReporterConfig.TAG_LINE);
201 *     xmlBuffer.addData(line);
202 *     xmlBuffer.pop();
203 * }
204 * xmlBuffer.pop();
205 }
206 /**
207 * private void writeSuite(XmlSuite xmlSuite, ISuite suite) {
208 * switch (config.getFileFragmentationLevel()) {
209 * case XMLReporterConfig.FF_LEVEL_NONE:
210 *     writeSuiteToBuffer(xmlBuffer, suite);
211 *     break;
212 * case XMLReporterConfig.FF_LEVEL_SUITE:
213 * case XMLReporterConfig.FF_LEVEL_SUITE_RESULT:
214 *     File suitefile = referenceSuite(rootBuffer, suite);
215 *     writeSuiteToFile(suitefile, suite);
216 * }
217 }
218 }
219 /**
220 * private void writeSuiteGroups(XMLStringBuffer xmlBuffer, ISuite suite) {
221 * xmlBuffer.push(XMLReporterConfig.TAG_GROUPS);
222 * Map<String, Collection<ITestNGMethod>> methodsByGroups = suite.getMethodsByGroups();
223 * for (String groupName : methodsByGroups.keySet()) {
224 *     Properties groupAttrs = new Properties();
225 *     groupAttrs.setProperty(XMLReporterConfig.ATTR_NAME, groupName);
226 *     xmlBuffer.push(XMLReporterConfig.TAG_GROUP, groupAttrs);
227 *     Set<ITestNGMethod> groupMethods = getUniqueMethodSet(methodsByGroups.get(groupName));
228 *     for (ITestNGMethod groupMethod : groupMethods) {
229 *         Properties methodAttrs = new Properties();
230 *         methodAttrs.setProperties(groupMethod.getAnnotations());
231 *         methodAttrs.setProperty(XMLReporterConfig.ATTR_METHOD_SIG, groupMethod.toString());
232 *         methodAttrs.setProperty(XMLReporterConfig.ATTR_CLASS, groupMethod.getClassName());
233 *         xmlBuffer.addEmptyElement(XMLReporterConfig.TAG_METHOD, methodAttrs);
234 *     }
235 *     xmlBuffer.pop();
236 * }
237 * xmlBuffer.pop();
238 }
239 */
240 /**
241 * private void writeSuiteToBuffer(XMLStringBuffer xmlBuffer, ISuite suite) {
242 * for (String line : output) {
243 *     xmlBuffer.push(XMLReporterConfig.TAG_LINE);
244 *     xmlBuffer.addData(line);
245 *     xmlBuffer.pop();
246 * }
247 * xmlBuffer.pop();
248 }
249 /**
250 * private void writeSuite(XmlSuite xmlSuite, ISuite suite) {
251 * switch (config.getFileFragmentationLevel()) {
252 * case XMLReporterConfig.FF_LEVEL_NONE:
253 *     writeSuiteToBuffer(xmlBuffer, suite);
254 *     break;
255 * case XMLReporterConfig.FF_LEVEL_SUITE:
256 * case XMLReporterConfig.FF_LEVEL_SUITE_RESULT:
257 *     File suitefile = referenceSuite(rootBuffer, suite);
258 *     writeSuiteToFile(suitefile, suite);
259 * }
260 }
261 /**
262 * private void writeSuiteToFile(File suitefile, ISuite suite) {
263 * XMLStringBuffer xmlWriter = new XMLStringBuffer("<*>");
264 * WriteResultWriter suiteResultWriter = new XMLSuiteResultWriter(config);
265 * for (Map.Entry<String, ISuiteResult> result : results.entrySet()) {
266 *     suiteResultWriter.writeSuiteResult(xmlWriter, result.getValue());
267 * }
268 * xmlWriter.pop();
269 }
270 */
271 /**
272 * private void writeSuiteToFile(File suitefile, ISuite suite) {
273 * XMLStringBuffer xmlWriter = new XMLStringBuffer("<*>");
274 * WriteResultWriter suiteResultWriter = new XMLSuiteResultWriter(config);
275 * WriteResultWriter parentDirWriter = new WriteResultWriter(suitefile.getParentFile());
276 * File parentDir = suitefile.getParentFile();
277 * if (parentDir.exists() || suitefile.getParentFile().mkdirs()) {
278 *     Utils.writeFile(parentDir.getAbsolutePath(), "testing-results.xml", xmlWriter.toXML());
279 * }
280 * }
281 }
282 }

```

```

199     for (String line : output) {
200         xmlBuffer.push(XMLReporterConfig.TAG_LINE);
201         xmlBuffer.addData(line);
202         xmlBuffer.pop();
203     }
204     xmlBuffer.pop();
205 }
206 /**
207 * private void writeSuite(XmlSuite xmlSuite, ISuite suite) {
208 * switch (config.getFileFragmentationLevel()) {
209 * case XMLReporterConfig.FF_LEVEL_NONE:
210 *     writeSuiteToBuffer(xmlBuffer, suite);
211 *     break;
212 * case XMLReporterConfig.FF_LEVEL_SUITE:
213 * case XMLReporterConfig.FF_LEVEL_SUITE_RESULT:
214 *     File suitefile = referenceSuite(rootBuffer, suite);
215 *     writeSuiteToFile(suitefile, suite);
216 * }
217 }
218 }
219 /**
220 * private void writeSuiteGroups(XMLStringBuffer xmlBuffer, ISuite suite) {
221 * xmlBuffer.push(XMLReporterConfig.TAG_GROUPS);
222 * Map<String, Collection<ITestNGMethod>> methodsByGroups = suite.getMethodsByGroups();
223 * for (String groupName : methodsByGroups.keySet()) {
224 *     Properties groupAttrs = new Properties();
225 *     groupAttrs.setProperty(XMLReporterConfig.ATTR_NAME, groupName);
226 *     xmlBuffer.push(XMLReporterConfig.TAG_GROUP, groupAttrs);
227 *     Set<ITestNGMethod> groupMethods = getUniqueMethodSet(methodsByGroups.get(groupName));
228 *     for (ITestNGMethod groupMethod : groupMethods) {
229 *         Properties methodAttrs = new Properties();
230 *         methodAttrs.setProperty(XMLReporterConfig.ATTR_NAME, groupMethod.getMethodName());
231 *         methodAttrs.setProperty(XMLReporterConfig.ATTR_METHOD_SIG, groupMethod.toString());
232 *         methodAttrs.setProperty(XMLReporterConfig.ATTR_CLASS, groupMethod.getClassName());
233 *         xmlBuffer.addEmptyElement(XMLReporterConfig.TAG_METHOD, methodAttrs);
234 *     }
235 *     xmlBuffer.pop();
236 * }
237 * xmlBuffer.pop();
238 }
239 /**
240 * private void writeSuiteToBuffer(XMLStringBuffer xmlBuffer, ISuite suite) {
241 * xmlBuffer.push(XMLReporterConfig.TAG_SUITE, getSuiteAttributes(suite));
242 * writePoolBuildInformation(suite, rootBuffer);
243 * writeSuiteGroups(xmlBuffer, suite);
244 * Map<String, ISuiteResults> results = suite.getResults();
245 * XMLSuiteResultWriter suiteResultWriter = new XMLSuiteResultWriter(config);
246 * for (Map.Entry<String, ISuiteResult> result : results.entrySet()) {
247 *     suiteResultWriter.writeSuiteResult(xmlBuffer, result.getValue());
248 * }
249 * xmlBuffer.pop();
250 }
251 */
252 /**
253 * private void writeSuiteToFile(File suitefile, ISuite suite) {
254 * XMLStringBuffer xmlWriter = new XMLStringBuffer("<*>");
255 * WriteResultWriter suiteResultWriter = new XMLSuiteResultWriter(config);
256 * WriteResultWriter parentDirWriter = new WriteResultWriter(suitefile.getParentFile());
257 * File parentDir = suitefile.getParentFile();
258 * if (parentDir.exists() || suitefile.getParentFile().mkdirs()) {
259 *     Utils.writeFile(parentDir.getAbsolutePath(), "testing-results.xml", xmlWriter.toXML());
260 * }
261 * }
262 }

```

PLUGIN:

```

28  * Java class for anonymous complex type.
29  *
30  * <!--
31  * The following schema fragment specifies the expected content contained within
32  * this class.
33  *
34  * <!--
35  * <!-->
36  * <!-->
37  * </pre>
38  *
39  */
40 @XmlAccessorType(XmlAccessType.FIELD)
41 @XmlElement(name = "plugin")
42 @XmlRootElement(name = "mountplugins")
43 public class Mauiplugins {
44     @XmlElement(required = true)
45     protected List<Plugin> plugin;
46
47     /**
48      * Gets the value of the plugin property.
49      *
50      * <p>
51      * This accessor method returns a reference to the live list, not a
52      * snapshot. Therefore any modification you make to the returned list will
53      * be present inside the JAXB object. This is why there is not a
54      * <CODE>set</CODE> method for the plugin property.
55      *
56      * <p>
57      * For example, to add a new item, do as follows:
58      *
59      * <pre>
60      *     getPlugin().add(newItem);
61      * </pre>
62      *
63      * Objects of the following type(s) are allowed in the list {@link Plugin }
64      *
65      */
66     public List<Plugin> getPlugin() {
67         if (plugin == null) {
68             plugin = new ArrayList<Plugin>();
69         }
70         return this.plugin;
71     }
72 }

```

Members calling constructors of 'WebPageSection' - in workspace

```

28  * Java class for anonymous complex type.
29  *
30  * <!--
31  * The following schema fragment specifies the expected content contained within
32  * this class.
33  *
34  * <!--
35  * <!-->
36  * <!-->
37  * </pre>
38  *
39  */
40 @XmlAccessorType(XmlAccessType.FIELD)
41 @XmlElement(name = "plugin")
42 @XmlRootElement(name = "mountplugins")
43 public class Mauiplugins {
44     @XmlElement(required = true)
45     protected List<Plugin> plugin;
46
47     /**
48      * Gets the value of the plugin property.
49      *
50      * <p>
51      * This accessor method returns a reference to the live list, not a
52      * snapshot. Therefore any modification you make to the returned list will
53      * be present inside the JAXB object. This is why there is not a
54      * <CODE>set</CODE> method for the plugin property.
55      *
56      * <p>
57      * For example, to add a new item, do as follows:
58      *
59      * <pre>
60      *     getPlugin().add(newItem);
61      * </pre>
62      *
63      * Objects of the following type(s) are allowed in the list {@link Plugin }
64      *
65      */
66     public List<Plugin> getPlugin() {
67         if (plugin == null) {
68             plugin = new ArrayList<Plugin>();
69         }
70         return this.plugin;
71     }
72 }

```

Members calling constructors of 'WebPageSection' - in workspace

The screenshot shows the Java code for the `Method` class. The code includes annotations for XML accessors and methods, and it defines a protected list of `Page` objects and a private string `name`. The `getPage()` method returns a reference to the live list, while the `setPage()` method sets the value of the `name` property.

```
37 * @link{complexType}
38 * </pre>
39 *
40 */
41
42 @XmlAccessorType(XmlAccessType.FIELD)
43 @XmlElement(name = "method")
44 @XmlType(propOrder = {"page"})
45 public class Method {
46
47     protected List<Page> page;
48     @XmlAttribute(required = true)
49     protected String name;
50
51     /**
52      * Gets the value of the name property.
53      *
54      * @return possible object is {@link String }
55      */
56
57     public String getName() {
58         return name;
59     }
60
61     /**
62      * Gets the value of the page property.
63      *
64      * <p>
65      * This accessor method returns a reference to the live list, not a
66      * snapshot. Therefore any modification you make to the returned list will
67      * be present inside the JAXB object. This is why there is not a
68      * <CODE>set</CODE> method for the page property.
69      *
70      * <p>
71      * For example, to add a new item, do as follows:
72      *
73      * <pre>
74      * getPage().add(newItem);
75      * </pre>
76      *
77      * <p>
78      * Objects of the following type(s) are allowed in the list {@link Page }
79      *
80      */
81
82     public List<Page> getPage() {
83         if (page == null) {
84             page = new ArrayList<Page>;
85         }
86         return this.page;
87     }
88
89     /**
90      * Sets the value of the name property.
91      *
92      * @param value
93      *          allowed object is {@link String }
94      */
95
96     public void setName(String value) {
97         this.name = value;
98     }
99
100 }
101 }
```

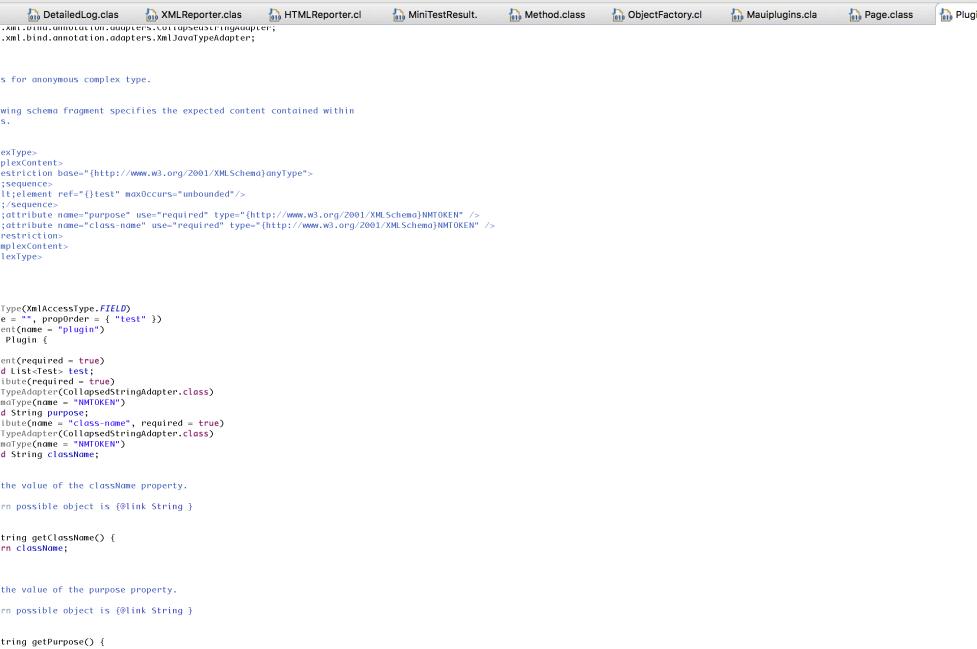
The screenshot shows the Java code for the `ObjectFactory` class. It contains factory methods for creating various objects such as `Mauiplugins`, `Method`, `Page`, `Plugin`, and `Test`.

```
0 package com.ebay.mauil.reporter.pluginmodel;
1
2 import javax.xml.bind.annotation.XmlRegistry;
3
4 /**
5  * This object contains Factory methods for each Java content interface and Java
6  * element interface generated in the generated package.
7  *
8  * An ObjectFactory allows you to programmatically construct new instances of the
9  * Java representation for XML content. The Java representation of XML content
10 * can consist of schema derived interfaces and classes representing the binding
11 * or schema type definitions, element declarations and model groups. Factory
12 * methods for each of these are provided in this class.
13 */
14
15 @XmlRegistry
16 public class ObjectFactory {
17
18     /**
19      * Create a new ObjectFactory that can be used to create new instances of
20      * schema derived classes for package: generated
21      */
22
23     public ObjectFactory() {
24
25     }
26
27     /**
28      * Create an instance of {@link Mauiplugins }
29      */
30     public Mauiplugins createMauiplugins() {
31         return new Mauiplugins();
32     }
33
34     /**
35      * Create an instance of {@link Method }
36      */
37     public Method createMethod() {
38         return new Method();
39     }
40
41     /**
42      * Create an instance of {@link Page }
43      */
44     public Page createPage() {
45         return new Page();
46     }
47
48     /**
49      * Create an instance of {@link Plugin }
50      */
51     public Plugin createPlugin() {
52         return new Plugin();
53     }
54
55     /**
56      * Create an instance of {@link Test }
57      */
58     public Test createTest() {
59         return new Test();
60     }
61 }
```



The screenshot shows an IDE interface with a toolbar at the top containing various icons for file operations, search, and debugging. The left sidebar displays a project structure with files like `RaiseClaimTestPlan.java`, `DetailedLog.class`, `XMLReporter.class`, `HTMLReporter.class`, `MiniTestResult.class`, `Method.class`, `ObjectFactory.class`, `Mailplugins.class`, and `Page.class`. The main editor area contains Java code for XML binding, specifically for a class named `Page`. The code includes annotations such as `@XmlAccessorType`, `@XmlElement`, and `@XmlAttribute`. The code is annotated with Javadoc-style comments explaining the schema fragment and the properties of the `Page` class.

```
2 * This file was generated by the JaxBIM Architecture For XML Binding(JAXB) Reference Implementation, vJAXB 2.1.10 in JDK 6
3 *
4 * package com.ebay.mapi.reporter.pluginmodel;
5 *
6 * import javax.xml.bind.annotation.XmlAccessType;
7 * import javax.xml.bind.annotation.XmlAccessorType;
8 * import javax.xml.bind.annotation.XmlElement;
9 * import javax.xml.bind.annotation.XmlRootElement;
10 * import javax.xml.bind.annotation.XmlType;
11 *
12 */
13 *
14 /**
15 * <p>
16 * Java class for anonymous complex type.
17 * </p>
18 * <p>
19 * The following schema fragment specifies the expected content contained within
20 * this class.
21 *
22 * <pre>
23 * <xsd:complexType>
24 *   <xsd:complexContent>
25 *     <xsd:restriction base="http://www.w3.org/2001/XMLSchema:anyType">
26 *       <xsd:attribute name="class-name" use="required" type="http://www.w3.org/2001/XMLSchema:string" />
27 *     </xsd:restriction>
28 *   </xsd:complexContent>
29 * </xsd:complexType>
30 * </pre>
31 */
32 *
33 /**
34 * @XmlAccessorType(XmlAccessType.FIELD)
35 * @XmlType(name = "")
36 * @XmlRootElement(name = "page")
37 */
38 public class Page {
39
40     @XmlAttribute(name = "class-name", required = true)
41     protected String className;
42
43     /**
44      * Gets the value of the className property.
45      *
46      * @return possible object is {@link String }
47      */
48
49     /**
50      * Returns a string representation of the object. For detailed
51      * information on this method, see http://www.w3.org/TR/2004/REC-xml-20040208/#cdata-sections
52      */
53     /**
54      * Sets the value of the className property.
55      *
56      * @param value
57      *     allowed object is {@link String }
58      */
59
60     /**
61      * Sets the value of the className property.
62      *
63      * @param value
64      */
65     public void setClassName(String value) {
66         this.className = value;
67     }
68 }
69 }
```



The screenshot shows the Eclipse IDE interface with the Java EE perspective selected. The top menu bar includes 'File', 'Edit', 'Select', 'Run', 'Debug', 'View', 'Search', 'Help', and 'Quick Access'. The toolbar contains icons for file operations like Open, Save, Cut, Copy, Paste, Find, and Run. The left sidebar has 'Project Explorer' and 'Outline' tabs. The main editor area displays Java code for a class named 'Plugin'. The code includes annotations for XML schema mapping, such as @XmlAccessorType(FIELD), @XmlAttribute(name = "test"), and @XmlElement(name = "plugin"). It also features several protected methods for getting properties like 'className' and 'purpose'. The code is color-coded for syntax highlighting.

```
1 package com.mau.plugin;
2 import java.util.List;
3 import javax.xml.bind.annotation.adapters.CollapsedStringAdapter;
4 import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;
5 import javax.xml.bind.annotation.adapters.XmlType;
6
7 /**
8  * Java class for anonymous complex type.
9  *
10 * The following schema fragment specifies the expected content contained within
11 * this class.
12 *
13 * <pre>
14 *   <complexType>
15 *     <complexContent>
16 *       <restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
17 *         <sequence>
18 *           <element ref="{}test" maxOccurs="unbounded"/>
19 *           <element name="purpose" type="string" />
20 *           <element name="class-name" type="string" />
21 *         </sequence>
22 *       </restriction>
23 *     </complexContent>
24 *   </complexType>
25 * </pre>
26 *
27 * This class was generated by JBoss Seam 2.1.0.Final.
28 */
29
30 package com.mau.plugin;
31
32 import javax.xml.bind.annotation.*;
33
34 /**
35  * Java class for anonymous complex type.
36  *
37  * The following schema fragment specifies the expected content contained within
38 * this class.
39 *
40 * <pre>
41 *   <complexType>
42 *     <complexContent>
43 *       <restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
44 *         <sequence>
45 *           <element ref="{}test" maxOccurs="unbounded"/>
46 *           <element name="purpose" type="string" />
47 *           <element name="class-name" type="string" />
48 *         </sequence>
49 *       </restriction>
50 *     </complexContent>
51 *   </complexType>
52 * </pre>
53 */
54
55 /**
56  * Java class for the <code>test</code> element.
57  *
58  * This class was generated by JBoss Seam 2.1.0.Final.
59 */
60
61 package com.mau.plugin;
62
63 /**
64  * Gets the value of the test property.
65  *
66  * @return possible object is @link String
67  */
68
69 public String getTest() {
70     return test;
71 }
72
73 /**
74  * Gets the value of the className property.
75  *
76  * @return possible object is @link String
77  */
78
79 public String getClassName() {
80     return className;
81 }
82
83 /**
84  * Gets the value of the purpose property.
85  *
86  * @return possible object is @link String
87  */
88
89 public String getPurpose() {
90     return purpose;
91 }
92
93 /**
94  * Gets the value of the test property.
95  */
96
```

```
70     }
71     return className;
72 }
73 /**
74 * Gets the value of the purpose property.
75 *
76 * @return possible object is {@link String }
77 */
78 public String getPurpose() {
79     return purpose;
80 }
81 }
82 /**
83 * Gets the value of the test property.
84 *
85 * <p>
86 * This accessor method returns a reference to the live list, not a
87 * snapshot. Therefore any modification you make to the returned list will
88 * be present inside the JAXB object. This is why there is not a
89 * <CODE>set</CODE> method for the test property.
90 * </p>
91 * For example, to add a new item, do as follows:
92 *
93 * <pre>
94 *     getTest().add(newItem);
95 * </pre>
96 *
97 */
98 public List<Test> getTest() {
99     if (test == null) {
100         test = new ArrayList<Test>();
101     }
102     return test;
103 }
104 /**
105 * Sets the value of the className property.
106 *
107 * @param value
108 *     allowed object is {@link String }
109 */
110 public void setClassName(String value) {
111     this.className = value;
112 }
113 /**
114 * Sets the value of the purpose property.
115 *
116 * @param value
117 *     allowed object is {@link String }
118 */
119 public void setPurpose(String value) {
120     this.purpose = value;
121 }
122 }
123 /**
124 * Sets the value of the test property.
125 *
126 * @param value
127 *     allowed object is {@link Test }
128 */
129 public void setTest(List<Test> value) {
130     this.test = value;
131 }
132 }
133 }
134 }
```

Read-Only Smart Insert 2:1

```
16 import javax.xml.bind.annotation.XmlRootElement;
17 import javax.xml.bind.annotation.XmlType;
18 /**
19 * <p>
20 * Java class for anonymous complex type.
21 *
22 * <p>
23 * The following schema fragment specifies the expected content contained within
24 * this class.
25 *
26 * <pre>
27 * <complexType>
28 *   <complexContent>
29 *     <restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
30 *       <sequence>
31 *         <element ref="{}page" maxOccurs="unbounded" minOccurs="0"/>
32 *         <element ref="{}method" maxOccurs="unbounded" minOccurs="0"/>
33 *         <element ref="{}test" maxOccurs="unbounded" minOccurs="0"/>
34 *       </sequence>
35 *       <attribute name="class-name" use="required" type="{}string" />
36 *     </restriction>
37 *   </complexContent>
38 * </complexType>
39 * </pre>
40 *
41 */
42 @XmlAccessorType(XmlAccessType.FIELD)
43 @XmlAttribute(name = "", propOrder = { "page", "method" })
44 @XmlRootElement(name = "test")
45 public class Test {
46
47     protected List<Page> page;
48     protected List<Method> method;
49     @XmlAttribute(name = "class-name", required = true)
50     protected String className;
51
52     /**
53      * Gets the value of the className property.
54      *
55      * @return possible object is {@link String }
56      */
57     public String getClassName() {
58         return className;
59     }
60
61     /**
62      * Gets the value of the method property.
63      *
64      * <p>
65      * This accessor method returns a reference to the live list, not a
66      * snapshot. Therefore any modification you make to the returned list will
67      * be present inside the JAXB object. This is why there is not a
68      * <CODE>set</CODE> method for the method property.
69      * </p>
70      * For example, to add a new item, do as follows:
71      *
72      * <pre>
73      *     getMethod().add(newItem);
74      * </pre>
75      */
76     public List<Method> getMethod() {
77         return method;
78     }
79
80     /**
81      * Objects of the following type(s) are allowed in the list {@link Method }
82      */
83 }
```

Read-Only Smart Insert 2:1

The screenshot shows a Java IDE interface with a code editor containing Java code. The code is annotated with Javadoc-style comments. The editor has a toolbar at the top and a status bar at the bottom.

```
68 * snapshot. Therefore any modification you make to the returned list will
69 * be present inside the JAXB object. This is why there is not a
70 * <CODE>set</CODE> method for the method property.
71 *
72 * <p>
73 * For example, to add a new item, do as follows:
74 *
75 * <pre>
76 *     * getMethod().add(newItem);
77 * </pre>
78 *
79 * <p>
80 * Objects of the following type(s) are allowed in the list {@link Method }
81 *
82 *
83 */
84 @public List<Method> getMethod() {
85     if (method == null)
86         method = new ArrayList<Method>();
87     return this.method;
88 }
89 /**
90 * Gets the value of the page property.
91 *
92 * <p>
93 * This accessor method returns a reference to the live list, not a
94 * snapshot. Therefore any modification you make to the returned list will
95 * be present inside the JAXB object. This is why there is not a
96 * <CODE>set</CODE> method for the page property.
97 *
98 * <p>
99 * For example, to add a new item, do as follows:
100 *
101 * <pre>
102 *     * getPage().add(newItem);
103 * </pre>
104 *
105 * <p>
106 * Objects of the following type(s) are allowed in the list {@link Page }
107 *
108 */
109 @public List<Page> getPage() {
110     if (page == null)
111         page = new ArrayList<Page>();
112     return this.page;
113 }
114 /**
115 * Sets the value of the className property.
116 *
117 * @param value
118 *     allowed object is {@link String }
119 */
120 @public void setClassName(String value) {
121     this.className = value;
122 }
123 }
```

Read-Only | Smart Insert | 2 : 1