RIDE File Edit Source Refactoring Navigate Search Project Run eBay Window Help 63% Thu 1:28:10 AM Shukla, Gaurav(AWF)

Java EE - com.ebay.maui.util.APIUtil - RIDE for Raptor 2 - /Users/gaurshukla/Documents/Drivers/Workspace_MAUI_08_18

Quick Access | Java EE | Debug

Projec | Navig | Type

- Maven Dependencies
  - CartServiceCommonTypeLibraryV4-4.0.4
  - mauiutil-1.1.6.jar - /Users/gaurshukla/.m
    - com.ebay.maui.util
      - APIUtil.class
        - APIUtil
      - ArrayUtil.class
      - AssertionUtil.class
      - AxisObjectUtil.class
      - CSSUtil.class
      - CSVUtil.class
      - DateUtil.class
      - DBUtil.class
      - EscapeHelper.class
      - ExpectedConditionsUtil.class
      - FileTransferAPIHelper.class
      - FileUtil.class
      - GrepUtil.class
      - HttpUtil.class
      - JsonDataProviderUtil.class
      - MetaDataUtil.class
      - ProtobufMessageBodyReaderWrite
      - ProtostuffMessageBodyReaderWrit
      - ReflectUtil.class
      - RESTAPIUtil.class
      - RESTClient.class
      - RESTResponse.class
      - SiteUtil.class
      - SpreadSheetUtil.class
      - SSHUtil.class
      - StringInputStream.class
      - StringUtil.class
      - SystemUtil.class
      - WebUXUtil.class
      - XmlUtil.class
      - XPathUtil.class
    - com.ebay.maui.util.accessibility
    - com.ebay.maui.util.htmlsource
    - com.ebay.maui.util.internal.api
    - com.ebay.maui.util.internal.bes

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```
8  t java.io.StringWriter;
9  t java.lang.reflect.Method;
10 t java.net.URL;
11 t java.util.Collections;
12 t java.util.HashMap;
13 t java.util.Iterator;
14 t java.util.Map;
15 t java.util.Properties;
16
17 t javax.xml.parsers.DocumentBuilder;
18 t javax.xml.parsers.DocumentBuilderFactory;
19 t javax.xml.soap.SOAPElement;
20
21 t org.apache.axis.message.SOAPHeaderElement;
22 t org.apache.log4j.Logger;
23 t org.apache.log4j.Priority;
24 t org.apache.xerces.parsers.DOMParser;
25 t org.apache.xml.serialize.OutputFormat;
26 t org.apache.xml.serialize.XMLSerializer;
27 t org.apache.xpath.XPathAPI;
28 t org.w3c.dom.CharacterData;
29 t org.w3c.dom.Document;
30 t org.w3c.dom.Element;
31 t org.w3c.dom.Node;
32 t org.w3c.dom.NodeList;
33 t org.xml.sax.InputSource;
34
35 t com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.CSGetTokenRequestType;
36 t com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.CSGetTokenResponseType;
37 t com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.EBayCSAPISoapBindingStub;
38 t com.ebay.maui.controller.Assertion;
39 t com.ebay.maui.controller.Context;
40 t com.ebay.maui.controller.ContextManager;
41 t com.ebay.maui.controller.Logging;
42 t com.ebay.maui.driver.api.APIDriver;
43 t com.ebay.maui.driver.api.APIRequest;
44 t com.ebay.maui.exception.APIException;
45 t com.ebay.maui.exception.XMLException;
46 t com.ebay.maui.util.internal.context.APIContext;
```

Outline | Task Li

- APIUtil
  - APIUtil()
  - logger : Logger
  - ebayTokenMap : Map<String
  - getAPIContext() : APIContex
  - convertDocumentAsFormatS
  - convertDocumentAsFormatS
  - convertStrAsDocument(Strir
  - deleteXMLElementByConten
  - deleteXMLElementByXPath(S
  - executeAPICall(String, Prope
  - executeAPICallAndVerifyAck
  - executeAPICallAndVerifyRes
  - executeAPICallAndVerifyRes
  - executeAPIRequest(Abstract
  - executeAPIRequest(Abstract
  - executeAPIRequest(String, A
  - executeAPIRequest(String, A
  - executeCS1APIRequest(Strir
  - executeCSAPIRequest(Abstr
  - executeCSAPIRequest(String
  - executeCSAPIRequest(String
  - executeCSAPIRequest(String
  - executeCSAPIRequest(String
  - executeRTMAPIRequest(Stri
  - executeSchemaAPIRequest(

Markers | Properties | Servers | Data Source Explorer | Snippets | Console | Progress | Results of running class RaiseClaimTestPlan | Package Explorer | Call Hierarchy

No operations to display at this time.

1 Reminder

V3 to Raptor Migration Office Hours
01 / 1.2.012 DATABASE (8) P VC/ Dial-in: 1-866-27...
In 2 minutes

Snooze | Dismiss

Read-Only | Smart Insert

Quick Access | Java EE | Debug

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```
1  package com.ebay.maui.util;
2
3  import java.io.BufferedInputStream;
4  import java.io.File;
5  import java.io.FileInputStream;
6  import java.io.IOException;
7  import java.io.StringReader;
8  import java.io.StringWriter;
9  import java.lang.reflect.Method;
10 import java.net.URL;
11 import java.util.Collections;
12 import java.util.HashMap;
13 import java.util.Iterator;
14 import java.util.Map;
15 import java.util.Properties;
16
17 import javax.xml.parsers.DocumentBuilder;
18 import javax.xml.parsers.DocumentBuilderFactory;
19 import javax.xml.soap.SOAPElement;
20
21 import org.apache.axis.message.SOAPHeaderElement;
22 import org.apache.log4j.Logger;
23 import org.apache.log4j.Priority;
24 import org.apache.xerces.parsers.DOMParser;
25 import org.apache.xml.serialize.OutputFormat;
26 import org.apache.xml.serialize.XMLSerializer;
27 import org.apache.xpath.XPathAPI;
28 import org.w3c.dom.CharacterData;
29 import org.w3c.dom.Document;
30 import org.w3c.dom.Element;
31 import org.w3c.dom.Node;
32 import org.w3c.dom.NodeList;
33 import org.xml.sax.InputSource;
34
35 import com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.CSGetTokenRequestType;
36 import com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.CSGetTokenResponseType;
37 import com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.EBayCSAPISoapBindingStub;
38 import com.ebay.maui.controller.Assertion;
39 import com.ebay.maui.controller.Context;
40 import com.ebay.maui.controller.ContextManager;
41 import com.ebay.maui.controller.Logging;
42 import com.ebay.maui.driver.api.APIDriver;
43 import com.ebay.maui.driver.api.APIRequest;
44 import com.ebay.maui.exception.APIException;
45 import com.ebay.maui.exception.XMLException;
46 import com.ebay.maui.util.internal.context.APIContext;
47 import com.ebay.maui.util.internal.type.Site;
48 import com.ebay.qa.xmldog.Config;
49 import com.ebay.sdk.ApiAccount;
50 import com.ebay.sdk.ApiCall;
51 import com.ebay.sdk.ApiContext;
52 import com.ebay.sdk.ApiCredential;
53 import com.ebay.sdk.eBayAccount;
54 import com.ebay.soap.eBLBaseComponents.AbstractRequestType;
55 import com.ebay.soap.eBLBaseComponents.AbstractResponseType;
56
57 @SuppressWarnings("deprecation")
58 public class APIUtil {
59     private static Logger logger = Logging.getLogger(APIUtil.class);
60     private static final Map<String, String> ebayTokenMap = Collections.synchronizedMap(new HashMap<String, String>());
61     private static APIContext getAPIContext(){
62         APIContext apiContext = (APIContext)ContextManager.getThreadContext().getApiContext();
63         if(null==apiContext){
64             Context context = ContextManager.getThreadContext();
65             apiContext = new APIContext();
66             apiContext.load(context.getTestNGContext(), context);
67             apiContext = (APIContext)ContextManager.getThreadContext().getApiContext();
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
            apiContext.fromQueryContext(getHttpQueryContext(), context);
            apiContext = (APIContext)ContextManager.getThreadContext().getApiContext();
        }

        return apiContext;
    }
    /**
     * Convert the XML Document as String
     *
     * @param doc
     * @return
     * @throws IOException
     */
    public static String convertDocumentAsFormatStr(Document doc) throws IOException {
        return convertDocumentAsFormatStr(doc, System.getProperty("line.separator"));
    }

    /**
     * Convert the XML Document as String
     *
     * @param doc
     * @param lineSeparator
     * @return
     * @throws IOException
     */
    public static String convertDocumentAsFormatStr(Document doc, String lineSeparator) throws IOException {
        OutputFormat format = new OutputFormat(doc);
        format.setPreserveSpace(false);
        format.setIndent(2);
        format.setLineSeparator(lineSeparator);
        StringWriter stringOut = new StringWriter();
        XMLSerializer serializer = new XMLSerializer(stringOut, format);
        serializer.serialize(doc);
        return stringOut.toString();
    }

    /**
     * Convert String as XML Document
     *
     * @param input
     * @param nameSpaceSupport
     * @return
     * @throws Exception
     */
    public static Document convertStrAsDocument(String input, boolean nameSpaceSupport) throws Exception {
        StringReader stringStream = new StringReader(input);
        InputSource source = new InputSource(stringStream);
        DOMParser parser = new DOMParser();
        parser.setFeature("http://xml.org/sax/features/namespaces", nameSpaceSupport);
        parser.parse(source);
        Document doc = parser.getDocument();
        stringStream.close();
        return doc;
    }

    /**
     * <p>
     * Delete the xml element by specified xml tag name.
     * </p>
     * <p>
     * Example: when you call this method with (xmlTagName="Username")
     * </p>
     * <p>
     * Input:
     * </p>
     *
     * <pre>
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
     * <pre>
     * {@code
     * <?xml version="1.0" encoding="utf-8"?>
     * <GeteBayOfficialTimeRequest xmlns="urn:ebay:apis:eBLBaseComponents">
     *   <Version>547</Version>
     *   <RequesterCredentials>
     *     <Username>apitest1</Username>
     *     <Password>ebay</Password>
     *   </RequesterCredentials>
     * </GeteBayOfficialTimeRequest>
     * }
     * </pre>
     * <p>
     * Output:
     * </p>
     *
     * <pre>
     * {@code
     * <?xml version="1.0" encoding="utf-8"?>
     * <GeteBayOfficialTimeRequest xmlns="urn:ebay:apis:eBLBaseComponents">
     *   <Version>547</Version>
     *   <RequesterCredentials>
     *     <Username>apitest1</Username>
     *     <Password>ebay</Password>
     *   </RequesterCredentials>
     * </GeteBayOfficialTimeRequest>
     * }
     * </pre>
     *
     * @param xmlContent
     * @param deletedContent
     * @return
     * @throws Exception
     */
    public static String deleteXMLElementByContent(String xmlContent, String xmlTagName) throws Exception {
        Document doc = convertStrAsDocument(xmlContent, true);
        xmlContent = convertDocumentAsFormatStr(doc, "");
        xmlContent = xmlContent.replaceAll(">[\\s\\t]+<", "><");

        xmlContent = xmlContent.replaceAll("<" + xmlTagName + ".+<\\/" + xmlTagName + ">", "");

        doc = convertStrAsDocument(xmlContent, true);
        return convertDocumentAsFormatStr(doc);
    }

    /**
     * <p>
     * Delete the xml element by specified Xpath.
     * </p>
     * <p>
     * Example: when you call this method with
     * (Xpath="//GeteBayOfficialTimeRequest/RequesterCredentials/Username")
     * </p>
     * <p>
     * Input:
     * </p>
     *
     * <pre>
     * {@code
     * <?xml version="1.0" encoding="utf-8"?>
     * <GeteBayOfficialTimeRequest xmlns="urn:ebay:apis:eBLBaseComponents">
     *   <Version>547</Version>
     *   <RequesterCredentials>
     *     <Username>apitest1</Username>
     *     <Password>ebay</Password>
     *   </RequesterCredentials>
     * </GeteBayOfficialTimeRequest>
     * }
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
     * </GeteBayOfficialTimeRequest>
     * }
     * </pre>
     * <p>
     * Output:
     * </p>
     *
     * <pre>
     * {@code
     * <?xml version="1.0" encoding="utf-8"?>
     * <GeteBayOfficialTimeRequest xmlns="urn:ebay:apis:eBLBaseComponents">
     *   <Version>547</Version>
     *   <RequesterCredentials>
     *     <Password>ebay</Password>
     *   </RequesterCredentials>
     * </GeteBayOfficialTimeRequest>
     * }
     * </pre>
     *
     * @param xmlContent
     * @param xPpathLocation
     * @return
     * @throws Exception
     */
    public static String deleteXMLElementByXPath(String xmlContent, String xPpathLocation) throws Exception {
        DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document doc = builder.parse(new InputSource(new StringReader(xmlContent)));

        Element resolver = doc.createElement("NamespaceResolver");
        resolver.setAttribute("xmlns:m", "http://www.kijiji.com/soap");
        resolver.setAttribute("xmlns:SOAP", "http://www.w3.org/2003/05/soap-envelope");

        Node n = XPathAPI.selectSingleNode(doc, xPpathLocation, resolver);
        Node parent = n.getParentNode();
        parent.removeChild(n);

        return convertDocumentAsFormatStr(doc);
    }

    /**
     * Execute API call
     *
     * @param endPoint
     * @param headers
     * @param xmlInput
     * @param expectedResults
     * @return
     * @throws Exception
     */
    public static String executeAPICall(String endPoint, Properties headers, String xmlInput) throws Exception {
        APIRequest request = new APIRequest(endPoint, null, headers, xmlInput);
        APIDriver driver = new APIDriver(request);
        return driver.executeAPIRequest().getResponseXML();
    }

    /**
     * Execute API call and verify the 'Ack'
     *
     * @param endPoint
     * @param headers
     * @param xmlInput
     * @param expectedOut
     * @return
     * @throws Exception
     */
    public static String executeAPICallAndVerifyAck(String endPoint, Properties headers, String xmlInput, String expectedOut) throws Exception {
        HashMap<String, String> expectedResults = new HashMap<String, String>();
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
        expectedResults.put("Ack", expectedOut);

        return executeAPICallAndVerifyResponse(endPoint, headers, xmlInput, expectedResults);
    }

    /**
     * Execute API call and verify the specified xml tag which is stored in hash
     * map expectedResults.
     *
     * @param endPoint
     * @param headers
     * @param xmlInput
     * @param expectedResults
     * @return
     * @throws Exception
     */
    public static String executeAPICallAndVerifyResponse(String endPoint, Properties headers, String xmlInput, HashMap<String, String> expectedResults)
            throws Exception {
        String responsedXml = executeAPICall(endPoint, headers, xmlInput);

        if (expectedResults != null) {
            Iterator<String> it = expectedResults.keySet().iterator();
            {
                while (it.hasNext()) {
                    String xmlTag = it.next();
                    String expectedValue = expectedResults.get(xmlTag);
                    String realValue = XPathUtil.getXPathContent(responsedXml, "//*[local-name()='" + xmlTag + "']")[0];
                    Assertion.assertEquals(realValue, expectedValue, "API Call didn't get expected Results");
                }
            }
        }

        responsedXml = convertDocumentAsFormatStr(convertStrAsDocument(responsedXml, true));
        return responsedXml;
    }

    /**
     * Execute API call and verify the specified xml tag
     *
     * @param endPoint
     * @param headers
     * @param xmlInput
     * @param xmlTag
     * @param expectedOut
     * @return
     * @throws Exception
     */
    public static String executeAPICallAndVerifyResponse(String endPoint, Properties headers, String xmlInput, String xmlTag, String expectedOut)
            throws Exception {
        HashMap<String, String> expectedResults = new HashMap<String, String>();
        expectedResults.put(xmlTag, expectedOut);

        return executeAPICallAndVerifyResponse(endPoint, headers, xmlInput, expectedResults);
    }

    /**
     * Execute API Request
     *
     * @param request
     * @param account
     * @return responseType
     * @throws APIException
     */
    public static AbstractResponseType executeAPIRequest(AbstractRequestType request, eBayAccount account) throws XMLException, APIException {
        String endPoint = getAPIContext().getSoapEndPoint();
        return executeAPIRequest(endPoint, request, account.getUsername(), account.getPassword());
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
    }

    /**
     * Execute API Request
     *
     * @param request
     * @param userName
     * @param password
     * @return responseType
     * @throws APIException
     */
    public static AbstractResponseType executeAPIRequest(AbstractRequestType request, String userName, String password) throws XMLException, APIException {
        String endPoint = getAPIContext().getSoapEndPoint();
        return executeAPIRequest(endPoint, request, userName, password);
    }

    /**
     * Execute API Request
     *
     * @param endPoint
     * @param request
     * @param userName
     * @param password
     * @return responseType
     * @throws APIException
     */
    public static AbstractResponseType executeAPIRequest(String endPoint, AbstractRequestType request, String userName, String password) throws APIException {
        return executeAPIRequest(endPoint, request, userName, password, null, null);
    }

    /**
     * Execute API Request
     *
     * @param endPoint
     * @param request
     * @param userName
     * @param password
     * @param goldenFile
     * @param xmlCompareConfig
     * @return responseType
     * @throws APIException
     */
    public static AbstractResponseType executeAPIRequest(String endPoint, AbstractRequestType request, String userName, String password, String goldenFile,
            Config xmlCompareConfig) throws APIException {
        ApiCall apiCall = new ApiCall();
        try {
            apiCall.setApiContext(getApiContext(userName, password, endPoint));
        } catch (XMLException e) {
            logger.error("Exception:" + e);
            throw new APIException(e);
        }

        AbstractResponseType responseType = null;
        try {
            responseType = apiCall.execute(request);

        } catch (Exception e) {
            logger.error("Exception:" + e);
            throw new APIException(e);
        } finally {
            APIDriver.generateLogMessage(endPoint, new Properties(), apiCall.getRequestXml(), new Properties(), apiCall.getResponseXml(), goldenFile,
                    xmlCompareConfig);
        }
        return responseType;
    }

    public static String executeCS1APIRequest(String callName, String xmlInput) throws APIException {
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
    public static String executeCS1APIRequest(String callName, String xmlInput) throws APIException {
        Properties parameters = new Properties();
        Context context = ContextManager.getThreadContext();
        APIContext apiContext = getAPIContext();
        parameters.setProperty(APIContext.CS_VERSION, apiContext.getCsVersion());
        parameters.setProperty(APIContext.CS_CALL_NAME, callName);
        parameters.setProperty(APIContext.CS_SOAP_ACTION, callName);
        parameters.setProperty(APIContext.CS_SITEID, Site.valueOf(context.getSite()).getId() + "");
        parameters.setProperty(APIContext.CS_APP_NAME, apiContext.getCsAppName());
        parameters.setProperty(APIContext.CS_CERT_NAME, apiContext.getCsCertName());
        parameters.setProperty(APIContext.CS_DEV_NAME, apiContext.getCsDevName());
        parameters.setProperty(APIContext.CS_ROUTING, apiContext.getCsRouting());
        parameters.setProperty(APIContext.CS_CLIENT, apiContext.getCsClient());

        String serverEndPoint = apiContext.getCs1EndPoint();
        APIRequest request = new APIRequest(serverEndPoint, parameters, parameters, xmlInput);
        APIDriver driver = new APIDriver(request);

        return driver.executeAPIRequest().getResponseXML();
    }

    public static com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.AbstractResponseType executeCSAPIRequest(
            com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.AbstractRequestType request, String callName, String token) throws APIException {
        String serverEndPoint = getAPIContext().getCsEndPoint();
        return executeCSAPIRequest(serverEndPoint, request, callName, token);
    }

    public static com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.AbstractResponseType executeCSAPIRequest(String endPoint,
            com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.AbstractRequestType request, String callName, String token) throws APIException {
        return executeCSAPIRequest(endPoint, request, callName, token, null, null);
    }

    public static com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.AbstractResponseType executeCSAPIRequest(String endPoint,
            com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.AbstractRequestType request, String callName, String userName, String password)
            throws APIException {
        return executeCSAPIRequest(endPoint, request, callName, getEBayCSToken(userName, password), null, null);
    }

    public static com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.AbstractResponseType executeCSAPIRequest(String endPoint,
            com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.AbstractRequestType request, String callName, String token, String goldenFile,
            Config xmlCompareConfig) throws APIException {

        com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.AbstractResponseType response = null;

        if (endPoint.indexOf("?") < 0) {
            endPoint = endPoint + "?callname=" + callName;
            if (request.getVersion() != null) {
                endPoint = endPoint + "&version=" + request.getVersion();
            } else {
                throw new RuntimeException("Version can't be null, please set it first!");
            }
        }

        try {
            SOAPHeaderElement header = new SOAPHeaderElement("urn:ebay:apis:eBLBaseComponents", "RequesterCredentials");
            header.addNamespaceDeclaration("", "urn:ebay:apis:eBLBaseComponents");
            SOAPElement tokenEl = header.addChildElement("ebl:eBayAuthToken");
            tokenEl.addNamespaceDeclaration("ebl", "urn:ebay:apis:eBLBaseComponents");
            tokenEl.addTextNode(token);

            EBayCSAPISoapBindingStub stub = new EBayCSAPISoapBindingStub();
            stub._setProperty(EBayCSAPISoapBindingStub.ENDPOINT_ADDRESS_PROPERTY, endPoint);
            stub.setHeader(header);

            @SuppressWarnings("rawtypes")
            Class[] parameterTypes = { request.getClass() };
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```
462             Method stubMethod = stub.getClass().getMethod(callName, parameterTypes);
463
464             response = (com.ebay.domain.apisoap.pres.service.hosting.soap.basecomponents.AbstractResponseType) stubMethod.invoke(stub, request);
465
466         } catch (Exception e) {
467             logger.error("Exception:" + e);
468             throw new APIException(e);
469         } finally {
470             //todo, regenerate log message, VIC
471 //          String requestXml = "";
472 //          String responseXml = "";
473 //          if (request != null) {
474 //              requestXml = AxisObjectUtil.serializeAxisCSBean(request);
475 //          }
476 //          if (response != null) {
477 //              responseXml = AxisObjectUtil.serializeAxisCSBean(response);
478 //          }
479 //          APIDriver.generateLogMessage(endPoint, new Properties(), requestXml, new Properties(), responseXml, goldenFile, xmlCompareConfig);
480         }
481         return response;
482     }
483
484     /**
485      * Execute CS API Request
486      *
487      * @param callName
488      * @param xmlInput
489      * @return responseXML
490      * @throws APIException
491      */
492     public static String executeCSAPIRequest(String callName, String xmlInput) throws APIException {
493         Properties parameters = new Properties();
494         Context context = ContextManager.getThreadContext();
495         APIContext apiContext = getAPIContext();
496         parameters.setProperty(APIContext.CS_VERSION, apiContext.getCsVersion());
497         parameters.setProperty(APIContext.CS_CALL_NAME, callName);
498         parameters.setProperty(APIContext.CS_SOAP_ACTION, callName);
499         parameters.setProperty(APIContext.CS_SITEID, Site.valueOf(context.getSite()).getId() + "");
500         parameters.setProperty(APIContext.CS_APP_NAME, apiContext.getCsAppName());
501         parameters.setProperty(APIContext.CS_CERT_NAME, apiContext.getCsCertName());
502         parameters.setProperty(APIContext.CS_DEV_NAME, apiContext.getCsDevName());
503         parameters.setProperty(APIContext.CS_ROUTING, apiContext.getCsRouting());
504         parameters.setProperty(APIContext.CS_CLIENT, apiContext.getCsClient());
505
506         String serverEndPoint = apiContext.getCsEndPoint();
507         APIRequest request = new APIRequest(serverEndPoint, parameters, parameters, xmlInput);
508         APIDriver driver = new APIDriver(request);
509
510         return driver.executeAPIRequest().getResponseXML();
511     }
512
513     /**
514      * Execute RTM API Request
515      *
516      * @param callName
517      * @param xmlInput
518      * @return responseXML
519      * @throws APIException
520      */
521     public static String executeRTMAPIRequest(String callName, String xmlInput) throws APIException {
522         Properties headers = new Properties();
523         APIContext apiContext = getAPIContext();
524         headers.setProperty(APIContext.RTM_CALL_NAME, callName);
525         headers.setProperty(APIContext.RTM_CALL_VERSION, apiContext.getRtmVersion());
526
527         String serverEndPoint = apiContext.getRtmEndPoint();
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```
528         APIRequest request = new APIRequest(serverEndPoint, headers, headers, xmlInput);
529         APIDriver driver = new APIDriver(request);
530
531         return driver.executeAPIRequest().getResponseXML();
532     }
533
534     /**
535      * Execute Schema API Request
536      *
537      * @param callName
538      * @param xmlInput
539      * @return responseXML
540      * @throws Exception
541      */
542     public static String executeSchemaAPIRequest(String callName, String xmlInput) throws Exception {
543         Properties headers = new Properties();
544         Context context = ContextManager.getThreadContext();
545         APIContext apiContext = getAPIContext();
546         headers.setProperty(APIContext.SCHEMA_COMPATIBILITY_LEVEL, apiContext.getSchemaCompatibilityLevel());
547         headers.setProperty(APIContext.SCHEMA_CALL_NAME, callName);
548         headers.setProperty(APIContext.SCHEMA_CERT_NAME, apiContext.getSchemaCertName());
549         headers.setProperty(APIContext.SCHEMA_APP_NAME, apiContext.getSchemaAppName());
550         headers.setProperty(APIContext.SCHEMA_DEV_NAME, apiContext.getSchemaDevName());
551         headers.setProperty(APIContext.SCHEMA_DETAIL_LEVEL, apiContext.getSchemaDetailLevel());
552         headers.setProperty(APIContext.SCHEMA_SITEID, Site.valueOf(context.getSite()).getId() + "");
553         headers.setProperty(APIContext.API_IS_INTERNAL_CLIENT, apiContext.getApiIsInternalClient());
554
555         String serverEndPoint = apiContext.getSchemaEndPoint();
556         APIRequest request = new APIRequest(serverEndPoint, null, headers, xmlInput);
557         APIDriver driver = new APIDriver(request);
558
559         return driver.executeAPIRequest().getResponseXML();
560     }
561
562     /**
563      * Execute executeSoapAPIRequest Request
564      *
565      * @param headers
566      * @param xmlInput
567      * @return
568      * @throws Exception
569      */
570     public static String executeSoapAPIRequest(Properties headers, String xmlInput) throws Exception {
571         APIContext apiContext = getAPIContext();
572
573         String endPoint = apiContext.getSoapEndPoint();
574         APIRequest request = new APIRequest(endPoint, null, headers, xmlInput);
575         APIDriver driver = new APIDriver(request);
576         return driver.executeAPIRequest().getResponseXML();
577     }
578
579     /**
580      * Execute SOAP API Request
581      *
582      * @param callName
583      * @param xmlInput
584      * @return responseXML
585      * @throws APIException
586      */
587     public static String executeSoapAPIRequest(String callName, String xmlInput) throws APIException {
588         Properties parameters = new Properties();
589         Context context = ContextManager.getThreadContext();
590         APIContext apiContext = getAPIContext();
591         parameters.setProperty(APIContext.SOAP_VERSION, apiContext.getSoapVersion());
592         parameters.setProperty(APIContext.SOAP_ACTION, callName);
593         parameters.setProperty(APIContext.SOAP_CALL_NAME, callName);
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
        parameters.setProperty(APIContext.SOAP_SITEID, Site.valueOf(context.getSite()).getId() + "");
        parameters.setProperty(APIContext.SOAP_APP_NAME, apiContext.getSoapAppName());
        parameters.setProperty(APIContext.SOAP_ROUTING, apiContext.getSoapRouting());
        parameters.setProperty(APIContext.SOAP_CLIENT, apiContext.getSoapClient());

        Properties headers = new Properties();
        headers.setProperty(APIContext.API_IS_INTERNAL_CLIENT, apiContext.getApiIsInternalClient());

        String serverEndPoint = apiContext.getSoapEndPoint();
        APIRequest request = new APIRequest(serverEndPoint, parameters, headers, xmlInput);
        APIDriver driver = new APIDriver(request);

        return driver.executeAPIRequest().getResponseXML();
    }

    /**
     * Execute SOA Request
     *
     * @param endPoint
     * @param headers
     * @param xmlInput
     * @return
     * @throws Exception
     */
    public static String executeSOARequest(String endPoint, Properties headers, String xmlInput) throws Exception {
        APIRequest request = new APIRequest(endPoint, null, headers, xmlInput);
        APIDriver driver = new APIDriver(request);
        return driver.executeAPIRequest().getResponseXML();
    }

    /**
     *
     * Assumption: Golden File has datestamp, timestamp, and all dynamic data
     * file removed before saved as golden file.
     *
     * @param subFile
     *            (golden file w/o datestamp or dynamic data)
     * @param wholeFile
     *            (wholestring of api/soa request, may contain header and
     *            timestamp0
     * @return boolean true if match
     * @throws IOException
     *             <p>
     *             Example of usage:
     *             </p>
     *             {@code
     *
     * assertTrue(SellingUsefulUtils.FileSubStringComparisionUtil(goldenFilePath, respZRequestedXML), "Message");
     * NOTE: GoldenFileXML.txt should be saved as *.txt,content cut and paste from *.xml as view from notepad,  not browser.
     * }
     * @author josnguyen & srividya
     */
    public static boolean FileSubStringComparisionUtil(String subFile, String wholeString) throws IOException {
        BufferedInputStream f = null;
        String goldenOutString = null;
        boolean found = false;

        byte[] buffer = new byte[(int) new File(subFile).length()];

        try {

            f = new BufferedInputStream(new FileInputStream(subFile));
            f.read(buffer);
            goldenOutString = new String(buffer);

            System.out.println("GOLDENFILECONTENT: " + goldenOutString);
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
            System.out.println("GOLDENFILECONTENT: " + goldenOutString);
            int index = IndexOf(wholeString.toCharArray(), goldenOutString.toCharArray());

            logger.log(Priority.INFO, "Index value now is: " + index);
            logger.log(Priority.INFO, "Length of Golden is: " + goldenOutString.length());

            if (index > 0) {
                logger.log(Priority.INFO, "MATCHES!!!\n");
                found = true;

            } else if (index < 0) {
                logger.log(Priority.INFO, "NO MATCHES!!!\n");
                found = false;
            }

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

        } finally {
            f.close();
        }

        return found;
    }

    /**
     * This is an overloaded version of previous, in the case you want to pass
     * in an entire path to your source(dynamic) file...
     *
     * @param subFile
     *            - path to expected golden file
     * @param wholeFile
     *            - path to sourceFile, which might contain dynamic data such as
     *            timestamp...
     * @param fileFlag
     *            - true if it is a file...
     * @return boolean true if match
     * @throws IOException
     * @author josnguyen & srividya
     */
    public static boolean FileSubStringComparisionUtil(String subFile, String wholeFile, boolean fileFlag) throws IOException {
        boolean found = false;

        if (fileFlag && wholeFile != null) {
            BufferedInputStream f = null;
            String goldenOutString = null;
            byte[] buffer = new byte[(int) new File(subFile).length()];

            BufferedInputStream fs = null;
            String dynamicOutFile = null;
            byte[] bufferS = new byte[(int) new File(wholeFile).length()];

            try {
                f = new BufferedInputStream(new FileInputStream(subFile));
                f.read(buffer);
                goldenOutString = new String(buffer);

                fs = new BufferedInputStream(new FileInputStream(wholeFile));
                fs.read(buffer);
                dynamicOutFile = new String(bufferS);

                int index = IndexOf(dynamicOutFile.toCharArray(), goldenOutString.toCharArray());
                System.out.print(index);

                if (index > 0 && index == goldenOutString.length()) {
                    System.out.println("MATCHES!!!");
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
                    found = true;
                } else if (index < 0 || index != goldenOutString.length()) {
                    System.out.println("NO MATCHES!!!");
                    found = false;
                }
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                f.close();
            }

            return found;

        } else {
            return false;
        }
    }

    public static String formatRequestXML(String requestXML, Properties headers) {
        if (headers == null || headers.size() == 0) {
            return requestXML;
        }
        // if Element <Version> is existed in xml and header also contain
        // version info, then overwrite the xml version content.
        if (requestXML.indexOf("<Version>") > -1) {
            String version = "";
            if (headers.containsKey(APIContext.SCHEMA_COMPATIBILITY_LEVEL)) {
                version = headers.getProperty(APIContext.SCHEMA_COMPATIBILITY_LEVEL);
            } else if (headers.containsKey(APIContext.SOAP_VERSION)) {
                version = headers.getProperty(APIContext.SOAP_VERSION);
            } else if (headers.containsKey(APIContext.RTM_CALL_VERSION)) {
                version = headers.getProperty(APIContext.RTM_CALL_VERSION);
            } else if (headers.containsKey(APIContext.CS_VERSION)) {
                version = headers.getProperty(APIContext.CS_VERSION);
            }

            if (version != null && !version.trim().equals("")) {
                requestXML = requestXML.replaceFirst("<Version>\\d*<\\/Version>", "<Version>" + version + "</Version>");
            }
        }

        if (requestXML.indexOf("ebl:DevId") > -1) {
            String devId = "";
            if (headers.containsKey(APIContext.CS_DEV_NAME)) {
                devId = headers.getProperty(APIContext.CS_DEV_NAME);
            }

            if (devId != null && !devId.trim().equals("")) {
                requestXML = requestXML.replaceFirst("<ebl:DevId>\\w*<\\/ebl:DevId>", "<ebl:DevId>" + devId + "</ebl:DevId>");
            }
        }

        if (requestXML.indexOf("ebl:AppId") > -1) {
            String appId = "";
            if (headers.containsKey(APIContext.CS_APP_NAME)) {
                appId = headers.getProperty(APIContext.CS_APP_NAME);
            }

            if (appId != null && !appId.trim().equals("")) {
                requestXML = requestXML.replaceFirst("<ebl:AppId>\\w*<\\/ebl:AppId>", "<ebl:AppId>" + appId + "</ebl:AppId>");
            }
        }

        if (requestXML.indexOf("ebl:AuthCert") > -1) {
            String authCert = "";

            if (headers.containsKey(APIContext.CS_CERT_NAME)) {
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
                authCert = headers.getProperty(APIContext.CS_CERT_NAME);
            }

            if (authCert != null && !authCert.trim().equals("")) {
                requestXML = requestXML.replaceFirst("<ebl:AuthCert>\\w*<\\/ebl:AuthCert>", "<ebl:AuthCert>" + authCert + "</ebl:AuthCert>");
            }
        }
        return requestXML;
    }

    /**
     * @param fileName
     *            - input string name of file only. ex: "myFile.txt"
     * @return String representing full absolute path to file. ex:
     *         "C:\myVob\ecdf\.....\myFile.txt"
     * @throws Exception
     * @author josnguyen & srividya
     */
    public static String getAbsolutePathToFile(String fileName, @SuppressWarnings("rawtypes") Class clazz) throws Exception {

        String fullPath = null;
        try {
            URL url = clazz.getResource(fileName);
            fullPath = url.getPath();
            if (System.getProperty("os.name").contains("Windows") && fullPath.startsWith("/")) {
                fullPath = fullPath.substring(1);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return fullPath;
    }

    private static ApiContext getApiContext(String userName, String password, String serverEndPoint) throws XMLException, APIException {
        Context context = ContextManager.getThreadContext();
        APIContext mauiApiContext = getAPIContext();

        ApiContext apiContext = new ApiContext();
        apiContext.setSite(Site.valueOf(context.getSite()).getSiteCodeType());
        ApiCredential cred = apiContext.getApiCredential();
        ApiAccount account = cred.getApiAccount();
        account.setDeveloper(mauiApiContext.getSchemaDevName());
        account.setApplication(mauiApiContext.getSchemaAppName());
        account.setCertificate(mauiApiContext.getSchemaCertName());
        apiContext.setErrorLanguage("en_US");
        apiContext.getApiLogging().setLogHTTPHeaders(true);
        apiContext.setApiServerUrl(serverEndPoint);

        cred.seteBayToken(getEBayToken(userName, password));
        return apiContext;
    }

    public static String getEBayCSToken(String userName, String password) throws APIException {
        String endPoint = getAPIContext().getCsEndPoint();
        return getEBayCSToken(endPoint, userName, password);
    }

    /**
     * Get the eBay CS Token for eBay Account
     *
     * @param account
     * @return
     * @throws XMLException
     * @throws APIException
     */
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
    public synchronized static String getEBayCSToken(String endPoint, String userName, String password) throws APIException {
        EBayCSAPISoapBindingStub stub = null;
        CSGetTokenRequestType request = null;
        CSGetTokenResponseType response = null;

        APIContext apiContext = getAPIContext();

        endPoint = endPoint + "?" + APIContext.CS_CALL_NAME + "=CSGetToken&" + APIContext.CS_VERSION + "=" + apiContext.getCsVersion();

        try {
            SOAPHeaderElement header = new SOAPHeaderElement("urn:ebay:apis:eBLBaseComponents", "RequesterCredentials");
            header.addNamespaceDeclaration("", "urn:ebay:apis:eBLBaseComponents");

            SOAPElement credEl = header.addChildElement("ebl:Credentials");
            credEl.addNamespaceDeclaration("ebl", "urn:ebay:apis:eBLBaseComponents");
            SOAPElement devIdEl = credEl.addChildElement("ebl:DevId");
            devIdEl.addTextNode(apiContext.getCsDevName());
            SOAPElement appIdEl = credEl.addChildElement("ebl:AppId");
            appIdEl.addTextNode(apiContext.getCsAppName());
            SOAPElement authCertEl = credEl.addChildElement("ebl:AuthCert");
            authCertEl.addTextNode(apiContext.getCsCertName());
            SOAPElement userNameEl = credEl.addChildElement("ebl:Username");
            userNameEl.addTextNode(userName);
            SOAPElement passwordEl = credEl.addChildElement("ebl:Password");
            passwordEl.addTextNode(password);

            stub = new EBayCSAPISoapBindingStub();
            stub._setProperty(EBayCSAPISoapBindingStub.ENDPOINT_ADDRESS_PROPERTY, endPoint);
            stub.setHeader(header);

            request = new CSGetTokenRequestType();
            request.setVersion(apiContext.getCsVersion());
            response = stub.CSGetToken(request);
            String token = response.getEBayAuthToken();

            return token;
        } catch (Exception e) {
            logger.error("Exception:" + e);
            throw new APIException(e);
        } finally {
            //todo regenerate log message, VIC
//          String requestXml = AxisObjectUtil.serializeAxisCSBean(request);
//          String responseXml = AxisObjectUtil.serializeAxisCSBean(response);
//          Logging.logAPICall("", APIDriver.generateLogMessage(endPoint, new Properties(), requestXml, new Properties(), responseXml, null, null), false);
//          APIDriver.generateLogMessage(endPoint, new Properties(), requestXml, new Properties(), responseXml, null, null);
        }
    }

    /**
     * Get the eBay Token for eBay Account
     *
     * @param userName
     * @param password
     * @return
     * @throws XMLException
     * @throws APIException
     */
    public static String getEBayToken(String userName, String password) throws XMLException, APIException {
        return getEBayToken(userName, password, new Properties());
    }

    /**
     * Get the eBay Token for eBay Account
     *
     * @param userName
     * @param password
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
     * @param password
     * @param headers
     * @return
     * @throws XMLException
     * @throws APIException
     */
    public static String getEBayToken(String userName, String password, Properties headers) throws XMLException, APIException {
        if(headers == null) {
            try {
                return APIUtil.getEBayToken(userName, password);
            }
            catch (Exception e){}
        }
        String key = userName + headers.getProperty(APIContext.SCHEMA_APP_NAME);
        if (ebayTokenMap.containsKey(key)) {
            return ebayTokenMap.get(key);
        }
        Context context = ContextManager.getThreadContext();
        String inputXml = "<?xml version=\"1.0\" encoding="utf-8\"?>" + "<GetTokenRequest xmlns=\"urn:ebay:apis:eBLBaseComponents\">" + "<Version>" + ((APIContext) context.getApiContext()).getSchemaCompatibilityLevel
                + "<RequesterCredentials>" + "<Username>" + userName + "</Username>" + "<Password>" + password + "</Password>" + "</RequesterCredentials>" + "</GetTokenRequest>";

        try {
            APIContext apiContext = (APIContext) context.getApiContext();
            headers.setProperty(APIContext.SCHEMA_COMPATIBILITY_LEVEL, apiContext.getSchemaCompatibilityLevel());
            headers.setProperty(APIContext.SCHEMA_CALL_NAME, "GetToken");
            if (headers.getProperty(APIContext.SCHEMA_CERT_NAME) == null)
                headers.setProperty(APIContext.SCHEMA_CERT_NAME, apiContext.getSchemaCertName());
            if (headers.getProperty(APIContext.SCHEMA_APP_NAME) == null)
                headers.setProperty(APIContext.SCHEMA_APP_NAME, apiContext.getSchemaAppName());
            if (headers.getProperty(APIContext.SCHEMA_DEV_NAME) == null)
                headers.setProperty(APIContext.SCHEMA_DEV_NAME, apiContext.getSchemaDevName());
            headers.setProperty(APIContext.SCHEMA_DETAIL_LEVEL, apiContext.getSchemaDetailLevel());
            headers.setProperty(APIContext.SCHEMA_SITEID, "0");
            headers.setProperty(APIContext.API_IS_INTERNAL_CLIENT, apiContext.getApiIsInternalClient());

            String serverEndPoint = apiContext.getSchemaEndPoint();
            APIRequest request = new APIRequest(serverEndPoint, null, headers, inputXml);
            APIDriver driver = new APIDriver(request);

            String responsXml = driver.executeAPIRequest().getResponseXML();
            String[] results = XPathUtil.getXPathContent(responsXml, "//*[local-name()='eBayAuthToken']");
            if (results != null && results.length > 0) {
                ebayTokenMap.put(key, results[0]);
                return results[0];
            } else {
                throw new APIException("Unable to retrieve eBayAuthToken.");
            }
        } catch (Exception e) {
            throw new APIException("Unable to retrieve eBayAuthToken.");
        }
    }


    /**
     * Function to Index for FileSubStringComparisionUtil
     *
     * @param str
     * @param substr
     * @return int - number of characters matched. Example of usage: int index =
     *         IndexOf(goldenOutString.toCharArray() ,
     *         inputSubString.toCharArray());
     */
    private static int IndexOf(char[] str, char[] substr) {
        boolean match;
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
        int LENGTH_OF_ERROR_POINT_TO_PROVIDE = 10;
        char[] noMatchPoint = new char[LENGTH_OF_ERROR_POINT_TO_PROVIDE];
        int i = 0, j = 0;
        @SuppressWarnings("unused")
        int lastIPos = 0, lastJPos = 0;

        for (i = 0; i < str.length - substr.length + 1; ++i) {
            if (i > 0)
                lastIPos = i;
            match = true;
            for (j = 0; j < substr.length; ++j) {
                if (j > 0)
                    lastJPos = j; // Track last value of J before mismatch, once
                                  // matched.
                if (str[i + j] != substr[j]) {
                    match = false;
                    break;
                }
            }
            if (match)
                return i;

        }

        int k;
        if (i == str.length - substr.length + 1) {// meaning there was a
                                                  // mismatch somewhere in
                                                  // between

            for (k = 0; k < LENGTH_OF_ERROR_POINT_TO_PROVIDE; k++) {
                noMatchPoint[k] = str[i++]; // Return back the next 5 or so
                                            // characters from the point of
                                            // mismatch to help debug
            }
            System.out.printf("!!!DEBUGGING!!!: ASSISTED WORD TO HELP LOCATE MISMATCH IN OUTPUT IS: \"%s\"\n", String.valueOf(noMatchPoint));
        }
        return -1;
    }

    /**
     * <p>
     * Insert a new xml element by specified xml tag name.
     * </p>
     * <p>
     * Example: when you call this method with
     * (xmlTagName="RequesterCredentials", addedContent="<Test>eCAF</Test>")
     * </p>
     * <p>
     * Input:
     * </p>
     *
     * <pre>
     * {@code
     * <?xml version="1.0" encoding="utf-8"?>
     * <GeteBayOfficialTimeRequest xmlns="urn:ebay:apis:eBLBaseComponents">
     *   <Version>547</Version>
     *   <RequesterCredentials>
     *     <Username>apitest1</Username>
     *     <Password>ebay</Password>
     *   </RequesterCredentials>
     * </GeteBayOfficialTimeRequest>
     * }
     * </pre>
     * <p>
     * Output:
     * </p>
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
     * <pre>
     * {@code
     * <?xml version="1.0" encoding="utf-8"?>
     * <GeteBayOfficialTimeRequest xmlns="urn:ebay:apis:eBLBaseComponents">
     *   <Version>547</Version>
     *   <RequesterCredentials>
     *     <Test>eCAF</Test>
     *     <Username>apitest1</Username>
     *     <Password>ebay</Password>
     *   </RequesterCredentials>
     * </GeteBayOfficialTimeRequest>
     * }
     * </pre>
     *
     * @param xmlContent
     * @param xmlTagName
     * @param addedContent
     * @return
     * @throws Exception
     */
    public static String insertXMLElementByContent(String xmlContent, String xmlTagName, String addedContent) throws Exception {
        Document doc = convertStrAsDocument(xmlContent, true);
        xmlContent = convertDocumentAsFormatStr(doc, "");
        xmlContent = xmlContent.replaceAll(">[\\s\\t]+<", "><");

        addedContent = addedContent.replaceAll("\\n", "").replaceAll("\r", "").replaceAll(">[\\s\\t]+<", "><");

        xmlContent = xmlContent.replaceAll("</" + xmlTagName + ">", "<" + xmlTagName + ">" + addedContent);

        doc = convertStrAsDocument(xmlContent, true);
        return convertDocumentAsFormatStr(doc);

    }

    /**
     * <p>
     * Insert a new xml element by specified Xpath.
     * </p>
     * <p>
     * Example: when you call this method with
     * (Xpath="//GeteBayOfficialTimeRequest/RequesterCredentials",
     * value="<Test>eCAF</Test>")
     * </p>
     * <p>
     * Input:
     * </p>
     *
     * <pre>
     * {@code
     * <?xml version="1.0" encoding="utf-8"?>
     * <GeteBayOfficialTimeRequest xmlns="urn:ebay:apis:eBLBaseComponents">
     *   <Version>547</Version>
     *   <RequesterCredentials>
     *     <Username>apitest1</Username>
     *     <Password>ebay</Password>
     *   </RequesterCredentials>
     * </GeteBayOfficialTimeRequest>
     * }
     * </pre>
     * <p>
     * Output:
     * </p>
     *
     * <pre>
     * {@code
     * <?xml version="1.0" encoding="utf-8"?>
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```
1121     * <GeteBayOfficialTimeRequest xmlns="urn:ebay:apis:eBLBaseComponents">
1122     *   <Version>547</Version>
1123     *   <RequesterCredentials>
1124     *     <Username>apitest1</Username>
1125     *     <Password>ebay</Password>
1126     *     <Test>eCAF</Test>
1127     *   </RequesterCredentials>
1128     * </GeteBayOfficialTimeRequest>
1129     * }
1130     * </pre>
1131     *
1132     * @param xmlContent
1133     * @param xPpathLocation
1134     * @param value
1135     * @return
1136     * @throws Exception
1137     */
1138    public static String insertXMLElementByXPath(String xmlContent, String xPpathLocation, String value) throws Exception {
1139        DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
1140        Document doc = builder.parse(new InputSource(new StringReader(xmlContent)));
1141
1142        Element resolver = doc.createElement("NamespaceResolver");
1143        resolver.setAttribute("xmlns:m", "http://www.kijiji.com/soap");
1144        resolver.setAttribute("xmlns:SOAP", "http://www.w3.org/2003/05/soap-envelope");
1145
1146        Node parent = XPathAPI.selectSingleNode(doc, xPpathLocation, resolver);
1147        Document tempDoc = convertStrAsDocument("<?xml version=\"1.0\"?><wrapper>" + value.trim() + "</wrapper>", true);
1148        Node tempNode = tempDoc.getDocumentElement();
1149        NodeList nl = tempNode.getChildNodes();
1150        for (int i = 0; i < nl.getLength(); i++) {
1151            Element oneToMove = (Element) nl.item(i);
1152            Node newOneToMove = doc.importNode(oneToMove, true);
1153            parent.appendChild(newOneToMove);
1154        }
1155
1156        return convertDocumentAsFormatStr(doc);
1157
1158    }
1159
1160    /**
1161     * @author josnguyen
1162     * @param version
1163     * @param apiCallName
1164     * @return
1165     * @throws Exception
1166     */
1167    public static Properties schemaAPISetHeaderProperties(String version, String apiCallName) throws Exception {
1168        Properties headers = new Properties();
1169        headers.setProperty(APIContext.SCHEMA_CERT_NAME, "AdminCertificate"); // certName
1170        headers.setProperty(APIContext.SCHEMA_APP_NAME, "AdminApp"); // appName
1171        headers.setProperty(APIContext.SCHEMA_DEV_NAME, "Admintest1"); // devName
1172        headers.setProperty(APIContext.SCHEMA_SITEID, "0"); // siteID
1173        headers.setProperty(APIContext.SCHEMA_DETAIL_LEVEL, "0");
1174        headers.setProperty(APIContext.SCHEMA_COMPATIBILITY_LEVEL, version);
1175        headers.setProperty(APIContext.SCHEMA_CALL_NAME, apiCallName);
1176
1177        return headers;
1178    }
1179
1180    /**
1181     * @author josnguyen
1182     * @param detailLevel
1183     * @param compatibleLevel
1184     * @param apiCallName
1185     * @return
1186     * @throws Exception
1187     */
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```
1187     */
1188    public static Properties schemaAPISetHeaderProperties(String certname, String appName, String devName, String siteId, String detailLevel,
1189            String compatibleLevel, String apiCallName) throws Exception {
1190        Properties headers = new Properties();
1191        headers.setProperty(APIContext.SCHEMA_CERT_NAME, certname); // certName
1192        headers.setProperty(APIContext.SCHEMA_APP_NAME, appName); // appName
1193        headers.setProperty(APIContext.SCHEMA_DEV_NAME, devName); // devName
1194        headers.setProperty(APIContext.SCHEMA_SITEID, siteId); // siteID
1195        headers.setProperty(APIContext.SCHEMA_DETAIL_LEVEL, detailLevel);
1196        headers.setProperty(APIContext.SCHEMA_COMPATIBILITY_LEVEL, compatibleLevel);
1197        headers.setProperty(APIContext.SCHEMA_CALL_NAME, apiCallName);
1198
1199        return headers;
1200    }
1201
1202    /**
1203     * @author josnguyen
1204     * @param userName
1205     * @param passWord
1206     * @param saleTemplateID
1207     * @return
1208     * @throws Exception
1209     */
1210
1211    public static Properties schemaAPISetRequestProperties(String userName, String passWord) throws Exception {
1212        Properties requestParams = new Properties();
1213        requestParams.setProperty("Username", userName);
1214        requestParams.setProperty("Password", passWord);
1215
1216        return requestParams;
1217
1218    }
1219
1220    /**
1221     * @author josnguyen
1222     * @param userName
1223     * @param passWord
1224     * @param version
1225     * @param detailLevel
1226     * @return
1227     * @throws Exception
1228     */
1229    public static Properties schemaAPISetRequestProperties(String userName, String passWord, String version) throws Exception {
1230        Properties requestParams = new Properties();
1231        requestParams.setProperty("Username", userName);
1232        requestParams.setProperty("Password", passWord);
1233        requestParams.setProperty("Version", version);
1234
1235        return requestParams;
1236    }
1237
1238    public static Context setMauiContext(Site site) {
1239        APIContext mauiApiContext = null;
1240        Context context = ContextManager.getThreadContext();
1241
1242        if (context.getApiContext() != null) {
1243            if (site != null)
1244                context.setSite(site.getCode());
1245        } else {
1246            mauiApiContext = new APIContext();
1247            mauiApiContext.load(context.getTestNGContext(), context);
1248
1249            context.setSite(site.getCode());
1250        }
1251        ContextManager.setThreadContext(context);
1252        ContextManager.setGlobalContext(context);
1253        return context;
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```
1254     }
1255
1256     /**
1257      * <p>
1258      * Update the current value of xml element by specified xml tag name.
1259      * </p>
1260      * <p>
1261      * Example: when you call this method with (xmlTagName="Username",
1262      * newContent="NEW_ID")
1263      * </p>
1264      * <p>
1265      * Input:
1266      * </p>
1267      *
1268      * <pre>
1269      * {@code
1270      * <?xml version="1.0" encoding="utf-8"?>
1271      * <GeteBayOfficialTimeRequest xmlns="urn:ebay:apis:eBLBaseComponents">
1272      *   <Version>547</Version>
1273      *   <RequesterCredentials>
1274      *     <Username>apitest1</Username>
1275      *     <Password>ebay</Password>
1276      *   </RequesterCredentials>
1277      * </GeteBayOfficialTimeRequest>
1278      * }
1279      * </pre>
1280      * <p>
1281      * Output:
1282      * </p>
1283      *
1284      * <pre>
1285      * {@code
1286      * <?xml version="1.0" encoding="utf-8"?>
1287      * <GeteBayOfficialTimeRequest xmlns="urn:ebay:apis:eBLBaseComponents">
1288      *   <Version>547</Version>
1289      *   <RequesterCredentials>
1290      *     <Username>NEW_ID</Username>
1291      *     <Password>ebay</Password>
1292      *   </RequesterCredentials>
1293      * </GeteBayOfficialTimeRequest>
1294      * }
1295      * </pre>
1296      *
1297      * @param xmlContent
1298      * @param xmlTagName
1299      * @param newContent
1300      * @return
1301      * @throws Exception
1302      */
1303     public static String updateXMLElementByContent(String xmlContent, String xmlTagName, String newContent) throws Exception {
1304         Document doc = convertStrAsDocument(xmlContent, true);
1305         xmlContent = convertDocumentAsFormatStr(doc, "");
1306         xmlContent = xmlContent.replaceAll(">[\\s\\t]+<", "><");
1307
1308         xmlContent = xmlContent.replaceAll("<" + xmlTagName + ".+?<\\/" + xmlTagName + ">", "<" + xmlTagName + ">" + newContent + "</" + xmlTagName + ">");
1309
1310         doc = convertStrAsDocument(xmlContent, true);
1311         return convertDocumentAsFormatStr(doc);
1312     }
1313
1314     /**
1315      * <p>
1316      * Update the current value of xml element by specified Xpath.
1317      * </p>
1318      * <p>
1319      * Example: when you call this method with
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```
1319      * Example: when you call this method with
1320      * (Xpath="//GeteBayOfficialTimeRequest/RequesterCredentials/Username",
1321      * value="NEW_ID")
1322      * </p>
1323      * <p>
1324      * Input:
1325      * </p>
1326      *
1327      * <pre>
1328      * {@code
1329      * <?xml version="1.0" encoding="utf-8"?>
1330      * <GeteBayOfficialTimeRequest xmlns="urn:ebay:apis:eBLBaseComponents">
1331      *   <Version>547</Version>
1332      *   <RequesterCredentials>
1333      *     <Username>apitest1</Username>
1334      *     <Password>ebay</Password>
1335      *   </RequesterCredentials>
1336      * </GeteBayOfficialTimeRequest>
1337      * }
1338      * </pre>
1339      * <p>
1340      * Output:
1341      * </p>
1342      *
1343      * <pre>
1344      * {@code
1345      * <?xml version="1.0" encoding="utf-8"?>
1346      * <GeteBayOfficialTimeRequest xmlns="urn:ebay:apis:eBLBaseComponents">
1347      *   <Version>547</Version>
1348      *   <RequesterCredentials>
1349      *     <Username>NEW_ID</Username>
1350      *     <Password>ebay</Password>
1351      *   </RequesterCredentials>
1352      * </GeteBayOfficialTimeRequest>
1353      * }
1354      * </pre>
1355      *
1356      * @param xmlContent
1357      * @param xPpathLocation
1358      * @param value
1359      * @return
1360      * @throws Exception
1361      */
1362     public static String updateXMLElementByXPath(String xmlContent, String xPpathLocation, String value) throws Exception {
1363         DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
1364         Document doc = builder.parse(new InputSource(new StringReader(xmlContent)));
1365
1366         Element resolver = doc.createElement("NamespaceResolver");
1367         resolver.setAttribute("xmlns:m", "http://www.kijiji.com/soap");
1368         resolver.setAttribute("xmlns:SOAP", "http://www.w3.org/2003/05/soap-envelope");
1369
1370         Node n = XPathAPI.selectSingleNode(doc, xPpathLocation, resolver);
1371         CharacterData cd = (CharacterData) n.getFirstChild();
1372         cd.setData(value);
1373
1374         return convertDocumentAsFormatStr(doc);
1375
1376     }
1377
1378     public static void setApiSoapEndPoint(String apiSoapEndPoint) {
1379         getAPIContext().setSoapEndPoint(apiSoapEndPoint);
1380     }
1381
1382     public static void setApiSchemaEndPoint(String apiSchemaEndPoint) {
1383         getAPIContext().setSchemaEndPoint(apiSchemaEndPoint);
1384     }
```

Read-Only | Smart Insert | 34 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
 */
public final class ArrayUtil {

    /**
     * Checks if the elements in the array are sorted
     */
    public static boolean isSorted(String[] str, boolean isStr) {
        boolean asc = false, desc = false;

        if (isStr) {
            for (int z = 0; z < str.length - 1; z++) {
                if (str[z].compareTo(str[z + 1]) == 0)
                    continue;
                if (str[z].compareTo(str[z + 1]) < 0) {
                    asc = true;
                    if (desc)
                        return false;
                }
                if (str[z].compareTo(str[z + 1]) > 0) {
                    desc = true;
                    if (asc)
                        return false;
                }
            }
        } else {
            for (int z = 0; z < str.length - 1; z++) {
                if (Double.valueOf(str[z]).compareTo(Double.valueOf(str[z + 1])) == 0)
                    continue;
                if (Double.valueOf(str[z]).compareTo(Double.valueOf(str[z + 1])) < 0) {
                    asc = true;
                    if (desc)
                        return false;
                }
                if (Double.valueOf(str[z]).compareTo(Double.valueOf(str[z + 1])) > 0) {
                    desc = true;
                    if (asc)
                        return false;
                }
            }
        }
        return true;
    }

    /**
     * Checks if the elements in the array are sorted in ascennding
     */
    public static boolean isSortedAsc(String[] str, boolean isStr) {

        if (isStr) {
            for (int z = 0; z < str.length - 1; z++) {
                if (str[z].compareTo(str[z + 1]) == 0)
                    continue;
                if (str[z].compareTo(str[z + 1]) < 0) {
                    continue;
                } else {
                    return false;
                }
            }
        } else {
            for (int z = 0; z < str.length - 1; z++) {
                if (Double.valueOf(str[z]).compareTo(Double.valueOf(str[z + 1])) == 0)
                    continue;
                if (Double.valueOf(str[z]).compareTo(Double.valueOf(str[z + 1])) < 0) {
                    continue;
                } else {
                    return false;
```

Read-Only | Smart Insert | 1 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class

```java
                    return false;
                }
            }
        }
        return true;
    }

    /**
     * Checks if the elements in the array are sorted in ascennding
     */
    public static boolean isSortedAsc(String[] str, boolean isStr) {

        if (isStr) {
            for (int z = 0; z < str.length - 1; z++) {
                if (str[z].compareTo(str[z + 1]) == 0)
                    continue;
                if (str[z].compareTo(str[z + 1]) < 0) {
                    continue;
                } else {
                    return false;
                }
            }
        } else {
            for (int z = 0; z < str.length - 1; z++) {
                if (Double.valueOf(str[z]).compareTo(Double.valueOf(str[z + 1])) == 0)
                    continue;
                if (Double.valueOf(str[z]).compareTo(Double.valueOf(str[z + 1])) < 0) {
                    continue;
                } else {
                    return false;
                }
            }
        }
        return true;
    }

    /**
     * Checks if the elements in the array are sorted in descending
     */
    public static boolean isSortedDesc(String[] str, boolean isStr) {

        if (isStr) {
            for (int z = 0; z < str.length - 1; z++) {
                if (str[z].compareTo(str[z + 1]) == 0)
                    continue;
                if (str[z].compareTo(str[z + 1]) < 0) {
                    return false;
                } else {
                    continue;
                }
            }
        } else {
            for (int z = 0; z < str.length - 1; z++) {
                if (Double.valueOf(str[z]).compareTo(Double.valueOf(str[z + 1])) == 0)
                    continue;
                if (Double.valueOf(str[z]).compareTo(Double.valueOf(str[z + 1])) < 0) {
                    return false;
                } else {
                    continue;
                }
            }
        }
        return true;
    }

}
```

Read-Only | Smart Insert | 1 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class | AxisObjectUtil.class

```java
package com.ebay.maui.util;

import static org.testng.internal.EclipseInterface.ASSERT_LEFT;
import static org.testng.internal.EclipseInterface.ASSERT_MIDDLE;
import static org.testng.internal.EclipseInterface.ASSERT_RIGHT;

import java.util.List;

import org.apache.commons.lang3.StringUtils;
import org.testng.Assert;
import org.testng.Reporter;

import com.ebay.maui.controller.ContextManager;
import com.ebay.maui.controller.Logging;
import com.ebay.maui.util.internal.web.LinksChecker;

/**
 * A enhanced Assertion supports soft assert. It uses soft assert if context
 * softAssertEnabled is true.
 *
 * soft assert - test case continues when validation fails.
 *
 */
public class AssertionUtil {

    public static final String COLON = " : ";
    private static final String IS_NULL = " IS NULL";
    private static final String IS_NOT_NULL = " IS NOT NULL";
    private static final String ASSERTION_PASS = "ASSERTION PASS: ";
    private static final String ASSERTION_FAILED = "ASSERTION FAILED: ";

    private static void _addVerificationFailure(Throwable e) {
    ContextManager.getThreadContext().addVerificationFailures(Reporter.getCurrentTestResult(), e);
    }

    private static String failFormat(Object actual, Object expected, String message) {
    String formatted = ASSERTION_FAILED;
    if (null != message) {
        formatted = message + COLON;
    }
    return formatted + ASSERT_LEFT + expected + ASSERT_MIDDLE + actual + ASSERT_RIGHT;
    }

    private static String passFormat(Object actual, String message) {
    return ASSERTION_PASS + message + " as expected " + COLON + actual;
    }

    // ///////////////// Assertion Methods ////////////////////////////
    public static void assertEquals(Object actual, Object expected, String message) {
    if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
        softAssertEquals(actual, expected, message);
    } else {
        try {
        Assert.assertEquals(actual, expected, message);
        } catch (AssertionError e) {
        Logging.error(failFormat(actual, expected, message));
        throw e;
        }
    }
    Logging.info(passFormat(actual, message));
    }

    public static void assertNotEquals(Object actual, Object expected, String message) {
    if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
        softAssertNotEquals(actual, expected, message);
    } else {
        try {
```

Read-Only | Smart Insert | 217 : 80

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class | AxisObjectUtil.class

```java
    } else {
        try {
        Assert.assertNotEquals(actual, expected, message);
        } catch (AssertionError e) {
        Logging.error(failFormat(actual, expected, message));
        throw e;
        }
    }
    Logging.info(passFormat(actual, message));
    }

    public static void assertContains(String actual, String contained, String message) {
    if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
        softAssertContains(actual, contained, message);
    } else {
        if (StringUtils.isEmpty(actual)) {
        throw new AssertionError(message + " is null.");
        }
        try {
        Assert.assertTrue(actual.contains(contained), message);
        } catch (AssertionError e) {
        Logging.error(message + COLON + " does not contain " + ASSERT_LEFT + contained + ASSERT_MIDDLE + actual + ASSERT_RIGHT);
        throw e;
        }
    }
    Logging.info(passFormat(actual, message));
    }

    public static void assertSame(Object actual, Object expected, String message) {
    if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
        softAssertSame(actual, expected, message);
    } else {
        try {
        Assert.assertSame(actual, expected, message);
        } catch (AssertionError e) {
        Logging.error(failFormat(actual, expected, message));
        throw e;
        }
    }
    Logging.info(passFormat(actual, message));
    }

    public static void assertNotSame(Object actual, Object expected, String message) {
    if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
        softAssertNotSame(actual, expected, message);
    } else {
        try {
        Assert.assertNotSame(actual, expected, message);
        } catch (AssertionError e) {
        Logging.error(failFormat(actual, expected, message));
        throw e;
        }
    }
    Logging.info(passFormat(actual, message));
    }

    public static void assertTrue(boolean condition, String message) {
    if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
        softAssertTrue(condition, message);
    } else {
        try {
        Assert.assertTrue(condition, message);
        } catch (AssertionError e) {
        Logging.error(failFormat(false, true, message));
        throw e;
        }
    }
```

Read-Only | Smart Insert | 217 : 80

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class | AxisObjectUtil.class

```
    Logging.info(ASSERTION_PASS + "Assertion result is expected = TRUE - [" + message + "]");
    }

    public static void assertFalse(boolean condition, String message) {
    if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
        softAssertFalse(condition, message);
    } else {
        try {
        Assert.assertFalse(condition, message);
        } catch (AssertionError e) {
        Logging.error(failFormat(true, false, message));
        throw e;
        }
    }
    Logging.info(ASSERTION_PASS + "Assertion result is expected = FALSE - [" + message + "]");
    }

    public static void assertNull(Object object, String message) {
    if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
        softAssertNull(object, message);
    } else {
        try {
        Assert.assertNull(object, message);
        } catch (AssertionError e) {
        Logging.error(message + IS_NOT_NULL);
        throw e;
        }
    }
    Logging.info(passFormat(IS_NULL, message));
    }

    public static void assertNotNull(Object object, String message) {
    if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
        softAssertNotNull(object, message);
    } else {
        try {
        Assert.assertNotNull(object, message);
        } catch (AssertionError e) {
        Logging.error(message + IS_NULL);
        throw e;
        }
    }
    Logging.info(passFormat(IS_NOT_NULL, message));
    }

    public static void assertUrlNoResponse(String url, String resourceName, boolean useProxy) {
    if (StringUtils.isEmpty(url)) {
        Logging.error(resourceName + ": URL is empty.");
        Assert.fail("URL is empty");
    }
    boolean ret = LinksChecker.validateResponseCode(url, resourceName, useProxy);
    String msg = resourceName + " : No response of URL.";
    if (!ret) {
        Logging.error(msg + " - " + url);
        if (ContextManager.getThreadContext().isSoftAssertEnabled()) {
        _addVerificationFailure(new AssertionError(msg));
        } else {
        throw new AssertionError(msg);
        }
    }
    Logging.info(ASSERTION_PASS + resourceName + " URL is OK - " + url);
    }

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

Read-Only | Smart Insert | 217 : 80

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class | AxisObjectUtil.class

```
    public static List<Throwable> getVerificationFailures() {
    return ContextManager.getThreadContext().getVerificationFailures(Reporter.getCurrentTestResult());
    }

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

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

    try {
        Assert.assertEquals(actual, expected, message);
        Logging.info(passFormat(actual, message));
    } catch (Throwable e) {
        Logging.error(failFormat(actual, expected, message));
        _addVerificationFailure(e);
    }
    }

    public static void softAssertNotEquals(Object actual, Object expected, String message) {

    try {
        Assert.assertNotEquals(actual, expected, message);
        Logging.info(passFormat(actual, message));
    } catch (Throwable e) {
        Logging.error(failFormat(actual, expected, message));
        _addVerificationFailure(e);
    }
    }

    public static boolean softAssertEqualsWithReturn(Object actual, Object expected, String message) {

    try {
        Assert.assertEquals(actual, expected, message);
        Logging.info(passFormat(actual, message));
    } catch (Throwable e) {
        Logging.error(failFormat(actual, expected, message));
        _addVerificationFailure(e);
        return false;
    }
    return true;
    }

    public static void softAssertFalse(boolean condition, String message) {
    try {
        Assert.assertFalse(condition, message);
        Logging.info(ASSERTION_PASS + "Assertion result is expected = FALSE - [" + message + "]");
    } catch (Throwable e) {
        Logging.error(failFormat(true, false, message));
        _addVerificationFailure(e);
    }
    }

    public static void softAssertNotNull(Object object, String message) {
    try {
        Assert.assertNotNull(object, message);
        Logging.info(passFormat(IS_NOT_NULL, message));
    } catch (Throwable e) {
        Logging.error(message + IS_NOT_NULL);
        _addVerificationFailure(e);
    }
    }

    public static void softAssertNotSame(Object actual, Object expected, String message) {
    try {
        Assert.assertNotSame(actual, expected, message);
        Logging.info(passFormat(actual, message));
    } catch (Throwable e) {
```

Read-Only | Smart Insert | 217 : 80

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class | AxisObjectUtil.class

```java
    }
    }

    public static void softAssertNotNull(Object object, String message) {
    try {
        Assert.assertNotNull(object, message);
        Logging.info(passFormat(IS_NOT_NULL, message));
    } catch (Throwable e) {
        Logging.error(message + IS_NOT_NULL);
        _addVerificationFailure(e);
    }
    }

    public static void softAssertNotSame(Object actual, Object expected, String message) {
    try {
        Assert.assertNotSame(actual, expected, message);
        Logging.info(passFormat(actual, message));
    } catch (Throwable e) {
        Logging.error(failFormat(actual, expected, message));
        _addVerificationFailure(e);
    }
    }

    public static void softAssertNull(Object object, String message) {
    try {
        Assert.assertNull(object, message);
        Logging.info(passFormat(IS_NULL, message));
    } catch (Throwable e) {
        Logging.error(message + IS_NULL);
        _addVerificationFailure(e);
    }
    }

    public static void softAssertSame(Object actual, Object expected, String message) {
    try {
        Assert.assertSame(actual, expected, message);
        Logging.info(passFormat(actual, message));
    } catch (Throwable e) {
        Logging.error(failFormat(actual, expected, message));
        _addVerificationFailure(e);
    }
    }

    public static void softAssertTrue(boolean condition, String message) {
    try {
        Assert.assertTrue(condition, message);
        Logging.info(ASSERTION_PASS + "Assertion result is expected - TRUE - [" + message + "]");
    } catch (Throwable e) {
        Logging.error(failFormat(false, true, message));
        _addVerificationFailure(e);
    }
    }

    public static void softAssertContains(String actual, String contained, String message) {
    try {
        if (StringUtils.isEmpty(actual)) {
        throw new AssertionError(message + IS_NULL);
        }
        Assert.assertTrue(actual.contains(contained), message);
        Logging.info(passFormat(actual, message));
    } catch (Throwable e) {
        Logging.error(message + COLON + " does not contain " + ASSERT_LEFT + contained + ASSERT_MIDDLE + actual + ASSERT_RIGHT);
        _addVerificationFailure(e);
    }
    }
}
```

Read-Only | Smart Insert | 217 : 80

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class | AxisObjectUtil.class | CSSUtil.class

```java
package com.ebay.maui.util;

 * CSS Selectors

import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.commons.lang.StringEscapeUtils;

import com.ebay.maui.exception.MauiException;

public class CSSUtil {

    private static final String NAME_PATTERN = "(\\w+)";
    private static final String VALUE_PATTERN = "([\\w|[\\p{Punct}&&[^'\"]]|\\s]+)";
    private static final String NODE_WITH_ATTRIBUTE = "(\\w+|\\*)\\s*(\\[(.*)\\])";
    private static final String OPERATOR_NAME_VALUE_PAIR_OR_FUNCTION = "\\s*(and|or)\\s*\\(?\\s*@" + NAME_PATTERN + "\\s*=\\s*['|\"]" + VALUE_PATTERN + "['|\"]|" + "\\s*(contains|starts-with|ends-with)\\s*\\(@" + NAM
        + VALUE_PATTERN + "['|\"]\\s*\\)|" + "\\s*text\\(\\)\\s*=\\s*['|\"]" + VALUE_PATTERN + "['|\"]\\)?)";

    public static String convertXPath2CSS(String xpathExpression) throws MauiException {

        if (!xpathExpression.startsWith("/"))
            throw new MauiException("Unable to convert xpath2css: " + xpathExpression);

        // Do not try to convert complex XPaths
        if (xpathExpression.indexOf("/@") != -1)
            throw new MauiException("Unable to convert xpath2css: " + xpathExpression);

        String[] tempXPathExpression = xpathExpression.split("\\|");
        StringBuffer sbCSS = new StringBuffer();
        for (int k = 0; k < tempXPathExpression.length; k++) {
            String[] tempXPath = tempXPathExpression[k].trim().split("//");
            for (int i = 0; i < tempXPath.length; i++) {
                String[] tempXPath2 = tempXPath[i].split("/");
                sbCSS.append(convertXPathNode2CSSNode(tempXPath2[0]));
                for (int j = 1; j < tempXPath2.length; j++) {
                    sbCSS.append(">").append(convertXPathNode2CSSNode(tempXPath2[j]));
                }
                sbCSS.append(" ");
            }
            sbCSS.deleteCharAt(sbCSS.length() - 1);
            sbCSS.append(",");
        }
        sbCSS.deleteCharAt(sbCSS.length() - 1);
        return sbCSS.toString().trim();
    }

    public static String convertXPathNode2CSSNode(String xpathExpression) throws MauiException {

        if (xpathExpression == null || xpathExpression.length() == 0)
            return "";

        Pattern p = Pattern.compile(NODE_WITH_ATTRIBUTE, Pattern.CASE_INSENSITIVE);
        Matcher m = p.matcher(xpathExpression);
        // System.out.println("NODE_WITH_ATTRIBUTE: " + xpathExpression);
        if (m.matches()) {
            String tagName = m.group(1);
            // System.out.println("NODE_WITH_ATTRIBUTE: " + xpathExpression + "----> " +tagName);
            StringBuffer sbCSS = new StringBuffer(tagName);
            if (3 <= m.groupCount()) {
                sbCSS.append("[");
                String xpathFunction = m.group(3); // That's the Function Group
                Pattern p3 = Pattern.compile(OPERATOR_NAME_VALUE_PAIR_OR_FUNCTION, Pattern.CASE_INSENSITIVE);
                Matcher m3 = p3.matcher(xpathFunction);

                // No need to convert XPaths with position //a[1]
                int position = -1;
```

Read-Only | Smart Insert | 1 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class | AxisObjectUtil.class | CSSUtil.class

```java
                Matcher m3 = p3.matcher(xpathFunction);

                // No need to convert XPaths with position //a[1]
                int position = -1;
                try {
                    position = Integer.parseInt(xpathFunction.trim());
                } catch (Exception e) {
                    // KEEP ME
                }
                if (position != -1)
                    throw new MauiException("Unable to convert xpath2css: " + xpathExpression);

                while (m3.find()) {

                    if ("and".equalsIgnoreCase(m3.group(1))) {
                        sbCSS.append("][");
                    } else if ("or".equalsIgnoreCase(m3.group(1))) {
                        sbCSS.append("],").append(tagName).append("[");
                    }

                    if (m3.group(3) != null) {
                        sbCSS.append(m3.group(3)).append("='").append(StringEscapeUtils.escapeHtml(m3.group(4))).append("'");
                    } else if (m3.group(5) != null) {
                        String verb = m3.group(5);
                        String value = StringEscapeUtils.escapeHtml(m3.group(7));
                        if ("contains".equalsIgnoreCase(verb)) {
                            sbCSS.append(m3.group(6)).append("*='").append(value).append("'");
                        } else if ("starts-with".equalsIgnoreCase(verb)) {
                            sbCSS.append(m3.group(6)).append("^='").append(value).append("'");
                        } else if ("ends-with".equalsIgnoreCase(verb)) {
                            sbCSS.append(m3.group(6)).append("$='").append(value).append("'");
                        } else {
                            throw new MauiException("Unable to convert xpath2css: " + xpathExpression);
                        }
                    } else if (m3.group(5) == null && (m3.group(6) != null || m3.group(7) != null)) {
                        throw new MauiException("Unable to convert xpath2css: " + xpathExpression);
                    } else if (m3.group(8) != null) {
                        sbCSS.insert(sbCSS.indexOf("["), ":contains('" + m3.group(8) + "')");
                    }

                    // System.out.println("\tOPERATOR_NAME_VALUE_PAIR: " + xpathFunction + "----> " + m3.group());
                    // for (int i = 1; i <= m3.groupCount(); i++) {
                    // System.out.println("\tOPERATOR_NAME_VALUE_PAIR: " + xpathFunction + "----> " + i + ": " + m3.group(i));
                    // }
                }

                if (sbCSS.charAt(sbCSS.length() - 1) == '[')
                    sbCSS.deleteCharAt(sbCSS.length() - 1);
                else
                    sbCSS.append("]");

                // System.out.println("\tOPERATOR_NAME_VALUE_PAIR: " + xpathFunction + "----> " + sbCSS.toString());

            }
            return sbCSS.toString();
        } else {
            if (xpathExpression.matches("(\\w+|\\*)") && !xpathExpression.matches("\\p{Punct}&&[^\\*]")) {
                return xpathExpression;
            } else {
                throw new MauiException("Unable to convert xpath2css: " + xpathExpression);
            }
        }

    }

}
```

Read-Only | Smart Insert | 1 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class | AxisObjectUtil.class | CSSUtil.class | CSVUtil.class

```java
package com.ebay.maui.util;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.TreeSet;

import org.apache.log4j.Logger;

import com.ebay.maui.controller.EasyFilter;
import com.ebay.maui.controller.Logging;
import com.ebay.maui.exception.MauiException;
import com.ebay.maui.util.internal.entity.TestObject;

public class CSVUtil {
    private static Logger logger = Logging.getLogger(CSVUtil.class);

    public static final String DOUBLE_QUOTE = "\"";
    public static final String DELIM_CHAR = ",";
    public static final String TAB_CHAR = " ";

    /**
     * Reads data from csv formatted file. Put the excel sheet in the same folder as the test case and specify clazz as <code>this.getClass()</code>.
     *
     * @param clazz
     * @param filename
     * @param fields
     * @param filter
     * @param readHeaders
     * @return
     * @throws Exception
     */
    public static Iterator<Object[]> getDataFromCSVFile(Class<?> clazz, String filename, String[] fields, EasyFilter filter, boolean readHeaders, boolean supportDPFilter) {
        return getDataFromCSVFile(clazz, filename, fields, filter, readHeaders, null, supportDPFilter);
    }

    public static Iterator<Object[]> getDataFromCSVFile(Class<?> clazz, String filename, String[] fields, EasyFilter filter, boolean readHeaders, String delimiter, boolean supportDPFilter) {

        InputStream is = null;
        try {
            if (clazz != null)
                is = clazz.getResourceAsStream(filename);
            else
                is = new FileInputStream(filename);

            if (is == null)
                return new ArrayList<Object[]>().iterator();

            // Get the sheet
            String[][] csvData = read(is, delimiter);

            List<Object[]> sheetData = new ArrayList<Object[]>();
            if (readHeaders) {
                List<Object> rowData = new ArrayList<Object>();
                if (fields == null) {
                    for (int j = 0; j < csvData[0].length; j++) {
```

Read-Only | Smart Insert | 1 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class | AxisObjectUtil.class | CSSUtil.class | CSVUtil.class

```java
                    rowData.add(csvData[0][j]);
                }
            } else {
                for (int i = 0; i < fields.length; i++) {
                    rowData.add(fields[i]);
                }
            }
            sheetData.add(rowData.toArray(new Object[rowData.size()]));
        }

        int testTitleColumnIndex = -1;
        int testSiteColumnIndex = -1;
        // Search for Title & Site column
        for (int i = 0; i < csvData[0].length; i++) {
            if (testTitleColumnIndex == -1 && TestObject.TEST_TITLE.equalsIgnoreCase(csvData[0][i])) {
                testTitleColumnIndex = i;
            } else if (testSiteColumnIndex == -1 && TestObject.TEST_SITE.equalsIgnoreCase(csvData[0][i])) {
                testSiteColumnIndex = i;
            }

            if (testTitleColumnIndex != -1 && testSiteColumnIndex != -1)
                break;
        }

        // Let's check for blank rows first
        // The first row is the header
        StringBuffer sbBlank = new StringBuffer();
        for (int i = 1; i < csvData.length; i++) {
            if (testTitleColumnIndex != -1 && testSiteColumnIndex != -1 && (csvData[i][testTitleColumnIndex].trim().length() == 0 || csvData[i][testSiteColumnIndex].trim().length() == 0)) {
                sbBlank.append(i + 1).append(',');
            }
        }
        if (sbBlank.length() > 0) {
            sbBlank.deleteCharAt(sbBlank.length() - 1);
            throw new MauiException("Blank TestTitle and/or Site value(s) found on Row(s) " + sbBlank.toString() + ".");
        }

        Set<String> uniqueDataSet = new TreeSet<String>();

        // Jerry: Add DataProviderTags filter
        /**
         * Modified by Gary to support include tags and exclude tags
         */
        if (supportDPFilter) {
            EasyFilter dpFilter = SpreadSheetUtil.getDPFilter();

            if (dpFilter != null) {
                if (filter == null) {
                    filter = dpFilter;
                } else {
                    filter = EasyFilter.and(filter, dpFilter);
                }
            }
        }
        // End
        // The first row is the header
        for (int i = 1; i < csvData.length; i++) {

            // Check for duplicate Title & Site
            if (testTitleColumnIndex != -1 && testSiteColumnIndex != -1) {
                String uniqueString = csvData[i][testTitleColumnIndex] + "$$$$####$$$$" + csvData[i][testSiteColumnIndex];
                if (uniqueDataSet.contains(uniqueString))
                    throw new MauiException("Duplicate TestTitle and Site combination found in the spreadsheet " + "with TestTitle = {" + csvData[i][testTitleColumnIndex] + "} " + "and Site = {" + csvData[i][testSi

                uniqueDataSet.add(uniqueString);
            }

```

Read-Only | Smart Insert | 1 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class | AxisObjectUtil.class | CSSUtil.class | CSVUtil.class

```java
            Map<String, Object> rowDataMap = new HashMap<String, Object>();
            List<Object> rowData = new ArrayList<Object>();

            // Create the mapping between headers and column data
            for (int j = 0; j < csvData[i].length; j++) {
                rowDataMap.put(csvData[0][j], csvData[i][j]);
            }

            if (fields == null) {
                for (int j = 0; j < csvData[0].length; j++) {
                    // Fix for null values not getting created when number of columns in a row is less than expected.
                    if (csvData[i].length > j)
                        rowData.add(csvData[i][j]);
                    else
                        rowData.add(null);
                }
            } else {
                for (int k = 0; k < fields.length; k++) {
                    rowData.add(SpreadSheetUtil.getValue(rowDataMap, fields[k]));
                }
            }

            // Jerry: Add DataProviderTags filter
            /**
             * Modified by Gary to support include tags and exclude tags
             */
            if (supportDPFilter) {
                SpreadSheetUtil.formatDPTags(rowDataMap);
            }
            // End
            if (filter == null || filter.match(rowDataMap)) {
                sheetData.add(rowData.toArray(new Object[rowData.size()]));
            }
        }

        if ((!readHeaders && sheetData.isEmpty()) || (readHeaders && sheetData.size() <= 1))
            logger.warn("No matching data found on csv file: " + filename + " with filter criteria: " + filter.toString());

        return sheetData.iterator();
    } catch (Throwable e) {
        throw new RuntimeException(e.getMessage());
    } finally {
        if (is != null) {
            try {
                is.close();
            } catch (Exception e) {
            }// KEEPME
        }
    }
}

/**
 * Get headers from a csv file
 * @param clazz - null means use the absolute file path, otherwise use relative path under the class
 * @param filename
 * @param delimiter - null means ","
 * @return
 */
public static ArrayList<String> getHeaderFromCSVFile(Class<?> clazz, String filename,String delimiter)
{
    if(delimiter==null)delimiter=",";

    InputStream is = null;
    try {
        if (clazz != null)
            is = clazz.getResourceAsStream(filename);
```

Read-Only | Smart Insert | 1 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class | AxisObjectUtil.class | CSSUtil.class | CSVUtil.class

```
200             is = clazz.getResourceAsStream(filename);
201         else
202             is = new FileInputStream(filename);
203
204         if (is == null)
205             return null;
206
207         // Get the sheet
208         String[][] csvData = read(is, delimiter);
209
210         ArrayList<String> rowData = new ArrayList<String>();
211
212         for (int j = 0; j < csvData[0].length; j++) {
213             rowData.add(csvData[0][j]);
214         }
215         return rowData;
216     } catch (Throwable e) {
217         throw new RuntimeException(e.getMessage());
218     } finally {
219         if (is != null) {
220             try {
221                 is.close();
222             } catch (Exception e) {
223             }// KEEPME
224         }
225     }
226
227 }
228
229 public static void main(String[] s) {
230     String line = "142843,\"testOneTimeCCMultiple,AccountPending\"," + "    Step 8. Verify user can make 1xCC payment for pending and active accounts " + "(for users with multiple \"accounts),US,,ecaf_seller_cc_pmt
231     String[] s2 = parseLine(line, DELIM_CHAR);
232     System.out.print(s2);
233 }
234
235 /**
236  * Parse a line
237  *
238  * @param line
239  * @param delim
240  * @return
241  */
242 public static String[] parseLine(String line, String delim) {
243     if (line == null || line.trim().length() == 0) {
244         return null;
245     }
246
247     List<String> tokenList = new ArrayList<String>();
248     String[] result = null;
249
250     String[] tokens = line.split(delim);
251     int count = 0;
252     while (count < tokens.length) {
253         if (tokens[count] == null || tokens[count].length() == 0) {
254             tokenList.add("");
255             count++;
256             continue;
257         }
258
259         if (tokens[count].startsWith(DOUBLE_QUOTE)) {
260             StringBuffer sbToken = new StringBuffer(tokens[count].substring(1));
261             while (count < tokens.length && !tokens[count].endsWith(DOUBLE_QUOTE)) {
262                 count++;
263                 sbToken.append(DELIM_CHAR).append(tokens[count]);
264             }
265             sbToken.deleteCharAt(sbToken.length() - 1);
266             tokenList.add(sbToken.toString());
```

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class | AxisObjectUtil.class | CSSUtil.class | CSVUtil.class

```
266             tokenList.add(sbToken.toString());
267         } else {
268             tokenList.add(tokens[count]);
269         }
270         count++;
271     }
272
273     if (tokenList.size() > 0) {
274         result = new String[tokenList.size()];
275         tokenList.toArray(result);
276     }
277     return result;
278
279 }
280
281 /**
282  * Parses an file and returns a String[][] object
283  *
284  * @param file
285  * @return
286  * @throws IOException
287  */
288 public static String[][] read(File file) throws IOException {
289     FileInputStream fis = new FileInputStream(file);
290     return read(fis);
291 }
292
293 public static String[][] read(InputStream is) throws IOException {
294     return read(is, null);
295 }
296
297 /**
298  * Parses an input stream and returns a String[][] object
299  *
300  * @param is
301  * @return
302  * @throws IOException
303  */
304 public static String[][] read(InputStream is, String delim) throws IOException {
305
306     String[][] result = null;
307     List<String[]> list = new ArrayList<String[]>();
308     String inputLine; // String that holds current file line
309
310     BufferedReader reader = new BufferedReader(new InputStreamReader(is,"UTF-8")); // KEEPME
311     while ((inputLine = reader.readLine()) != null) {
312         try {
313             String[] item = null;
314             if (delim == null)
315                 item = parseLine(inputLine, DELIM_CHAR);
316             else
317                 item = parseLine(inputLine, delim);
318             if (item != null)
319                 list.add(item);
320         } catch (Exception e) {// KEEPME
321         }
322     }
323     reader.close();
324
325     if (list.size() > 0) {
326         result = new String[list.size()][];
327         list.toArray(result);
328     }
329     return result;
330 }
331
332 /**
```

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListener.class | AxisObjectUtil.class | CSSUtil.class | CSVUtil.class

```
    }

    /**
     * Parses an file and returns a String[][] object
     *
     * @param file
     * @return
     * @throws IOException
     */
    public static String[][] read(File file) throws IOException {
        FileInputStream fis = new FileInputStream(file);
        return read(fis);
    }

    public static String[][] read(InputStream is) throws IOException {
        return read(is, null);
    }

    /**
     * Parses an input stream and returns a String[][] object
     *
     * @param is
     * @return
     * @throws IOException
     */
    public static String[][] read(InputStream is, String delim) throws IOException {

        String[][] result = null;
        List<String[]> list = new ArrayList<String[]>();
        String inputLine; // String that holds current file line

        BufferedReader reader = new BufferedReader(new InputStreamReader(is,"UTF-8")); // KEEPME
        while ((inputLine = reader.readLine()) != null) {
            try {
                String[] item = null;
                if (delim == null)
                    item = parseLine(inputLine, DELIM_CHAR);
                else
                    item = parseLine(inputLine, delim);
                if (item != null)
                    list.add(item);
            } catch (Exception e) {// KEEPME
            }
        }
        reader.close();

        if (list.size() > 0) {
            result = new String[list.size()][];
            list.toArray(result);
        }
        return result;
    }

    /**
     * Parses an URL and returns a String[][] object
     *
     * @param url
     * @return
     * @throws IOException
     */
    public static String[][] read(URL url) throws IOException {
        URLConnection con = url.openConnection();
        return read(con.getInputStream());
    }
}
```

Read-Only | Smart Insert | 1 : 1

MongoDBDriver.class | AUTH.class | APIUtil.class | ArrayUtil.class | AssertionUtil.class | AccessibilityListen | AxisObjectUtil.clas | CSSUtil.class | CSVUtil.class | DBUtil.class

```
package com.ebay.maui.util;

import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.ebay.maui.controller.Logging;
import com.ebay.maui.driver.db.DBDriver;
import com.ebay.maui.driver.db.DBDriver.DBHost;
import com.ebay.maui.exception.DBException;
import com.ebay.maui.exception.DBFinderException;
import com.ebay.maui.util.internal.type.DBHosts;

public class DBUtil {

    private static DBDriver driver = new DBDriver();

    /**
     * returns resultsSet As an Iterator.
     * @param query
     * @param columnNameList
     * @return resultsIterator
     */
    public synchronized static Iterator<String[]> getDBRecord(DBHost db,String query,
            List<String> columnNameList) {

        Iterator<String[]> resultsIterator = null;
        try {
            resultsIterator = driver.executeQuery(
                    db, query,
                    columnNameList);
        } catch (DBException e) {

            Logging.log("DBException thrown in DBUtils.getCaseInfo.." + e.getMessage());

        } catch (DBFinderException e) {

            Logging.log("DBFinderException thrown in DBUtils.getCaseInfo.."
                    + e.getMessage());

        }

        return resultsIterator;
    }


    /**
     * returns resultsSet As an Iterator.
     * @param query
     * @param columnNameList
     * @return resultsIterator
     */
    public synchronized static Iterator<String[]> getDBRecord(String query,
            List<String> columnNameList) {

        Iterator<String[]> resultsIterator = null;
        try {
            resultsIterator = driver.executeQuery(
                    DBHosts.StagingDB.DisputeHostPhysical, query,
                    columnNameList);
        } catch (DBException e) {

            Logging.log("DBException thrown in DBUtils.getCaseInfo.." + e.getMessage());

        } catch (DBFinderException e) {
```

Read-Only | Smart Insert | 1 : 1

```
66          } catch (DBFinderException e) {
67
68              Logging.log("DBFinderException throwm in DBUtils.getCaseInfo.."
69                      + e.getMessage());
70
71          }
72
73          return resultsIterator;
74      }
75
76
77      /**
78       * Returns the record set as a list
79       *
80       * @param query
81       * @param colNameList
82       * @return ArrayList
83       */
84      public synchronized static List<String[]> getDBRecordAsList(String query,
85              List<String> colNameList) {
86          Iterator<String[]> resIterator = getDBRecord(query, colNameList);
87          List<String[]> holdList = new ArrayList<String[]>();
88          while (resIterator.hasNext()) {
89              holdList.add(resIterator.next());
90          }
91          return holdList;
92      }
93      /**
94       * Retrieves a DBHost based on dbEnv and dbHostName <code>
95       * DBHost dbHost = getDBHost(DBHosts.StagingDB.class, "AAIAConfigHostPhysical");
96       * </code>
97       *
98       * @param dbEnv
99       * @param dbHostName
100      * @return
101      * @throws DBException
102      */
103     public static DBHost getEBayDBHost(Class<?> dbEnv, String dbHostName) throws DBException {
104
105         if (!dbEnv.getEnclosingClass().equals(DBHosts.class)) {
106             throw new DBException(dbEnv.getCanonicalName() + " is not a valid dbEnvironment.\n" + " Valid values are {DBHosts.StagingDB.class, DBHosts.Baby.class, " + "DBHosts.LPDBSAN.class & DBHosts.Wagner.class}");
107         }
108
109         Field[] dbHosts = dbEnv.getDeclaredFields();
110
111         for (int i = 0; i < dbHosts.length; i++) {
112             if (dbHosts[i].getName().equalsIgnoreCase(dbHostName)) {
113                 try {
114                     return (DBHost) dbHosts[i].get(null);
115                 } catch (Exception e) {
116                     throw new DBException("Ex", e);
117                 }
118             }
119         }
120
121         throw new DBException("No DBHost found with env: " + dbEnv.getCanonicalName() + " and dbHostName: " + dbHostName);
122     }
123
124     /**
125      * Returns specific record set from the resultsSet. Returns the first row when required
126      * record not found.
127      *
128      * @param query
129      * @param colNameList
130      * @return String array
131      */
132     public synchronized static String[] getSpecificDBRec(String query, int iDbRow)
```

```
80      * @param query
81      * @param colNameList
82      * @return ArrayList
83      */
84      public synchronized static List<String[]> getDBRecordAsList(String query,
85              List<String> colNameList) {
86          Iterator<String[]> resIterator = getDBRecord(query, colNameList);
87          List<String[]> holdList = new ArrayList<String[]>();
88          while (resIterator.hasNext()) {
89              holdList.add(resIterator.next());
90          }
91          return holdList;
92      }
93      /**
94       * Retrieves a DBHost based on dbEnv and dbHostName <code>
95       * DBHost dbHost = getDBHost(DBHosts.StagingDB.class, "AAIAConfigHostPhysical");
96       * </code>
97       *
98       * @param dbEnv
99       * @param dbHostName
100      * @return
101      * @throws DBException
102      */
103     public static DBHost getEBayDBHost(Class<?> dbEnv, String dbHostName) throws DBException {
104
105         if (!dbEnv.getEnclosingClass().equals(DBHosts.class)) {
106             throw new DBException(dbEnv.getCanonicalName() + " is not a valid dbEnvironment.\n" + " Valid values are {DBHosts.StagingDB.class, DBHosts.Baby.class, " + "DBHosts.LPDBSAN.class & DBHosts.Wagner.class}");
107         }
108
109         Field[] dbHosts = dbEnv.getDeclaredFields();
110
111         for (int i = 0; i < dbHosts.length; i++) {
112             if (dbHosts[i].getName().equalsIgnoreCase(dbHostName)) {
113                 try {
114                     return (DBHost) dbHosts[i].get(null);
115                 } catch (Exception e) {
116                     throw new DBException("Ex", e);
117                 }
118             }
119         }
120
121         throw new DBException("No DBHost found with env: " + dbEnv.getCanonicalName() + " and dbHostName: " + dbHostName);
122     }
123
124     /**
125      * Returns specific record set from the resultsSet. Returns the first row when required
126      * record not found.
127      *
128      * @param query
129      * @param colNameList
130      * @return String array
131      */
132     public synchronized static String[] getSpecificDBRec(String query, int iDbRow)
133     {
134         List<String[]> holdList = new ArrayList<String[]>();
135         List<String> colNameList = new ArrayList<String>();
136         holdList=getDBRecordAsList(query, colNameList);
137
138         int holdListSize = holdList.size();
139         if (holdListSize!=0 && holdListSize > iDbRow)
140             return  holdList.get(iDbRow);
141         else
142             return holdList.get(0);
143     }
144
145 }
146
```

APIUtil.class | ArrayUtil.class | AssertionUtil.c | AccessibilityLi | AxisObjectUtil. | CSSUtil.class | CSVUtil.class | DBUtil.class | EscapeHelper.cl | ExpectedConditi

```java
package com.ebay.maui.util;

import org.openqa.selenium.By;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.StaleElementReferenceException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.support.ui.ExpectedCondition;

public class ExpectedConditionsUtil {
    /**
     * An Expectation for checking that an element is either invisible or not
     * present on the DOM.
     *
     * @param locator used to find the element
     */
    public static ExpectedCondition<Boolean> activeOfElementLocated(
      final By locator) {
      return new ExpectedCondition<Boolean>() {
        public Boolean apply(WebDriver driver) {
          try {
            return driver.findElement(locator).getAttribute("class").equalsIgnoreCase("active");
          } catch (NoSuchElementException e) {
            // Returns true because the element is not present in DOM. The
            // try block checks if the element is present but is invisible.
            return true;
          } catch (StaleElementReferenceException e) {
            // Returns true because stale element reference implies that element
            // is no longer visible.
            return true;
          }
        }
      };
    }

    /**
     * An Expectation for checking that an element is either invisible or not
     * present on the DOM.
     *
     * @param locator used to find the element
     */
    public static ExpectedCondition<Boolean> inactiveOfElementLocated(
      final By locator) {
      return new ExpectedCondition<Boolean>() {
        public Boolean apply(WebDriver driver) {
          try {
            return driver.findElement(locator).getAttribute("class").equalsIgnoreCase("inactive");
          } catch (NoSuchElementException e) {
            // Returns true because the element is not present in DOM. The
            // try block checks if the element is present but is invisible.
            return true;
          } catch (StaleElementReferenceException e) {
            // Returns true because stale element reference implies that element
            // is no longer visible.
            return true;
          }
        }
      };
    }

}
```

Read-Only | Smart Insert | 1 : 1

AccessibilityLi | AxisObjectUtil. | CSSUtil.class | CSVUtil.class | DBUtil.class | EscapeHelper.cl | ExpectedConditi | FileTransferAPI | FileUtil.class | GrepUtil.class

```java
package com.ebay.maui.util;

import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.UnsupportedEncodingException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.charset.Charset;
import java.util.Enumeration;
import java.util.Map.Entry;
import java.util.Properties;
import java.util.zip.ZipEntry;
import java.util.zip.ZipFile;

import javax.activation.DataHandler;
import javax.activation.FileDataSource;
import javax.mail.MessagingException;
import javax.mail.internet.MimeBodyPart;

import org.apache.commons.httpclient.Header;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.methods.StringRequestEntity;
import org.apache.commons.httpclient.protocol.Protocol;
import org.apache.http.HttpStatus;

import com.ebay.kernel.util.Base64;
import com.ebay.maui.controller.ContextManager;
import com.ebay.maui.controller.Logging;
import com.ebay.maui.driver.api.APIDriver;
import com.ebay.maui.driver.api.EasySSLProtocolSocketFactory;
import com.ebay.maui.exception.APIException;
import com.ebay.maui.util.internal.type.APIType;


public class FileTransferAPIHelper {

    public static ByteArrayInputStream executeAPICallWithAttachment(String endPoint, APIType apiType, Properties headers, String xmlInput) throws Exception{
        Properties inputHeaders = new Properties();
        Properties responseHeaders = new Properties();
        inputHeaders.putAll(headers);

        ByteArrayInputStream[] response = new ByteArrayInputStream[1];

        int responseCode = 0;
        try {
            responseCode = executeCall(endPoint, inputHeaders, xmlInput, responseHeaders, response);
        } catch (Exception e1) {
            if (e1 instanceof APIException) {
                throw (APIException) e1;
            }
            throw new APIException(apiType + " Call returned error:", e1);
        }
        //String output = streamToString(response[0]);
        try{
            ByteArrayInputStream[]  message = parseResponseContent(response[0]);

            String m0 = streamToString(message[0],Charset.defaultCharset().name());

            //Logging.logAPIStep(apiType.name(), endPoint, inputHeaders, xmlInput, responseHeaders, m0, responseCode);
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class

```
66
67          //logging.logAPIStep(apiType.name(), endPoint, inputHeaders, xmlInput, responseHeaders, m0, responseCode);
68          APIDriver.generateLogMessage(endPoint, inputHeaders, xmlInput, responseHeaders, m0, null, null);
69
70          if (responseCode != HttpStatus.SC_OK) {
71              throw new APIException(apiType + " Call returned error: \n" + m0);
72          }
73
74          return message[1];
75
76      }
77      catch(Exception ex)
78      {
79          //log(output);
80          throw ex;
81      }
82  }
83
84  /**
85   * Use this method to execute API with File attachment
86   * You need to specify %FileSize1% and %FileData1% to store the size and data for the attachment
87   * @author junjshi
88   * @date 6/22/2011
89   *
90   * @param endPoint
91   * @param apiType
92   * @param headers
93   * @param xmlInput
94   * @param file
95   * @return
96   * @throws Exception
97   * @throws MessagingException
98   * @throws MessagingException
99   * @throws MessagingException
100  * @throws MessagingException
101  */
102 public static String executeAPIWithFile(String endPoint,APIType apiType,Properties headers,String xmlInput,File[] file) throws Exception
103 {
104     for(int i=0;i<file.length;i++)
105     {
106         DataHandler dh = new DataHandler(new FileDataSource(file[i]));
107
108         byte[] fileInfo = null;
109         if (file != null) {
110          FileInputStream fis = (FileInputStream)dh.getContent();
111          if (fis != null) {
112           int len = fis.available();
113           fileInfo = new byte[len];
114           fis.read(fileInfo);
115          }
116
117         }
118         String encodeStr = Base64.encode(fileInfo).toString();
119
120         xmlInput = xmlInput.replaceAll("%FileSize"+(i+1)+"%", file[i].length()+"");
121         xmlInput = xmlInput.replaceAll("%FileData"+(i+1)+"%", encodeStr);
122     }
123
124     //headers.setProperty("Accept-Encoding", "gzip,deflate");
125     headers.setProperty("Content-Type", "application/soap+xml;charset=UTF-8");
126     //return executeAPICall(endPoint, apiType, headers, xmlInput);
127     Properties responseHeaders = new Properties();
128     ByteArrayInputStream[] response = new ByteArrayInputStream[1];
129     int responseCode = 0;
130     try {
131         responseCode = executeCall(endPoint, headers, xmlInput, responseHeaders, response);
132     } catch (Exception e1) {
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class

```
132     } catch (Exception e1) {
133         if (e1 instanceof APIException) {
134             throw (APIException) e1;
135         }
136         throw new APIException(apiType + " Call returned error:", e1);
137     }
138
139     try{
140         String m0 = streamToString(response[0],"UTF-8");
141
142         //logAPIStep(apiType.name(), endPoint, headers, xmlInput, responseHeaders, m0, responseCode);
143         APIDriver.generateLogMessage(endPoint, headers, xmlInput, responseHeaders, m0, null, null);
144
145
146         if (responseCode != HttpStatus.SC_OK) {
147             throw new APIException(apiType + " Call returned error: \n" + m0);
148         }
149
150         return m0;
151
152     }
153     catch(Exception ex)
154     {
155         //log(output);
156         throw ex;
157     }
158
159
160 }
161
162 @SuppressWarnings("deprecation")
163     public static int executeCall(String serverURL, Properties requestHeaders, String requestXML, Properties responseHeaders, ByteArrayInputStream[] sbResponseXML) throws APIException {
164
165         if (requestXML == null)
166             throw new APIException("Input XML can't be null");
167
168
169         requestXML = requestXML.replace("\n", "");
170         requestXML = requestXML.replace("\r", "");
171         requestXML = requestXML.trim() + "\n";
172
173         HttpClient httpclient = new HttpClient();
174         httpclient.getHttpConnectionManager().getParams().setConnectionTimeout(ContextManager.getThreadContext().getWebSessionTimeout());
175
176         if (serverURL.toLowerCase().startsWith("https://")) {
177             String server = serverURL.substring(serverURL.indexOf("://") + 3, serverURL.indexOf(".com") + 4);
178             httpclient.getHostConfiguration().setHost(server, 443, new Protocol("https", new EasySSLProtocolSocketFactory(), 443));
179         }
180
181         PostMethod method = new PostMethod(serverURL);
182         if (requestHeaders != null) {
183             for (Entry<Object, Object> header : requestHeaders.entrySet()) {
184                 method.setRequestHeader((String) header.getKey(), (String) header.getValue());
185             }
186         }
187
188         // If soap request and request header is missing SOAPAction add it.
189         if (requestXML.indexOf("soapenv:Envelope") != -1 && (requestHeaders == null || !requestHeaders.containsKey("SOAPAction"))) {
190             method.setRequestHeader("SOAPAction", "");
191         }
192
193         BufferedReader br = null;
194
195         try {
196             String charSet = "UTF-8";
197             method.setRequestEntity(new StringRequestEntity(requestXML, (requestHeaders != null && requestHeaders.containsKey("Content-Type") ? requestHeaders.getProperty("Content-Type") : "text/xml"), charSet));
198
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class

```java
200                 int returnCode = httpclient.executeMethod(method);
201
202                 int len = method.getResponseBody().length;
203                 //Change the method to get response by bytes. Don't use string to get the response. It will break the structure.
204                 byte[] content = new byte[len];
205                 content = method.getResponseBody();
206                 sbResponseXML[0] = new ByteArrayInputStream(content);
207
208                 if (responseHeaders != null) {
209                     Header[] respHeaders = method.getResponseHeaders();
210                     responseHeaders.clear();
211                     for (int i = 0; i < respHeaders.length; i++) {
212                         responseHeaders.setProperty(respHeaders[i].getName(), respHeaders[i].getValue());
213                     }
214                 }
215
216                 return returnCode;
217             } catch (Exception e) {
218                 e.printStackTrace();
219                 throw new APIException(e);
220             } finally {
221                 method.releaseConnection();
222                 if (br != null)
223                     try {
224                         br.close();
225                     } catch (Exception fe) {
226                     }
227             }
228         }
229
230
231     public static File getFile(Class<?> clazz, String fileName) throws URISyntaxException
232     {
233         URL url = clazz.getResource(fileName);
234         return new File(url.toURI());
235     }
236
237     private static byte[] getSubString(byte[] buf, int fromidx,int length)
238     {
239         int i;
240
241             byte[] substr = new byte[length];
242             for (i = fromidx; i < buf.length && i < fromidx + substr.length; i++) {
243                 substr[i - fromidx] = buf[i];
244             }
245
246         return substr;
247     }
248
249     private static int indexOf(byte[] buf, byte[] substr, int fromidx)
250     {
251         //search "\r\n" from fromidx(0:the beginning of the bytes
252         int i, j;
253         for (i = fromidx; i < buf.length - substr.length; i++) {
254             for (j = 0; j < substr.length; j++) {
255                 if (substr[j] != buf[i+j]) {
256                     break;
257                 }
258             }
259             if (j == substr.length) {
260                 //System.out.println("find " + Hex.encodeHex(substr) + "at" + i);
261                 return i;
262             }
263         }
264         return -1;
265     }
266
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class

```java
266
267     private static byte[] mergeString( byte[] str1, byte[] str2)
268     {
269         byte[] outstr = new byte[str1.length+str2.length];
270
271         int i, j;
272         for (i = 0; i < str1.length; i++) {
273             outstr[i] = str1[i];
274         }
275         for (j = 0; j < str2.length; j++) {
276             outstr[i + j] = str2[j];
277         }
278         return outstr;
279     }
280
281
282     private static ByteArrayInputStream[] parseResponseContent(ByteArrayInputStream response) throws  IOException, MessagingException {
283         int len = response.available();
284         byte[] buf = new byte[len];
285         response.read(buf);
286         System.out.println("buf:"+buf);
287         int index1, index2, index3;
288         byte[] newLineFlag = { '\r', '\n' };
289         index1 = indexOf(buf, newLineFlag, 0);//i: boundary's length
290         System.out.println(buf.toString());
291         byte[] boundary = getSubString(buf, 0,index1);//--MIMEBoundaryurn_uuid_255781F55B6AA7446A1323053069787478 1--
292
293         byte[] nboundary = mergeString( newLineFlag, boundary);
294         byte[] nboundarynr = mergeString( nboundary, newLineFlag);//\n\r--MIMEBoundaryurn_uuid_255781F55B6AA7446A1323053069787478 1--
295         ByteArrayInputStream[] contents = new ByteArrayInputStream[2];//\n\r--MIMEBoundaryurn_uuid_255781F55B6AA7446A1323053069787478 1--\n\r
296
297         index2 = indexOf(buf, nboundarynr, index1+1);
298         byte[] part1,part2;
299         if(index2>0)
300         {
301             part1 = getSubString(buf, index1+2,index2 - index1 - 2);
302             index3 = indexOf(buf, nboundary, index2+1);
303             part2 = getSubString(buf, index2+nboundarynr.length,index3 - index2 - nboundarynr.length);
304
305             ByteArrayInputStream ins = new ByteArrayInputStream(part1);
306             MimeBodyPart part11 = new MimeBodyPart(ins);
307             ByteArrayInputStream in = (ByteArrayInputStream)part11.getContent();
308
309             contents[0] = in;
310
311             ByteArrayInputStream ins2 = new ByteArrayInputStream(part2);
312             MimeBodyPart part12 = new MimeBodyPart(ins2);
313             ByteArrayInputStream in2 = (ByteArrayInputStream)part12.getContent();
314
315             contents[1] = in2;
316         }else
317         {
318             index2 = indexOf(buf,"<?xml".getBytes(),index1+1);
319             part1 = getSubString(buf,index1+2,index2-index1-2);
320             index3 = indexOf(buf,nboundary,index2+1);
321             part2 = getSubString(buf,index2,index3-index2);
322
323             ByteArrayInputStream ins = new ByteArrayInputStream(part1);
324             MimeBodyPart part11 = new MimeBodyPart(ins);
325             ByteArrayInputStream in = (ByteArrayInputStream)part11.getContent();
326
327             contents[1] = in;
328
329             ByteArrayInputStream ins2 = new ByteArrayInputStream(part2);
330
331
332             contents[0] = ins2;
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class

```
            ByteArrayInputStream in = (ByteArrayInputStream)part11.getContent();

            contents[1] = in;

            ByteArrayInputStream ins2 = new ByteArrayInputStream(part2);


            contents[0] = ins2;
        }



         return contents;


        }

    private static String streamToString(ByteArrayInputStream stream,String charsetName) throws UnsupportedEncodingException
    {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        int j=-1;
        while ((j = stream.read()) != -1) {
            baos.write(j);
        }

        String m0 = baos.toString(charsetName);
        return m0;
    }


    /*
     * added to unzip the attachment.
     * inFilename: the zip file
     * direc: the folder which save the unziped files.
     */
    public static File unzipFile(String inFilename,String direc) throws FileNotFoundException, IOException {
        if(!direc.endsWith("\\")||!direc.endsWith("/"))
            direc = direc + "\\";
        File file = new File(inFilename);
        File temFile = null;
        int leng = 0;
        byte[] b = new byte[1024];
        ZipFile zipFile = new ZipFile(file);
        for (Enumeration<?> enume = zipFile.entries(); enume.hasMoreElements();)
        {
            ZipEntry entry = (ZipEntry) enume.nextElement();
            temFile = new File(direc + entry.getName());
            FileOutputStream fos = new FileOutputStream(temFile);
            InputStream  zis = zipFile.getInputStream(entry);
            //inputStream = zipFile.getInputStream(entry);
            while ((leng = zis.read(b)) != -1)
            {
                fos.write(b, 0, leng);
            }
            fos.close();
            zis.close();
            Logging.log("extract  to "+temFile);
        }
           return temFile;
        }

}
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class

```
package com.ebay.maui.util;

import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.StringWriter;
import java.util.List;
import java.util.Map;

import org.apache.log4j.Logger;
import org.apache.velocity.VelocityContext;
import org.apache.velocity.app.VelocityEngine;

import com.ebay.maui.controller.ContextManager;
import com.ebay.maui.helper.FileHelper;
import com.ebay.maui.helper.URLHelper;
import com.ebay.maui.util.internal.entity.GrepResult;

public class FileUtil {
    static Logger logger = Logger.getLogger(FileUtil.class);

    public static void copyFile(File srcPath, File dstPath) throws IOException {
        FileHelper.copyFile(srcPath, dstPath);
    }

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

    /**
     * Deletes a Directory with Files
     *
     * @param path
     * @return
     */
    public static boolean deleteDirectory(File path) {
        return FileHelper.deleteDirectory(path);
    }

    /**
     * Get content of an file and apply any changes through the properties. The template file can look like <code>
     * <pre>
     * <xmp>
     * <?xml version="1.0" encoding="utf-8"?>
     * <GeteBayOfficialTimeRequest xmlns="urn:ebay:apis:eBLBaseComponents">
     * <Version>547</Version>
     * <RequesterCredentials>
     * <Username>$username</Username>
     * <Password>$password</Password>
     * </RequesterCredentials>
     * </GeteBayOfficialTimeRequest>
     * </xmp>
     * </pre>
     * </code>
     *
     * Define a Properties object in your test case <code>
     * <pre>
     * Properties prop = new Properties();
     * prop.put("username","apitest1");
     * prop.put("password","ebay");
     * </pre>
     * </code>
     *
     * The output of this method will be <code>
     * <pre>
     * <xmp>
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class

```
     * <pre>
     * <xmp>
     * <?xml version="1.0" encoding="utf-8"?>
     * <GeteBayOfficialTimeRequest xmlns="urn:ebay:apis:eBLBaseComponents">
     * <Version>547</Version>
     * <RequesterCredentials>
     * <Username>apitest1</Username>
     * <Password>ebay</Password>
     * </RequesterCredentials>
     * </GeteBayOfficialTimeRequest>
     * </xmp>
     * </pre>
     * </code>
     *
     * @param TestClass
     * @param templateName
     * @param prop
     * @return
     */
    public static String getFileContent(Class<?> clazz, String templateName, Map<?, ?> prop) {
        try {
            if (prop == null || prop.isEmpty())
                return FileUtil.readFromFile(clazz.getResourceAsStream(templateName));
            else {
                VelocityEngine ve = new VelocityEngine();
                ve.setProperty("resource.loader", "class");
                ve.setProperty("class.resource.loader.class", "org.apache.velocity.runtime.resource.loader.ClasspathResourceLoader");
                ve.init();

                VelocityContext context = null;
                context = new VelocityContext(prop);
                StringWriter writer = new StringWriter();
                templateName = "/" + clazz.getPackage().getName().replace('.', '/') + "/" + templateName;
//              if (ve.mergeTemplate(templateName,"utf-8", context, writer)) {
//                  return writer.toString();
//              } else {
//                  throw new RuntimeException("Unable to get file content for " + templateName);
//              }
                InputStreamReader reader = new InputStreamReader(clazz.getResourceAsStream(templateName),"UTF-8");
                boolean success = ve.evaluate(context, writer, templateName, reader);
                if (success) {
                    return writer.toString();
                } else {
                    throw new RuntimeException("Unable to get file content for " + templateName);
                }
            }
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    /**
     * Search the file for a given pattern. Search result will return all occurrences of a given pattern.
     *
     * For example:
     *
     * 1. search a file for a line that starts with any alphabet grepTextFromFile("c:/tmp/autoidcore.txt", "^[a-z]"); 2. search a file for an item number grepTextFromFile("c:/tmp/autoidcore.txt", "2000081515");
     *
     * @param filename
     *            - full path to the file
     * @param pattern
     *            - text to seach. This can be a plain text or a regular expression
     * @return
     */
    public static List<GrepResult> grepTextFromFile(String filename, String pattern) {
        List<GrepResult> gr = null;
        try {
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class

```
        try {
            File f = new File(filename);
            gr = GrepUtil.grep(f, pattern);
        } catch (Exception e) {
            throw new RuntimeException("Unable to grep text from file. ", e);
        }

        return gr;
    }


    /**
     * Read contents of a file
     *
     * @param path
     * @returns content
     * @throws IOException
     */
    public static String readFromFile(File path) throws IOException {
        return FileHelper.readFromFile(path);
    }

    /**
     * Read contents From Stream
     *
     * @param path
     * @returns content
     * @throws IOException
     */
    public static String readFromFile(InputStream path) throws IOException {
        return FileHelper.readFromFile(path);
    }

    /**
     * Read contents of a file
     *
     * @param path
     * @returns content
     * @throws IOException
     */
    public static String readFromFile(String path) throws IOException {
        return readFromFile(new File(path));
    }

    /**
     * Write content to a file. Randomizes the file prefix to avoid collision of test cases when executed in parallel mode.
     *
     * @param fileNameWithExtnPostfix
     * @param content
     * @return relative file path
     */
    public static String saveToFile(String fileNameWithExtnPostfix, String content) {

        String folder = null;
        String extn = fileNameWithExtnPostfix.toLowerCase().split("\\.")[1];
        if (extn.startsWith("htm"))
            folder = "/htmls/";
        else if (extn.startsWith("xml"))
            folder = "/xmls/";
        else if (extn.equals("png") || extn.equals("jpg") || extn.equals("jpeg") || extn.equals("gif"))
            folder = "/screenshots/";
        else
            folder = "/textfiles/";

        String path = folder + URLHelper.getRandomHashCode(folder) + "_" + fileNameWithExtnPostfix;
        try {
            FileUtil.writeToFile(ContextManager.getThreadContext().getOutputDirectory() + path, content);
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class

```java
        return readFromFile(new File(path));
    }

    /**
     * Write content to a file. Randomizes the file prefix to avoid collision of test cases when executed in parallel mode.
     *
     * @param fileNameWithExtnPostfix
     * @param content
     * @return relative file path
     */
    public static String saveToFile(String fileNameWithExtnPostfix, String content) {

        String folder = null;
        String extn = fileNameWithExtnPostfix.toLowerCase().split("\\.")[1];
        if (extn.startsWith("htm"))
            folder = "/htmls/";
        else if (extn.startsWith("xml"))
            folder = "/xmls/";
        else if (extn.equals("png") || extn.equals("jpg") || extn.equals("jpeg") || extn.equals("gif"))
            folder = "/screenshots/";
        else
            folder = "/textfiles/";

        String path = folder + URLHelper.getRandomHashCode(folder) + "_" + fileNameWithExtnPostfix;
        try {
            fileUtil.writeToFile(ContextManager.getThreadContext().getOutputDirectory() + path, content);
        } catch (IOException e) {
            throw new RuntimeException("Unable to write file content to " + ContextManager.getThreadContext().getOutputDirectory() + path);
        }
        return ContextManager.getThreadContext().getTestNGContext().getSuite().getName() + path;
    }

    /**
     * Creates Image from bytes and stores it
     *
     * @param path
     * @param screenShot
     */
    public static synchronized void writeImage(String path, byte[] byteArray) {
        FileHelper.writeImage(path, byteArray);
    }

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

    /**
     * Saves HTML Source
     *
     * @param path
     * @param selenium
     * @throws Exception
     */

    public static void writeToFile(String path, String content) throws IOException {
        FileHelper.writeToFile(path, content);
    }

}
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class | HttpUtil.class

```java
package com.ebay.maui.util;

import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Properties;

/**
 * This class provides HTTP URL relative methods.
 *
 */
public class HttpUtil {

    private static final int[] SUCCESSFUL_STATUS_CODE = { HttpURLConnection.HTTP_OK, HttpURLConnection.HTTP_CREATED, HttpURLConnection.HTTP_ACCEPTED };
    private static final int[] REDIRECTION_STATUS_CODE = { HttpURLConnection.HTTP_MULT_CHOICE, HttpURLConnection.HTTP_MOVED_PERM,
            HttpURLConnection.HTTP_MOVED_TEMP };
    private static final int[] SUCCESSFUL_REDIRECTION_STATUS_CODE = { HttpURLConnection.HTTP_OK, HttpURLConnection.HTTP_CREATED,
            HttpURLConnection.HTTP_ACCEPTED, HttpURLConnection.HTTP_MULT_CHOICE, HttpURLConnection.HTTP_MOVED_PERM, HttpURLConnection.HTTP_MOVED_TEMP };
    private static final int TIME_OUT = 10000;

    /**
     * Get HTTP status code when open a URL connection.
     * http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html
     *
     * @param url
     * @param enableProxy
     * @return HTTP status code
     */
    public static int getHTTPStatusCode(String url, boolean enableProxy) {
        int statusCode = 0;
        if (enableProxy) {
            Properties props = System.getProperties();
            props.setProperty("http.proxyHost", "qa-proxy.qa.ebay.com");
            props.setProperty("http.proxyPort", "80");
        }

        try {
            HttpURLConnection connection = (HttpURLConnection) new URL(url).openConnection();
            connection.setConnectTimeout(TIME_OUT);
            connection.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1");
            connection.connect();
            statusCode = connection.getResponseCode();
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        return statusCode;
    }

    /**
     * Get HTTP status code when open a URL connection.
     * http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html
     *
     * @param url
     * @return HTTP status code
     */
    public static int getHTTPStatusCode(String url) {
        return getHTTPStatusCode(url, false);
    }

    /**
     * Validates HTTP response by status code
     *
     * @param url
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class | HttpUtil.class

```
67      * @param url
68      * @param enableProxy
69      * @param statusCodeArray
70      * @return true or false
71      */
72     public static boolean validateHTTPResponse(String url, boolean enableProxy, int[] statusCodeArray) {
73         int statusCode = getHTTPStatusCode(url, enableProxy);
74
75         for (int i = 0; i < statusCodeArray.length; i++) {
76             if (statusCode == statusCodeArray[i]) {
77                 return true;
78             }
79         }
80
81         return false;
82     }
83
84     /**
85      * Validates HTTP response by status code
86      *
87      * @param url
88      * @param statusCodeArray
89      * @return true or false
90      */
91     public static boolean validateHTTPResponse(String url, int[] statusCodeArray) {
92         return validateHTTPResponse(url, false, statusCodeArray);
93     }
94
95     /**
96      * Validates successful HTTP response
97      *
98      * @param url
99      * @return true or false
100     */
101    public static boolean validateSuccessfulHTTPResponse(String url) {
102        return validateHTTPResponse(url, SUCCESSFUL_STATUS_CODE);
103    }
104
105    /**
106     * Validates redirection HTTP response
107     *
108     * @param url
109     * @return true or false
110     */
111    public static boolean validateRedirectionHTTPResponse(String url) {
112        return validateHTTPResponse(url, REDIRECTION_STATUS_CODE);
113    }
114
115    /**
116     * Validates correct HTTP response include successful and redirection
117     * response
118     *
119     * @param url
120     * @return true or false
121     */
122    public static boolean validateHTTPResponse(String url) {
123        return validateHTTPResponse(url, SUCCESSFUL_REDIRECTION_STATUS_CODE);
124    }
125
126    /**
127     * Validates successful HTTP response
128     *
129     * @param url
130     * @param enableProxy
131     * @return true or false
132     */
133    public static boolean validateSuccessfulHTTPResponse(String url, boolean enableProxy) {
```

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class | HttpUtil.class

```
94
95     /**
96      * Validates successful HTTP response
97      *
98      * @param url
99      * @return true or false
100     */
101    public static boolean validateSuccessfulHTTPResponse(String url) {
102        return validateHTTPResponse(url, SUCCESSFUL_STATUS_CODE);
103    }
104
105    /**
106     * Validates redirection HTTP response
107     *
108     * @param url
109     * @return true or false
110     */
111    public static boolean validateRedirectionHTTPResponse(String url) {
112        return validateHTTPResponse(url, REDIRECTION_STATUS_CODE);
113    }
114
115    /**
116     * Validates correct HTTP response include successful and redirection
117     * response
118     *
119     * @param url
120     * @return true or false
121     */
122    public static boolean validateHTTPResponse(String url) {
123        return validateHTTPResponse(url, SUCCESSFUL_REDIRECTION_STATUS_CODE);
124    }
125
126    /**
127     * Validates successful HTTP response
128     *
129     * @param url
130     * @param enableProxy
131     * @return true or false
132     */
133    public static boolean validateSuccessfulHTTPResponse(String url, boolean enableProxy) {
134        return validateHTTPResponse(url, enableProxy, SUCCESSFUL_STATUS_CODE);
135    }
136
137    /**
138     * Validates redirection HTTP response
139     *
140     * @param url
141     * @param enableProxy
142     * @return true or false
143     */
144    public static boolean validateRedirectionHTTPResponse(String url, boolean enableProxy) {
145        return validateHTTPResponse(url, enableProxy, REDIRECTION_STATUS_CODE);
146    }
147
148    /**
149     * Validates correct HTTP response include successful and redirection
150     * response
151     *
152     * @param url
153     * @param enableProxy
154     * @return true or false
155     */
156    public static boolean validateHTTPResponse(String url, boolean enableProxy) {
157        return validateHTTPResponse(url, enableProxy, SUCCESSFUL_REDIRECTION_STATUS_CODE);
158    }
159 }
160
```

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class | HttpUtil.class | JsonDataProviderUtil.class

```java
* @author junjshi, created on Aug 7, 2013
package com.ebay.maui.util;

import java.io.IOException;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.LinkedList;
import java.util.Map;
import java.util.Map.Entry;

import jxl.common.Logger;

import com.ebay.maui.controller.EasyFilter;
import com.google.gson.Gson;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;

/**
 * This class is used to do data provider for Json
 * @author junjshi
 *
 */
public class JsonDataProviderUtil {
    private static final Logger logger = Logger
            .getLogger(JsonDataProviderUtil.class);

    private LinkedHashMap<String, Class<?>> entityClazzMap;
    private String filename;
    private Class<?> clazz;
    private EasyFilter filter;

    public JsonDataProviderUtil setFilter(EasyFilter filter) {
        this.filter = filter;
        return this;
    }

    public JsonDataProviderUtil setEntityClazzMap(
            LinkedHashMap<String, Class<?>> entityClazzMap) {
        this.entityClazzMap = entityClazzMap;
        return this;
    }

    public JsonDataProviderUtil setFileName(String filename) {
        this.filename = filename;
        return this;
    }

    public JsonDataProviderUtil setClazz(Class<?> clazz) {
        this.clazz = clazz;
        return this;
    }

    public Iterator<Object[]> getData() throws IOException {

        String file = null;
        if (clazz != null) {
            file = clazz.getResource(filename).getFile();
        } else {
            file = filename;
        }

        String data = FileUtil.readFromFile(file);

        Gson gson = new Gson();
        LinkedList<Object[]> datas = new LinkedList<Object[]>();
        JsonElement[] rows = gson.fromJson(data, JsonElement[].class);
```

Read-Only | Smart Insert | 2 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class | HttpUtil.class | JsonDataProviderUtil.class

```java
        return this;
    }

    public Iterator<Object[]> getData() throws IOException {

        String file = null;
        if (clazz != null) {
            file = clazz.getResource(filename).getFile();
        } else {
            file = filename;
        }

        String data = FileUtil.readFromFile(file);

        Gson gson = new Gson();
        LinkedList<Object[]> datas = new LinkedList<Object[]>();
        JsonElement[] rows = gson.fromJson(data, JsonElement[].class);

        for (int i = 0; i < rows.length; i++) {

            JsonObject row = rows[i].getAsJsonObject();
            Iterator<Entry<String, JsonElement>> iterator = row.entrySet()
                    .iterator();
            LinkedList<Object> objects = new LinkedList<Object>();// store all
                                                                  // of the
                                                                  // data for
                                                                  // this row
                                                                  // as
                                                                  // objects
            Map<String, Object> rowDataMap = new HashMap<String, Object>();
            while (iterator.hasNext()) {
                Entry<String, JsonElement> entry = iterator.next();

                String key = entry.getKey();
                Class<?> c = entityClazzMap.get(key);
                if (c != null) {
                    Object jo = gson.fromJson(entry.getValue(), c);
                    objects.add(jo);
                } else
                    logger.warn(key
                            + ": does not exists in class map, please define it");
                if (!entry.getValue().isJsonObject()) {
                    rowDataMap.put(key, entry.getValue().toString());
                } else {
                    try {
                        Iterator<Entry<String, JsonElement>> it = entry
                                .getValue().getAsJsonObject().entrySet()
                                .iterator();
                        while (it.hasNext()) {
                            Entry<String, JsonElement> e = it.next();
                            rowDataMap.put(key + ":" + e.getKey(), e.getValue()
                                    .toString());
                        }
                    } catch (Exception ex) {
                        ex.printStackTrace();
                    }
                }
            }
            if (filter == null || filter.match(rowDataMap)) {
                datas.add(objects.toArray());
            }
        }
        return datas.iterator();
    }

}
```

Read-Only | Smart Insert | 2 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class | ProtobufMessageBodyReaderWriter.class | ReflectUtil.class | RESTAPIUtil.class

```java
package com.ebay.maui.util;

import java.io.ByteArrayInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLDecoder;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.util.Iterator;
import java.util.Map.Entry;
import java.util.Properties;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.X509TrustManager;

import org.json.JSONException;
import org.json.JSONObject;

import sun.misc.BASE64Encoder;

import com.ebay.maui.controller.ContextManager;
import com.ebay.maui.controller.Logging;
import com.ebay.maui.helper.CALHelper;
import com.ebay.maui.helper.FileHelper;
import com.ebay.maui.helper.URLHelper;

/**
 * This class is a utility help us do the Restful API testing. Using pure http
 * request to fullfill the request.
 *
 * @author junjshi
 *
 */
public class RESTAPIUtil {
    public static final String GET = "GET";
    public static final String PUT = "PUT";
    public static final String DELETE = "DELETE";
    public static final String POST = "POST";
    public static final boolean quiet = true;

    private static String generateLogMessage(String endPoint,
            Properties reqHeaders, String requestXML, Properties respHeaders,
            String responseXML) {
        if (ContextManager.getGlobalContext() == null
                || ContextManager.getGlobalContext().getTestNGContext() == null)
            return "";
        StringBuffer sbMessage = new StringBuffer("");

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

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

        if (reqHeaders != null) {
```

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class | ProtobufMessageBodyReaderWriter.class | ReflectUtil.class | RESTAPIUtil.class

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

        if (requestXML != null) {
            String postfix = ".json";
            if (reqHeaders != null && reqHeaders.size() > 0) {
                String dataFormat = reqHeaders
                        .getProperty("X-EBAY-SOA-RESPONSE-DATA-FORMAT");
                if (dataFormat == null) {
                    dataFormat = "JSON";
                }
                if (dataFormat.contains("XML")) {
                    postfix = ".xml";
                }
            }
            try {
                FileHelper.writeToFile(outputDir + "/xmls/" + inputfilename
                        + postfix, requestXML);
            } catch (IOException e) {
                e.printStackTrace();
            }
            sbMessage.append(" | <a href='" + subFolder + "/xmls/"
                    + inputfilename + postfix + "' target=input>input</a>");
        }

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

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class | ProtobufMessageBodyReaderWriter.class | ReflectUtil.class | RESTAPIUtil.class

```java
        if (responseXML != null) {
            String postfix = ".json";
            if (respHeaders != null && respHeaders.size() > 0) {
                String dataFormat = respHeaders
                        .getProperty("X-EBAY-SOA-RESPONSE-DATA-FORMAT");
                if (dataFormat == null) {
                    dataFormat = "JSON";
                    postfix = ".json";
                    try {
                        responseXML = new JSONObject(responseXML).toString(2);
                    } catch (JSONException e) {
                    }
                }
                if (dataFormat.contains("XML")) {
                    postfix = ".xml";
                }
            }

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

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

        if (respHeaders != null) {
            String rlogId = respHeaders
                    .getProperty("X-EBAY-API-CAL-TRANACTION-ID");
            if(rlogId==null){//support cal log for Teresa's requirement
                rlogId = respHeaders.getProperty("RlogId");
            }
            if (rlogId != null) {
                try {
                    sbMessage.append(" | <a href='"
//                            + http://cal.vip.qa.ebay.com/cgi/logview?p_cmd=show&p_env="
//                            + environment
//                            + "sql&p_logid="
//                            + rlogId.trim()
                                + CALHelper.getCalURL(URLDecoder.decode(rlogId,"UTF-8"))
                                + "' target=cal>RlogId</a>");
                } catch (UnsupportedEncodingException e) {
                    //ignore
                }
            }
        }

        Logging.logAPICall("", sbMessage.toString(), false);

        return sbMessage.toString();

    }

    /**
     * Use to send the request without header Sample: String output =
     * RESTAPIUtil.request(RESTAPIUtil.GET, url, null);
     * RESTAPIUtil.request(RESTAPIUtil.POST, url, input);
     *
     * @param method
     * @param url
     * @param bodyString
     * @return
     * @throws MalformedURLException
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class | ProtobufMessageBodyReaderWriter.class | ReflectUtil.class | RESTAPIUtil.class

```java
     * @throws IOException
     */
    public static String request(String method, String url, String bodyString)
            throws MalformedURLException, IOException {
        ByteArrayInputStream body = null;
        if (bodyString != null)
            body = new ByteArrayInputStream(bodyString.getBytes());
        // handle special character in the url like
        // "http://www.wmss.stratus.qa.ebay.com/wmssvc/EBAY/users/1192676159?paymentInstrumentRefId=7409694308&creditCardType=MASTER&paymentInstrumentType=CreditCard&client=EBP&lastFour=0424&expirationDate=21 Mar 2013 
        url = URLHelper.encode(url);
        return request(method, new URL(url), body);
    }

    public static RESTResponse getResponse(String method, String url,
            String bodyString) throws MalformedURLException, IOException {
        ByteArrayInputStream body = null;
        if (bodyString != null)
            body = new ByteArrayInputStream(bodyString.getBytes());
        // handle special character in the url like
        // "http://www.wmss.stratus.qa.ebay.com/wmssvc/EBAY/users/1192676159?paymentInstrumentRefId=7409694308&creditCardType=MASTER&paymentInstrumentType=CreditCard&client=EBP&lastFour=0424&expirationDate=21 Mar 2013 
        url = URLHelper.encode(url);
        return getResponse(method, new URL(url), body);
    }

    /**
     * Use to send the request with header Sample:
     * RESTAPIUtil.request(RESTAPIUtil.POST, url,props, input);
     *
     * @param method
     * @param url
     * @param props
     * @param bodyString
     * @return
     * @throws MalformedURLException
     * @throws IOException
     */
    public static String request(String method, String url, Properties props,
            String bodyString) throws MalformedURLException, IOException {
        ByteArrayInputStream body = null;
        if (bodyString != null)
            body = new ByteArrayInputStream(bodyString.getBytes());
        // handle special character in the url like
        // "http://www.wmss.stratus.qa.ebay.com/wmssvc/EBAY/users/1192676159?paymentInstrumentRefId=7409694308&creditCardType=MASTER&paymentInstrumentType=CreditCard&client=EBP&lastFour=0424&expirationDate=21 Mar 2013 
        url = URLHelper.encode(url);
        return request(method, new URL(url), props, body);
    }

    public static RESTResponse getResponse(String method, String url,
            Properties props, String bodyString) throws MalformedURLException,
            IOException {
        ByteArrayInputStream body = null;
        if (bodyString != null)
            body = new ByteArrayInputStream(bodyString.getBytes());
        // handle special character in the url like
        // "http://www.wmss.stratus.qa.ebay.com/wmssvc/EBAY/users/1192676159?paymentInstrumentRefId=7409694308&creditCardType=MASTER&paymentInstrumentType=CreditCard&client=EBP&lastFour=0424&expirationDate=21 Mar 2013 
        url = URLHelper.encode(url);
        return getResponse(method, new URL(url), props, body);
    }

    /**
     * Use to send the request with header, with InputStream
     *
     * @param method
     * @param url
     * @param body
     * @return
     * @throws IOException
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class | ProtobufMessageBodyReaderWriter.class | ReflectUtil.class | RESTAPIUtil.class

```java
    public static String request(String method, URL url, InputStream body)
            throws IOException {
        return request(method, url, null, null, body);
    }

    public static RESTResponse getResponse(String method, URL url,
            InputStream body) throws IOException {
        return getResponse(method, url, null, null, body);
    }

    private static void handleHTTPS(URL url) {
        SSLContext sc = null;
        // Handle HTTPS
        if (url.getProtocol().equalsIgnoreCase("https")) {
            // Accept all certs
            X509TrustManager[] trustAllCerts = new X509TrustManager[] { new X509TrustManager() {
                public void checkClientTrusted(
                        java.security.cert.X509Certificate[] certs,
                        String authType) {
                }

                public void checkServerTrusted(
                        java.security.cert.X509Certificate[] certs,
                        String authType) {
                }

                public java.security.cert.X509Certificate[] getAcceptedIssuers() {
                    return null;
                }

            } };

            try {
                sc = SSLContext.getInstance("SSL");
                sc.init(null, trustAllCerts, new java.security.SecureRandom());
                HttpsURLConnection.setDefaultSSLSocketFactory(sc
                        .getSocketFactory());
            } catch (NoSuchAlgorithmException e) {

                throw new RuntimeException(e);
            } catch (KeyManagementException e2) {
                throw new RuntimeException(e2);
            }

        }
        // Add below blocks to pass certificate verification
        // Create all-trusting host name verifier
        HostnameVerifier allHostsValid = new HostnameVerifier() {
            public boolean verify(String hostname, SSLSession session) {
                return true;
            }
        };
        // Install the all-trusting host verifier
        HttpsURLConnection.setDefaultHostnameVerifier(allHostsValid);

    }

    private static HttpURLConnection prepareConnection(URL url, String method,
            Properties headers) throws IOException {
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod(method);

        if (headers != null) {
            Iterator<Object> it = headers.keySet().iterator();
            while (it.hasNext()) {
                String key = (String) it.next();
                connection.setRequestProperty(key, headers.getProperty(key));
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class | ProtobufMessageBodyReaderWriter.class | ReflectUtil.class | RESTAPIUtil.class

```java
        return connection;
    }

    private static String prepareRequest(HttpURLConnection connection,
            InputStream body) throws IOException {

        // write body if we're doing POST or PUT
        byte buffer[] = new byte[8192];
        int read = 0;
        String requestBody = null;
        if (body != null) {
            connection.setDoOutput(true);

            OutputStream output = connection.getOutputStream();
            while ((read = body.read(buffer)) != -1) {
                output.write(buffer, 0, read);
            }
            requestBody = output.toString();
        }
        return requestBody;
    }

    /**
     * Use to send the request with header, with InputStream and header
     *
     * @param method
     *            - RESTAPIUtil.POST,RESTAPIUtil.GET,RESTAPIUtil.DELETE,
     *              RESTAPIUtil.PUT
     * @param url
     * @param body
     * @return
     * @throws IOException
     */
    public static String request(String method, URL url, Properties headers,
            InputStream body) throws IOException {
        RESTResponse response = getResponse(method, url, headers, body);

        // It's only for backward compatability
//      if (response.getStatusCode() != 200) {
            // wrapper the exception with response for validation purpose.
//          throw new APIException(response.getStatusCode(),
//                  response.getBody(), null);
//      }
        return response.getBody();
    }

    private static void log(String content) {
        Logging.log(content);
    }

    /**
     * Use to send request with authentication information
     *
     * @param method
     * @param url
     * @param username
     * @param password
     * @param body
     * @return
     * @throws IOException
     */
    public static String request(String method, URL url, String username,
            String password, InputStream body) throws IOException {
        Properties prop = new Properties();
        if (username != null) {
            // write auth header
            BASE64Encoder encoder = new BASE64Encoder();
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class | ProtobufMessageBodyReaderWriter.class | ReflectUtil.class | RESTAPIUtil.class

```java
            String encodedCredential = encoder
                    .encode((username + ":" + password).getBytes());
            prop.put("Authorization", "BASIC " + encodedCredential);
        }
        return request(method, url, prop, body);
    }

    public static RESTResponse getResponse(String method, URL url,
            String username, String password, InputStream body)
            throws IOException {
        Properties prop = new Properties();
        if (username != null) {
            // write auth header
            BASE64Encoder encoder = new BASE64Encoder();
            String encodedCredential = encoder
                    .encode((username + ":" + password).getBytes());
            prop.put("Authorization", "BASIC " + encodedCredential);
        }
        return getResponse(method, url, prop, body);
    }

    private static RESTResponse doRequest(HttpURLConnection connection)
            throws IOException {

        RESTResponse response = new RESTResponse();
        long time = System.currentTimeMillis();
        InputStream responseBodyStream = null;

        try {
            connection.connect();

            try {
                responseBodyStream = connection.getInputStream();
                String responseBody = readStream(responseBodyStream);
                response.setBody(responseBody);
                response.setStatusCode(200);
            } catch (IOException ex) {
                log("Response Code:" + connection.getResponseCode());
                String responseBody = readStream(connection.getErrorStream());
                log("Response Body:" + response);
                response.setBody(responseBody);
                response.setStatusCode(connection.getResponseCode());

            }

            // look at headers
            // the 0th header has a null key, and the value is the response line
            // ("HTTP/1.1 200 OK" or whatever)
            Properties respHeaders = new Properties();

            String header = null;
            String headerValue = null;
            int index = 0;
            while ((headerValue = connection.getHeaderField(index)) != null) {
                header = connection.getHeaderFieldKey(index);

                if (header != null)
                    respHeaders.put(header, headerValue);

                index++;
            }
            System.out.println("");

            response.setHeaders(respHeaders);
        } finally {
            if(responseBodyStream!=null) responseBodyStream.close();
            connection.disconnect();
```

Read-Only | Smart Insert | 1 : 1

FileTransferAPIHelper.class | FileUtil.class | GrepUtil.class | ProtobufMessageBodyReaderWriter.class | ReflectUtil.class | RESTAPIUtil.class

```java
        }

        time = System.currentTimeMillis() - time;
        // start printing output
        if (!quiet)
            System.out.println("[read " + response.getBody().length()
                    + " chars in " + time + "ms]");

        return response;
    }

    public static RESTResponse getResponse(String method, URL url,
            Properties headers, InputStream body) throws IOException {
        if (!quiet) {
            System.out.println("[issuing request: " + method + " " + url + "]");
        }
        handleHTTPS(url);

        HttpURLConnection connection = prepareConnection(url, method, headers);

        String requestBody = prepareRequest(connection, body);

        // do request

        RESTResponse response = doRequest(connection);

        // dump body
        // System.out.print(responseBody);
        System.out.flush();
        generateLogMessage(url.toString(), headers, requestBody,
                response.getHeaders(), response.getBody());
        return response;
    }

    private static String readStream(InputStream stream) throws IOException {
        byte buffer[] = new byte[8192];
        StringBuffer responseBody = new StringBuffer();
        int read = 0;
        while ((read = stream.read(buffer)) != -1) {
            responseBody.append(new String(buffer, 0, read));
        }
        return responseBody.toString();
    }

    protected static void usage() {
        System.out.println("Usage:");
        System.out.println("  For get and delete:");
        System.out
                .println("    java -jar restclient.jar [-quiet] user pass GET/DELETE url");
        System.out.println("");
        System.out
                .println("  For put and post, sending the contents of a file as the body:");
        System.out
                .println("    java -jar restclient.jar [-quiet] user pass PUT/POST url filename");
        System.out.println("");
        System.out
                .println("  -quiet will cause only the response body to be printed, otherwise");
        System.out
                .println("  the response headers and timing information is included.");
        System.exit(1);
    }

    private RESTAPIUtil() {
        super();
    }
}
```

Read-Only | Smart Insert | 1 : 1

RESTClient.class

```java
package com.ebay.maui.util;

import javax.ws.rs.core.MediaType;

import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.WebResource.Builder;
import com.sun.jersey.api.client.config.ClientConfig;
import com.sun.jersey.api.client.config.DefaultClientConfig;

/**
 * This class is a utility can help us do restful API testing.
 * It leverage Jersey client to do the request. Support protobuf and protostuff.
 * @author junjshi
 *
 */
public class RESTClient {
    private WebResource resource;

    private String url;

    private MediaType type = null;//MediaType.APPLICATION_XML_TYPE;

    /**
     * Constructor, must create with the url
     * @param url - the url used in the request
     */
    public RESTClient(String url) {
        this.url = url;
        ClientConfig cc = new DefaultClientConfig();
        cc.getClasses().add(ProtobufMessageBodyReaderWriter.class);
        cc.getClasses().add(ProtostuffMessageBodyReaderWriter.class);
        Client client = Client.create(cc);
        this.resource = client.resource(url);
    }



    public void setWebResource(WebResource resource) {
        this.resource = resource;
    }



    public void delete() {
        getWebResource().delete();
    }

    public <T> T delete(Class<T> clz) {
        T response = (T) getWebResource().accept(type).delete(clz);
        return response;
    }

    public <T> T get(Class<T> clz) {
        T response = null;
        if(type!=null)
            response = (T) getWebResource().accept(type).get(clz);
        else
            response = (T) this.resource.get(clz);
        return response;
    }

    public MediaType getType() {
        return type;
    }

    public String getUrl() {
```

RESTClient.class

```java
    public void delete() {
        getWebResource().delete();
    }

    public <T> T delete(Class<T> clz) {
        T response = (T) getWebResource().accept(type).delete(clz);
        return response;
    }

    public <T> T get(Class<T> clz) {
        T response = null;
        if(type!=null)
            response = (T) getWebResource().accept(type).get(clz);
        else
            response = (T) this.resource.get(clz);
        return response;
    }

    public MediaType getType() {
        return type;
    }

    public String getUrl() {
        return url;
    }

    public WebResource getWebResource() {
        return this.resource;
    }

    /**
     * Use to post request
     * Sample:
     * String output = client.type("application/json").post(String.class, input);
     * AddressBookProtos.Person p2 = client.type("application/x-protobuf").post(AddressBookProtos.Person.class, p);
     * PersonProtostuff p2 = client.type("application/x-protostuff").post(PersonProtostuff.class, p);
     * @param <T>
     * @param request
     * @param responseClz
     * @return
     */
    public <T> T post(Object request, Class<T> responseClz) {

        T response = getWebResource().type(type).post(responseClz, request);
        return response;
    }

    public <T> T put(Class<T> clz) {
        T response = (T) getWebResource().accept(type).put(clz);
        return response;
    }

    public void setType(MediaType type) {
        this.type = type;
    }

    public RESTClient setUrl(String url) {
        this.url = url;
        return this;
    }

    public Builder type(String type) {
        return this.resource.type(type);
    }

}
```

RESTResponse.class

```java
* @author jumjshi, created on Jan 19, 2014
package com.ebay.maui.util;

import java.util.Properties;


public class RESTResponse {
    private int statusCode;
    private String body;
    private Properties headers;
    private long time;

    public Properties getHeaders() {
        return headers;
    }



    public long getTime() {
        return time;
    }



    public void setTime(long time) {
        this.time = time;
    }



    public void setHeaders(Properties headers) {
        this.headers = headers;
    }



    public int getStatusCode() {
        return statusCode;
    }



    public void setStatusCode(int statusCode) {
        this.statusCode = statusCode;
    }



    public String getBody() {
        return this.body;
    }



    public void setBody(String body) {
        this.body = body;
    }

}
```

Read-Only | Smart Insert | 2 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class

```java
package com.ebay.maui.util;

import java.io.FileInputStream;
import java.io.InputStream;
import java.lang.reflect.Array;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.lang.reflect.ParameterizedType;
import java.lang.reflect.Type;
import java.math.BigDecimal;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;
import java.util.TreeSet;

import jxl.Sheet;
import jxl.Workbook;

import org.apache.log4j.Logger;

import com.ebay.maui.controller.ContextManager;
import com.ebay.maui.controller.EasyFilter;
import com.ebay.maui.exception.MauiException;
import com.ebay.maui.util.internal.entity.TestObject;

public class SpreadSheetUtil {

    private static Logger logger = Logger.getLogger(SpreadSheetUtil.class);
    /** Primitive type name -> class map. */

    private static final Map<Class<?>, Class<?>> PRIMITIVE_TYPE_MAP = new HashMap<Class<?>, Class<?>>();

    /** Setup the primitives map. */
    static {
        PRIMITIVE_TYPE_MAP.put(Boolean.TYPE, Boolean.class);
        PRIMITIVE_TYPE_MAP.put(Byte.TYPE, Byte.class);
        PRIMITIVE_TYPE_MAP.put(Character.TYPE, Character.class);
        PRIMITIVE_TYPE_MAP.put(Short.TYPE, Short.class);
        PRIMITIVE_TYPE_MAP.put(Integer.TYPE, Integer.class);
        PRIMITIVE_TYPE_MAP.put(Long.TYPE, Long.class);
        PRIMITIVE_TYPE_MAP.put(Float.TYPE, Float.class);
        PRIMITIVE_TYPE_MAP.put(Double.TYPE, Double.class);
    }

    @SuppressWarnings({ "rawtypes", "unchecked" })
    private static Object _readFieldValueObject(Class<?> fieldClz, Type type,
            Map<String, Object> dataMap, String combinedFieldName)
            throws Exception {
        Object fieldValue = null;

        if (fieldClz.isArray()) {// Take care of Arrays
            int size = getArraySize(dataMap, combinedFieldName);
            if (size > 0) {
                fieldValue = Array.newInstance(fieldClz.getComponentType(),
                        size);
                for (int j = 0; j < size; j++) {
                    if(combinedFieldName==null)
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class

```java
                    if(combinedFieldName==null)
                        Array.set(
                                fieldValue,
                                ),
                                readFieldValue(fieldClz.getComponentType(),
                                        "*+j, dataMap));
                    else
                        Array.set(
                                fieldValue,
                                ),
                                readFieldValue(fieldClz.getComponentType(),
                                        combinedFieldName + "." + j, dataMap));
                }
            }
        } else if (java.util.List.class.isAssignableFrom(fieldClz)) {// Take
                                                                     // care
                                                                     // of
                                                                     // Collections
            java.util.ArrayList list = java.util.ArrayList.class.newInstance();
            int size = getArraySize(dataMap, combinedFieldName);
            if (size > 0) {
                fieldValue = list;
                Class<?> itemClz = getListItemType(type);
                for (int j = 0; j < size; j++) {
                    list.add(readFieldValue(itemClz, combinedFieldName + "."
                            + j, dataMap));
                }
            }
        } else if (fieldClz.isAssignableFrom(java.util.Set.class)) {// Take care of Set
            java.util.Set list = java.util.LinkedHashSet.class.newInstance();
            int size = getArraySize(dataMap, combinedFieldName);
            if (size > 0) {
            fieldValue = list;
            Class<?> itemClz = getListItemType(type);
            for (int j = 0; j < size; j++) {
            list.add(readFieldValue(itemClz, combinedFieldName + "."
            + j, dataMap));
            }
            }
        }else {
            fieldValue = readFieldValue(fieldClz, combinedFieldName, dataMap);
        }

        return fieldValue;
    }

    protected static void formatDPTags(Map<String, Object> rowDataMap) {
        if (rowDataMap.get(TestObject.TEST_DP_TAGS) != null) {
            String dpTags = rowDataMap.get(TestObject.TEST_DP_TAGS).toString();
            if (dpTags.trim().length() > 0) {
                String[] dpTagArray = dpTags.split(",");
                String tempDPTags = "";
                for (int idx = 0; dpTagArray.length > 0
                        && idx < dpTagArray.length; idx++) {
                    tempDPTags = tempDPTags.concat("[" + dpTagArray[idx].trim()
                            + "]");
                    if (idx != dpTagArray.length - 1) {
                        tempDPTags = tempDPTags.concat(",");
                    }
                }
                rowDataMap.put(TestObject.TEST_DP_TAGS, tempDPTags);
            }
        }
    }

    public static int getArraySize(Map<String, Object> map, String key) {
        if(key==null)
            return map.size();
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class

```java
        if(key==null)
            return map.size();
        int count = 0;
        boolean valueFound = false;
        key = key.toLowerCase();
        for (Entry<String, Object> entry : map.entrySet()) {
            String key2 = entry.getKey();
            String value2 = (String) entry.getValue();
            if (value2 == null || value2.length() == 0)
                continue;

            key2 = key2.toLowerCase();
            if (key2.startsWith(key + ".")) {
                valueFound = true;
                String subst = key2.substring((key + ".").length());
                String[] ss = subst.split("\\.");
                try {
                    int value = Integer.parseInt(ss[0]);
                    count = (value > count ? value : count);
                } catch (Exception e) {
                }
            }
        }

        return (valueFound ? count + 1 : count);
    }

    /**
     * add for backward compatability
     * @param clazz
     * @param filename
     * @param sheetNumber
     * @param columnNames
     * @param filter
     * @return
     */
    public static synchronized Iterator<Object[]> getDataFromSpreadsheet(Class<?> clazz, String filename, int sheetNumber,
            String[] columnNames, EasyFilter filter){
        return SpreadSheetUtil.getDataFromSpreadsheet(clazz, filename, null, sheetNumber, columnNames, filter, false);
    }

    /**
     * Reads data from spreadsheet. If sheetName and sheetNumber both are
     * supplied the sheetName takes precedence. Put the excel sheet in the same
     * folder as the test case and specify clazz as <code>this.getClass()</code>
     * .
     */
    public static synchronized Iterator<Object[]> getDataFromSpreadsheet(
            Class<?> clazz, String filename, String sheetName, int sheetNumber,
            String[] fields, EasyFilter filter, boolean readHeaders) {
        return getDataFromSpreadsheet(clazz, filename, sheetName, sheetNumber,
                fields, filter, readHeaders, true);
    }

    /**
     * Reads data from spreadsheet. If sheetName and sheetNumber both are
     * supplied the sheetName takes precedence. Put the excel sheet in the same
     * folder as the test case and specify clazz as <code>this.getClass()</code>
     * .
     *
     * @param clazz
     * @param filename
     * @param sheetName
     * @param sheetNumber
     * @param fields
     * @param filter
     * @param readHeaders
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class

```java
     * @return
     * @throws Exception
     */

    public static synchronized Iterator<Object[]> getDataFromSpreadsheet(
            Class<?> clazz, String filename, String sheetName, int sheetNumber,
            String[] fields, EasyFilter filter, boolean readHeaders,
            boolean supportDPFilter) {

        System.gc(); // KEEPME

        // Let CSVUtil handle CSV Files
        if (filename.toLowerCase().endsWith(".csv")) {
            return CSVUtil.getDataFromCSVFile(clazz, filename, fields, filter,
                    readHeaders, supportDPFilter);
        }

        Workbook w = null;
        InputStream is = null;
        try {
            if (clazz != null) {
                is = clazz.getResourceAsStream(filename);
            } else {
                is = new FileInputStream(filename);
            }

            if (is == null) {
                return new ArrayList<Object[]>().iterator();
            }

            w = Workbook.getWorkbook(is);

            if (w.getSheetNames().length <= sheetNumber) {
                throw new MauiException("Sheet # " + sheetNumber + " for "
                        + filename + " not found.");
            }

            if (sheetName != null) {
                for (int i = 0; i < w.getSheetNames().length; i++) {
                    if (sheetName.equals(w.getSheetNames()[i])) {
                        sheetNumber = i;
                    }
                }
            }

            // Get the sheet
            Sheet sheet = w.getSheet(sheetNumber);

            // ignore blank columns
            int columnCount = sheet.getColumns();
            for (int j = 0; j < sheet.getColumns(); j++) {
                String content = sheet.getCell(j, 0).getContents();
                if (content == null || content.trim().length() == 0) {
                    //columnCount = j + 1;
                    columnCount = j;//Change by Jojo, for the 7th column, it's "", so the column count should be 7
                    break;
                }
            }

            List<Object[]> sheetData = new ArrayList<Object[]>();
            if (readHeaders) {
                List<Object> rowData = new ArrayList<Object>();
                if (fields == null) {
                    for (int j = 0; j < columnCount; j++) {
                        rowData.add(sheet.getCell(j, 0).getContents());
                    }
                } else {
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class

```java
                    for (int i = 0; i < fields.length; i++) {
                        rowData.add(fields[i]);
                    }
                }
                sheetData.add(rowData.toArray(new Object[rowData.size()]));
            }

            int testTitleColumnIndex = -1;
            int testSiteColumnIndex = -1;
            // Search for Title & Site column
            for (int i = 0; i < columnCount; i++) {
                if (testTitleColumnIndex == -1
                        && TestObject.TEST_TITLE.equalsIgnoreCase(sheet
                                .getCell(i, 0).getContents())) {
                    testTitleColumnIndex = i;
                } else if (testSiteColumnIndex == -1
                        && TestObject.TEST_SITE.equalsIgnoreCase(sheet.getCell(
                                i, 0).getContents())) {
                    testSiteColumnIndex = i;
                }

                if (testTitleColumnIndex != -1 && testSiteColumnIndex != -1) {
                    break;
                }
            }

            // Let's check for blank rows first
            // The first row is the header
            StringBuffer sbBlank = new StringBuffer();
            for (int i = 1; i < sheet.getRows(); i++) {
                if (testTitleColumnIndex != -1
                        && testSiteColumnIndex != -1
                        && ((sheet.getCell(testTitleColumnIndex, i)
                                .getContents() == null || sheet
                                .getCell(testTitleColumnIndex, i).getContents()
                                .trim().length() == 0) || (sheet.getCell(
                                testSiteColumnIndex, i).getContents() == null || sheet
                                .getCell(testSiteColumnIndex, i).getContents()
                                .trim().length() == 0))) {
                    sbBlank.append(i + 1).append(',');
                }
            }
            if (sbBlank.length() > 0) {
                sbBlank.deleteCharAt(sbBlank.length() - 1);
                throw new MauiException(
                        "Blank TestTitle and/or Site value(s) found on Row(s) "
                                + sbBlank.toString() + ".");
            }

            Set<String> uniqueDataSet = new TreeSet<String>();

            // Jerry: Add DataProviderTags filter
            /**
             * Modified by Gary to support include tags and exclude tags
             */
            if (supportDPFilter) {
                EasyFilter dpFilter = getDPFilter();

                if (dpFilter != null) {
                    if (filter == null) {
                        filter = dpFilter;
                    } else {
                        filter = EasyFilter.and(filter, dpFilter);
                    }
                }
            }
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class

```java
            }
            // End
            // The first row is the header
            for (int i = 1; i < sheet.getRows(); i++) {
                // Check for duplicate Title & Site
                if (testTitleColumnIndex != -1 && testSiteColumnIndex != -1) {
                    String uniqueString = sheet
                            .getCell(testTitleColumnIndex, i).getContents()
                            + "$$$$####$$$$"
                            + sheet.getCell(testSiteColumnIndex, i)
                                    .getContents();
                    if (uniqueDataSet.contains(uniqueString))
                        throw new MauiException(
                                "Duplicate TestTitle and Site combination found in the spreadsheet "
                                        + "with TestTitle = {"
                                        + sheet.getCell(testTitleColumnIndex, i)
                                                .getContents()
                                        + "} "
                                        + "and Site = {"
                                        + sheet.getCell(testSiteColumnIndex, i)
                                                .getContents() + "}");

                    uniqueDataSet.add(uniqueString);
                }

                Map<String, Object> rowDataMap = new HashMap<String, Object>();
                List<Object> rowData = new ArrayList<Object>();

                // Create the mapping between headers and column data
                for (int j = 0; j < columnCount; j++) {
                    rowDataMap.put(sheet.getCell(j, 0).getContents(), sheet
                            .getCell(j, i).getContents());
                }

                if (fields == null) {
                    for (int j = 0; j < columnCount; j++) {
                        rowData.add(sheet.getCell(j, i).getContents());
                    }
                } else {
                    for (int k = 0; k < fields.length; k++) {
                        rowData.add(getValue(rowDataMap, fields[k]));
                    }
                }

                // Jerry: Add DataProviderTags filter
                /**
                 * Modified by Gary to support include tags and exclude tags
                 */
                if (supportDPFilter) {
                    SpreadSheetUtil.formatDPTags(rowDataMap);
                }
                // End
                if (Filter == null || filter.match(rowDataMap)) {
                    sheetData.add(rowData.toArray(new Object[rowData.size()]));
                }
            }

            sheet = null;

            if ((!readHeaders && sheetData.isEmpty())
                    || (readHeaders && sheetData.size() <= 1))
                logger.warn("No matching data found on spreadsheet: "
                        + filename + " with filter criteria: "
                        + filter.toString());

            return sheetData.iterator();
        } catch (Throwable e) {
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class

```java
        } catch (Throwable e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        } finally {

            if (w != null) {
                w.close();
                w = null;
            }
            if (is != null) {
                try {
                    is.close();
                } catch (Exception e) {
                }// KEEPME
            }
        }
    }

    public static List<Object[]> getDataList(List<Object[]> table,
            EasyFilter filter) {

        List<Object[]> sheetData = new ArrayList<Object[]>();

        // The first row is the header
        String[] fields = (String[]) table.get(0);

        for (int i = 1; i < table.size(); i++) {
            Map<String, Object> rowDataMap = new HashMap<String, Object>();
            Object[] rowData = table.get(i);
            // Create the mapping between headers and column data
            for (int j = 0; j < rowData.length; j++) {
                rowDataMap.put(fields[j], rowData[j]);
            }
            if (filter == null || filter.match(rowDataMap)) {
                sheetData.add(rowData);
            }
        }

        sheetData.add(0, fields);

        return sheetData;
    }

    protected static EasyFilter getDPFilter() {
        String includedTags = ContextManager.getGlobalContext()
                .getDPTagsInclude();
        String excludedTags = ContextManager.getGlobalContext()
                .getDPTagsExclude();

        EasyFilter dpFilter = null;
        if (includedTags != null && includedTags.trim().length() > 0) {
            String[] includeTagsArray = includedTags.split(",");
            for (int idx = 0; includeTagsArray.length > 0
                    && idx < includeTagsArray.length; idx++) {
                if (dpFilter == null) {
                    dpFilter = EasyFilter.containsIgnoreCase(
                            TestObject.TEST_DP_TAGS,
                            "[" + includeTagsArray[0].trim() + "]");
                } else {
                    dpFilter = EasyFilter.or(dpFilter, EasyFilter
                            .containsIgnoreCase(TestObject.TEST_DP_TAGS, "["
                                    + includeTagsArray[idx].trim() + "]"));
                }
            }
        }
        if (excludedTags != null && excludedTags.trim().length() > 0) {
            String[] excludeTagsArray = excludedTags.split(",");
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class

```java
465             for (int idx = 0; excludeTagsArray.length > 0
466                     && idx < excludeTagsArray.length; idx++) {
467                 if (dpFilter == null) {
468                     dpFilter = EasyFilter.not(EasyFilter.containsIgnoreCase(
469                             TestObject.TEST_DP_TAGS, "["
470                                     + excludeTagsArray[idx].trim() + "]"));
471                 } else {
472                     dpFilter = EasyFilter.and(dpFilter, EasyFilter
473                             .not(EasyFilter.containsIgnoreCase(
474                                     TestObject.TEST_DP_TAGS, "["
475                                             + excludeTagsArray[idx].trim()
476                                             + "]")));
477                 }
478             }
479         }
480         return dpFilter;
481     }
482
483     /**
484      * Create Entity Objects based on data in spreadsheet.
485      *
486      * This method is only for Data Provider. Because it also filer the data
487      * based on the dpTagsInclude/dpTagsExclude which is defined in testng
488      * configuration file
489      */
490     public static Iterator<Object[]> getEntitiesFromSpreadsheet(Class<?> clazz,
491             LinkedHashMap<String, Class<?>> entityClazzMap, String filename,
492             int sheetNumber, String[] fields, EasyFilter filter)
493             throws Exception {
494         return SpreadSheetUtil.getEntitiesFromSpreadsheet(clazz,
495                 entityClazzMap, filename, null, sheetNumber, fields, filter);
496     }
497
498     /**
499      * Create Entity Objects based on data in spreadsheet.
500      *
501      * This method is only for Data Provider. Because it also filer the data
502      * based on the dpTagsInclude/dpTagsExclude which is defined in testng
503      * configuration file
504      *
505      * @param clazz
506      * @param entityClazzMap
507      * @param filename
508      * @param sheetName
509      * @param sheerNumber
510      * @param fields
511      * @param filter
512      * @param readHeaders
513      * @param supportDPFilter
514      * @return
515      * @throws Exception
516      */
517     public static Iterator<Object[]> getEntitiesFromSpreadsheet(Class<?> clazz,
518             LinkedHashMap<String, Class<?>> entityClazzMap, String filename,
519             String sheetName, int sheetNumber, String[] fields,
520             EasyFilter filter, boolean readHeaders, boolean supportDPFilter) throws Exception {
521
522         Iterator<Object[]> dataIterator = getDataFromSpreadsheet(clazz,
523                 filename, sheetName, sheetNumber, fields, filter, readHeaders, supportDPFilter);
524
525         List<Object[]> list = getEntityData(dataIterator,entityClazzMap);
526
527         return list.iterator();
528     }
529
530     /**
531      * Create Entity Objects based on data in spreadsheet.
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class

```java
531      * Create Entity Objects based on data in spreadsheet.
532      *
533      * This method is only for Data Provider. Because it also filer the data
534      * based on the dpTagsInclude/dpTagsExclude which is defined in testng
535      * configuration file
536      *
537      * @param clazz
538      * @param entityClazzMap
539      * @param filename
540      * @param sheetName
541      * @param sheerNumber
542      * @param fields
543      * @param filter
544      * @return
545      * @throws Exception
546      */
547     public static Iterator<Object[]> getEntitiesFromSpreadsheet(Class<?> clazz,
548             LinkedHashMap<String, Class<?>> entityClazzMap, String filename,
549             String sheetName, int sheetNumber, String[] fields,
550             EasyFilter filter) throws Exception {
551
552         Iterator<Object[]> dataIterator = getDataFromSpreadsheet(clazz,
553                 filename, sheetName, sheetNumber, fields, filter, true);
554
555         List<Object[]> list = getEntityData(dataIterator,entityClazzMap);
556
557         return list.iterator();
558     }
559     /**
560      * Create Entity Objects based on data in spreadsheet.
561      *
562      * This method is only for Data Provider. Because it also filer the data
563      * based on the dpTagsInclude/dpTagsExclude which is defined in testng
564      * configuration file
565      */
566     public static List<Object[]> getEntitiesListFromSpreadsheet(Class<?> clazz,
567             LinkedHashMap<String, Class<?>> entityClazzMap, String filename,
568             int sheetNumber, String[] fields, EasyFilter filter)
569             throws Exception {
570         return SpreadSheetUtil.getEntitiesListFromSpreadsheet(clazz,
571                 entityClazzMap, filename, null, sheetNumber, fields, filter);
572     }
573
574     /**
575      * Create Entity Objects based on data in spreadsheet.
576      *
577      * This method is only for Data Provider. Because it also filer the data
578      * based on the dpTagsInclude/dpTagsExclude which is defined in testng
579      * configuration file
580      *
581      * @param clazz
582      * @param entityClazzMap
583      * @param filename
584      * @param sheetName
585      * @param sheerNumber
586      * @param fields
587      * @param filter
588      * @return List<Object[]>
589      * @throws Exception
590      */
591     public static List<Object[]> getEntitiesListFromSpreadsheet(Class<?> clazz,
592             LinkedHashMap<String, Class<?>> entityClazzMap, String filename,
593             String sheetName, int sheetNumber, String[] fields,
594             EasyFilter filter) throws Exception {
595
596         Iterator<Object[]> dataIterator = getDataFromSpreadsheet(clazz,
597                 filename, sheetName, sheetNumber, fields, filter, true);
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class

```java
        List<Object[]> list = getEntityData(dataIterator,entityClazzMap);

        return list;
    }

    public static List<Object[]> getEntityData(Iterator<Object[]> dataIterator,LinkedHashMap<String, Class<?>> entityClazzMap) throws Exception
    {
        List<Object[]> list = new ArrayList<Object[]>();
        // Get the headers
        Object[] headerArray = null;
        if (dataIterator.hasNext()) {
            headerArray = dataIterator.next();
        }

        while (dataIterator.hasNext()) {

            Object[] rowDataArray = dataIterator.next();
            Map<String, Object> map = new LinkedHashMap<String, Object>();

            List<Object> rowData = new ArrayList<Object>();
            for (int j = 0; j < headerArray.length; j++) {
                String header = (String) headerArray[j];
                map.put(header, rowDataArray[j]);
            }

            Map<String, Boolean> temp = new HashMap<String, Boolean>();
            if (entityClazzMap != null) {
                for (Entry<String, Class<?>> entry : entityClazzMap.entrySet()) {
                    temp.put(entry.getKey(), Boolean.TRUE);
                    rowData.add(readObject(entry.getValue(), entry.getKey(),
                            map));
                }
            }

            for (int i = rowDataArray.length - 1; i >= 0; i--) {

                int docIdx = ((String) headerArray[i]).indexOf(".");
                if (docIdx < 0 && temp.get((String) headerArray[i])==null) {
                    rowData.add(0, rowDataArray[i]);
//              } else if (temp.get((String) headerArray[i]).substring(0,
//                      docIdx)) == null) {
//                  rowData.add(0, rowDataArray[i]);
                }
            }

            list.add(rowData.toArray(new Object[] { rowData.size() }));
        }
        return list;
    }

    public static Map<String, Object> getFieldsDataNeedToBeSet(
            Map<String, Object> map, String key) {
        Map<String, Object> result = new LinkedHashMap<String, Object>();

        for (String key2 : map.keySet()) {
            if (key2.equalsIgnoreCase(key)) {
                if(map.get(key2) != null)
                    result.put(key2, map.get(key2).toString());
            }
            if (key2.toLowerCase().startsWith(key.toLowerCase() + ".")) {
                if(map.get(key2) != null)
                    result.put(key2.substring(key.length() + 1), map.get(key2)
                            .toString());
            }
        }
        return result;
```

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class

```java
        return result;
    }

    public static Map<String, Object> getFieldsNeedToBeSet(
            Map<String, Object> map, String key) {
        Map<String, Object> result = new LinkedHashMap<String, Object>();
        String lastKey = "";
        for (String key2 : map.keySet()) {
            if (key2.equalsIgnoreCase(key))
                result.put(key2, map.get(key2));
            if (key2.toLowerCase().startsWith(key.toLowerCase() + ".")) {
                String newkey = key2.substring(key.length() + 1);
                if (newkey.contains("."))
                    newkey = newkey.substring(0, newkey.indexOf("."));
                if (!newkey.equalsIgnoreCase(lastKey))
                    result.put(newkey, map.get(key2));
            }
        }
        return result;
    }

    private static Class<?> getListItemType(Type type)
            throws ClassNotFoundException {

        Class<?> itemClz = null;
        if (type instanceof ParameterizedType) {
            ParameterizedType pt = (ParameterizedType) type;
            itemClz = Class.forName(pt.getActualTypeArguments()[0].toString()
                    .substring("class ".length()));
        }
        return itemClz;
    }

    public static Object getValue(Map<String, Object> map, String key) {
        for (Entry<String, Object> entry : map.entrySet()) {
            if ((entry.getKey() == null && key == null)
                    || (entry.getKey() != null && entry.getKey()
                            .equalsIgnoreCase(key)))
                return entry.getValue();
        }
        return null;
    }

    private static boolean isPrimitive(Class<?> clz) {
        if (clz.isPrimitive())
            return true;
        else if (clz.getCanonicalName().equals(
                "java.lang." + clz.getSimpleName()))
            return true;
        else
            return false;
    }

    public static void main(String[] args) throws ParseException
    {
        Date d = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'").parse("2012-07-13T21:05:39.000Z");
        System.out.println(d);
        Date d2 = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSSz").parse("2012-05-11T12:01:12.123PDT");
        System.out.println(d2);
    }

    private static Object readFieldValue(Class<?> fieldClz, String fieldName,
            Map<String, Object> dataMap) throws Exception {
        Object fieldValue = null;
        String tempValue = (String) getValue(dataMap, fieldName);

        // Return null when field is atomic and value is null or blank
```

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class

```java
        String tempValue = (String) getValue(dataMap, fieldName);

        // Return null when field is atomic and value is null or blank
        if ((tempValue == null || tempValue.length() == 0)
                && (fieldClz.isEnum()
                        || fieldClz.getName().equals("java.util.Calendar")
                        || fieldClz.getName().equals("java.math.BigDecimal") || isPrimitive(fieldClz)))
            return null;

        if (fieldClz.isEnum()) {
            try {
                fieldValue = fieldClz.getMethod("valueOf", String.class)
                        .invoke(fieldClz, tempValue);
            } catch (Exception e) {
                logger.warn("Ex", e);
            }
        } else if (fieldClz.getName().equals("java.util.Calendar")) {// Take
                                                                     // care
                                                                     // of
                                                                     // Date
            Calendar calendar = Calendar.getInstance();
            try{
                calendar.setTime(new SimpleDateFormat("MM/dd/yyyy").parse(tempValue));
            }
            catch(ParseException ex)
            {
                try{
                    calendar.setTime(new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'").parse(tempValue));
                }catch(ParseException ex2)
                {
                    calendar.setTime(new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSSz").parse(tempValue));

                }

            }
            fieldValue = calendar;
        } else if (fieldClz.getName().equals("java.math.BigDecimal")) {// Take
                                                                       // care
                                                                       // of
                                                                       // BigDecimal
            fieldValue = new BigDecimal(tempValue);
        } else if (isPrimitive(fieldClz)) {// Take care of primitives
            Constructor<?> constructor;
            try {
                if (fieldClz.getName().equals("java.lang.String")) {
                    fieldValue = tempValue;
                } else {
                    if (PRIMITIVE_TYPE_MAP.containsKey(fieldClz))
                        constructor = PRIMITIVE_TYPE_MAP.get(fieldClz)
                                .getConstructor(String.class);
                    else
                        constructor = fieldClz.getConstructor(String.class);

                    fieldValue = constructor.newInstance(tempValue);
                }
            } catch (Exception e) {
                logger.warn("Ex", e);
            }
        } else {// Non-atomic or Object field
            fieldValue = readObject(fieldClz, fieldName, dataMap);
        }

        return fieldValue;
    }

    public static Object readObject(Class<?> clz, String objectName,
            Map<String, Object> dataMap) throws Exception {
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class

```java
            Map<String, Object> dataMap) throws Exception {
        Object object = null;
        if (clz == null)
            return null;
        if (objectName == null)
            objectName = clz.getSimpleName();
        Map<String, Object> fieldMap = getFieldsNeedToBeSet(dataMap, objectName);
        Map<String, Object> datamap = getFieldsDataNeedToBeSet(dataMap,
                objectName);

        for (String fieldName : fieldMap.keySet()) {
            String first = "" + fieldName.charAt(0);
            String realfieldName = fieldName.replaceFirst(first,
                    first.toLowerCase());
            Object fieldValue = null;
            Class<?> type = null;
            try {
                    Class<?>[] parameterTypes = new Class<?>[] {};
                    Method method = clz
                            .getMethod("get" + fieldName, parameterTypes);
                    type = method.getReturnType();
                    fieldValue = _readFieldValueObject(type,
                            method.getGenericReturnType(), datamap, fieldName);
            } catch (NoSuchMethodException ex) {
                try {
                    Class<?>[] parameterTypes = new Class<?>[] {};
                    Method method = clz.getMethod("is" + fieldName,
                            parameterTypes);
                    type = method.getReturnType();
                    fieldValue = _readFieldValueObject(type,
                            method.getGenericReturnType(), datamap, fieldName);
                } catch (NoSuchMethodException ex2) {
                    try {
                        Field field = clz.getDeclaredField(realfieldName);
                        fieldValue = _readFieldValueObject(field.getType(), field.getGenericType(), datamap, fieldName);
                    }catch (NoSuchFieldException ex3) {
                        //handle array and primitive type
                        if(clz.isArray()){
                            fieldValue = _readFieldValueObject(clz,
                                    clz, datamap, null);
                            return fieldValue;
                        }
                        else if( isPrimitive(clz2)){
                            fieldValue = _readFieldValueObject(clz,
                                    null, datamap, objectName);
                            return fieldValue;
                        }
                        //Can't find field, get**,is*** method, set it to String.class
                        try{
                            fieldValue = _readFieldValueObject(String.class, String.class, datamap, fieldName);
                            type = String.class;
                        }
                        catch(Exception e)
                        {
                            logger.warn("Ex:"+clz.getName(), e);
                        }
                    }
                }
            }

            // execute the Setter Method
            try {
                if(fieldValue != null){

                    if(object == null)
                    {
                        try{
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class

```java
                                clz, datamap, null);
                        return fieldValue;
                    }
                    else if( isPrimitive(clz)){
                        fieldValue = _readFieldValueObject(clz,
                                null, datamap, objectName);
                        return fieldValue;
                    }
                    //Can't find field, get**,is*** method, set it to String.class
                    try{
                        fieldValue = _readFieldValueObject(String.class, String.class, datamap, fieldName);
                        type = String.class;
                    }
                    catch(Exception e)
                    {
                        logger.warn("Ex:"+clz.getName(), e);
                    }
                }
            }
        }

        // execute the Setter Method
        try {
            if(fieldValue != null){

                if(object == null)
                {
                    try{
                        object = clz.newInstance();
                    }catch(InstantiationException e)
                    {
                        //handle no null parameter constructor
                        Class<?>[] parameterTypes = new Class<?>[1];
                        parameterTypes[0] = fieldValue.getClass();
                        Constructor<?> constructor = clz.getDeclaredConstructor(parameterTypes);
                        constructor.setAccessible(true);
                        object = constructor.newInstance(fieldValue);
                        return object;
                    }
                }
            }
            if (fieldValue != null) {

                if (object == null)
                    object = clz.newInstance();
                try {
                    Class<?>[] parameterTypes = new Class<?>[1];
                    parameterTypes[0] = type;
                    Method method = object.getClass().getMethod(
                            "set" + fieldName, parameterTypes);
                    method.invoke(object, fieldValue);
                } catch (Exception ex) {
                    Field field2 = object.getClass().getDeclaredField(
                            realfieldName);
                    field2.setAccessible(true);
                    field2.set(object, fieldValue);
                }
            }
        } catch (Exception e) {
            logger.warn("Ex", e);
        }
    }

    return object;
  }
}
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.cla | SiteUtil.class | SpreadSheetUtil. | StringInputStrea | SSHUtil.class | StringUtil.class | SystemUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
package com.ebay.maui.util;

import java.util.Date;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.commons.lang.StringUtils;
import org.apache.commons.lang.time.DateUtils;

import com.ebay.maui.controller.Logging;

/**
 * StringUtil
 *
 * @author Shanth Kumar(Shanth.Kumar@ebay.com)
 * @date 11/09/11
 */
public final class StringUtil {

    private static final String NUM_PATTERN = "((-|\\+)?[0-9]+(\\.[0-9]+)?)+";

    public static String arrayToString(Long[] arr) {
        String str = null;
        if (arr != null && arr.length > 0) {
            str = "";
            for (int i = 0; arr != null && i < arr.length; i++) {
                str = str + arr[i].toString() + "|";
            }

            str = str.substring(0, str.length() - 1);
        }
        return str;
    }

    /**
     * get comma seperated string
     *
     * @param obj
     *            an array not as wrapper
     * @return comma separated string
     */
    public static String getCSV(Object[] obj) {

        if (obj == null || obj.length == 0)
            return "";

        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < obj.length; i++) {
            sb.append(obj[i] + ",");
        }
        return sb == null ? null : sb.toString().substring(0,
                sb.toString().length() - 1);
    }

    /**
     * Deprecated - use the Integer form of this method instead.
     *
     * Given a string that represents a month (i.e. "1" or "2"), this method will return the 3-letter
     * abbreviation for that month, so "1" would return "JAN", "2" would return "FEB", etc.
     *
     * @param month
     * @return
     */
    @Deprecated
    public static String ~~getMonthInWords~~( String month ) {
        // Parse out the desired month. If it can't be parsed into an int, it should throw an exception
        // Subtract 1 from the int, because the DateUtils are 0-indexed at January
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.cla | SiteUtil.class | SpreadSheetUtil. | StringInputStrea | SSHUtil.class | StringUtil.class | SystemUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
        int monthIndex;
        try {
            monthIndex = Integer.parseInt( month );
        } catch ( NumberFormatException ex ) {
            //TODO Keeping this as a null return so the functionality stays the same as before--but this should really throw an exception
            //Meanwhile we'll at least give a helpful error message
            Logging.log( "The String passed in cannot be parsed as an Integer: " + month );
            return null;
        }

        //Now that we've scrubbed the integer out of the string passed in, do the real work
        return getMonthInWords( monthIndex );
    }

    /**
     * Given an integer, this method will return the 3-letter abbreviation for that month, so 1
     * would return "JAN", 2 would return "FEB", etc.
     *
     * @param month
     * @return
     */
    protected static String getMonthInWords( int monthIndex ) {
        //Set up a new date object to have the month passed in
        Date monthDate;
        int offsetMonthIndex = monthIndex - 1;
        try {

            monthDate = DateUtils.setMonths( new Date(), offsetMonthIndex );
        } catch ( IllegalArgumentException ex ) {
            //TODO This still returns null, but these try/catch blocks should go away and these exceptions should bubble up
            //Meanwhile here's a helpful error message
            Logging.log( "The number passed in is out of bounds for a month. It should be only 1-12, but you passed in [" + offsetMonthIndex + "]" );
            return null;
        }

        //Format the string to return just the month, and only return the first 3 characters. We could also use SimpleDateFormat instead
        //TODO - this doesn't deal with locales, so we should consider setting it to Locale.US somewhere earlier
        String monthStr = String.format( "%tB", monthDate ).substring( 0, 3 );

        //Return it in upper case
        return monthStr.toUpperCase();
    }

    protected static int offsetMonthIndex(int monthIndex) {
        return monthIndex - 1;
    }

    public static boolean isBooleam(String str) {
        return ("false".equalsIgnoreCase(str) || "true".equalsIgnoreCase(str));
    }

    public static boolean isEmpty(String str) {
        return str == null || str.trim().length() == 0;
    }

    public static boolean isNumeric(String input) {

        return input.matches(NUM_PATTERN);

    }

    public static boolean isPositive(String str) {
        try {
            if (str == null || str.trim().length() == 0)
                return false;
            return Double.parseDouble(str) > 0;
        } catch (Exception e) {
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.cla | SiteUtil.class | SpreadSheetUtil. | StringInputStrea | SSHUtil.class | StringUtil.class | SystemUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
        } catch (Exception e) {
            return false;
        }

    }

    /**
     *
     * @param str
     *          inputr string
     * @return trimmed string
     */
    public static String safeTrim(String str) {
        return str == null ? null : str.toString().trim();
    }

    /**
     * Converts a double to string in two decimal format (no rounding--the number is just lopped off)
     * .9811539845 returns ".98"
     * 2165461.236 returns "2165461.23"
     *
     * @param doubleValue a floating point number
     * @return <code>String</code>, a String containing a number, set to two decimal places
     */
    public static String toDoubleDigit( double doubleValue ) {
        //String.format forces us to round the number up or down, and there don't seem to be any options to truncate.
        // This method cuts it down to 00.000 format, then substrings out the last decimal
        String formattedStr = String.format( "%-1.3f", doubleValue );
        return formattedStr.substring( 0, formattedStr.length() - 1 );
    }

    private StringUtil() {
    }

    /**
     * convert first letter of the string to upper case
     *
     * @param origin
     * @return
     */
    public static String toUpperCaseFirstLetter(String origin) {
        String ret = origin;
        if (StringUtils.isNotEmpty(origin)) {
            ret = origin.substring(0, 1).toUpperCase();
            ret = ret + origin.substring(1);
        }
        return ret;
    }

    /**
     * convert first letter of the string to lower case
     *
     * @param origin
     * @return
     */
    public static String toLowerCaseFirstLetter(String originStr) {
        String ret = null;
        if (StringUtils.isEmpty(originStr)) {
            ret = originStr;
        } else {
            ret = String.valueOf(originStr.charAt(0)).toLowerCase();
            ret = ret.concat(originStr.substring(1));
        }
        return ret;
    }

    /**
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.cla | SiteUtil.class | SpreadSheetUtil. | StringInputStrea | SSHUtil.class | StringUtil.class | SystemUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```
     * return get method name
     *
     * @param origin
     * @return
     */
    public static String getGetterName(String origin) {
        String ret = toUpperCaseFirstLetter(origin);
        if (StringUtils.isNotEmpty(ret)) {
            return "get" + ret;
        }
        return ret;
    }

    /**
     * return set method name
     *
     * @param origin
     * @return
     */
    public static String getSetterName(String origin) {
        String ret = toUpperCaseFirstLetter(origin);
        if (StringUtils.isNotEmpty(ret)) {
            return "set" + ret;
        }
        return ret;
    }

    /**
     * get the regex in string
     *
     * @param originalStr
     * @param regex
     * @return
     */
    public static String getMatchedRegex(String originalStr, String regex) {
        String ret = null;
        Pattern p1 = Pattern.compile(regex);
        Matcher m1 = p1.matcher(originalStr);
        if (m1.find()) {
            ret = m1.group(0);
        }
        return ret;
    }

    /**
     * replace with regex
     *
     * @param originalStr
     * @param regex
     * @return
     */
    public static String replaceWithRegex(String originalStr, String regex) {
        Pattern p1 = Pattern.compile(regex);
        Matcher m1 = p1.matcher(originalStr);
        return m1.replaceAll("");
    }

    public static boolean isEquals(String expected, String original){
        if(expected == null) return true;

        if(!expected.equals(original)){
            Logging.log(null,
                    "ASSERTION FAILED: expected is " + expected + ", " +
                    "but the displayed vaule is " + original, true, true);
            return false;
        }
        return true;
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | SystemUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```
package com.ebay.maui.util;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.lang.management.ManagementFactory;
import java.lang.management.RuntimeMXBean;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.apache.log4j.Level;
import org.apache.log4j.Logger;

import com.ebay.maui.controller.Logging;

public class SystemUtil {

    public static enum PsToolsCommandEnum {
        procexp, psexec, psfile, psgetsid, psinfo, pskill, pslist, psloggedon, psloglist, pspasswd, psservice, psshutdown, pssuspend, taskkill, tasklist;
    }

    private static Logger logger = Logging.getLogger(SystemUtil.class);

    public static final Map<Process, Date> processMap = Collections.synchronizedMap(new HashMap<Process, Date>());

    public static String PSTOOLS_FOLDER = System.getProperty("user.home") + "/pstools/";

    static {
        File f = new File(PSTOOLS_FOLDER + "/psexec.exe");
        if (!f.exists()) {
            new File(PSTOOLS_FOLDER).mkdirs();
            try {
                FileUtil.writeToFile(PSTOOLS_FOLDER + "/psexec.exe", SystemUtil.class.getResourceAsStream("psexec.dat"));
            } catch (IOException e) {
                logger.warn(e);
            }
        }
    }

    /**
     * Executes a remote process. Clears the standard output and standard error to avoid process blocking. Also wait for the process to complete if required.
     *
     * @param psCommand
     * @param command
     * @param workingFolder
     * @param hostname
     * @param user
     * @param password
     * @param wait
     * @return
     * @throws Exception
     */
    public static int exec(PsToolsCommandEnum psCommand, String command, String workingFolder, String hostname, String user, String password, boolean wait) throws Exception {

        if (!PsToolsCommandEnum.psshutdown.equals(psCommand) && (command == null || command.trim().length() == 0)) {
            throw new Exception("command is required.");
        }

        StringBuffer cmd = new StringBuffer("\"" + PSTOOLS_FOLDER);
        cmd.append(psCommand).append(".exe\" -accepteula");// AcceptEula will take care of automatically accepting License
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | SystemUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
        if (PsToolsCommandEnum.pskill.equals(psCommand)) {
            cmd.append(" -t ");
        } else if (PsToolsCommandEnum.psshutdown.equals(psCommand)) {
            cmd.append(" -r ");
        } else {
            if (wait) {
                cmd.append(" -n 10 ");
            } else {
                cmd.append(" -d ");
            }
            cmd.append(" -i 0 ");

            if (PsToolsCommandEnum.psexec.equals(psCommand) && workingFolder != null && workingFolder.trim().length() > 0) {
                cmd.append(" -w ").append(workingFolder);
            }
        }

        if (hostname != null && hostname.trim().length() > 0) {
            cmd.append(" \\\\").append(hostname);

            if (user == null || user.trim().length() == 0 || password == null || password.trim().length() == 0) {
                throw new Exception("username and password are required.");
            }

            cmd.append(" -u ").append(user);
            cmd.append(" -p ").append(password);
        }
        if (command != null)
            cmd.append(" ").append(command);

        // logger.info(cmd.toString());

        if (PsToolsCommandEnum.pskill.equals(psCommand)) {
            try {
                String procString = null;
                try {
                    Integer.parseInt(command);
                    procString = " /PID " + command;
                } catch (NumberFormatException e) {
                    procString = " /FI \"IMAGENAME eq " + command + "*\"";
                }
                Runtime.getRuntime().exec(PsToolsCommandEnum.taskkill + " /T /F /S " + hostname + " /U " + user + " /P " + password + procString);
            } catch (Throwable e) {
                Runtime.getRuntime().exec(cmd.toString());
            }
            return 0;
        }
        return exec(cmd.toString());
    }

    /**
     * Executes a native process. Clears the standard output and standard error to avoid process blocking. Also wait for the process to complete if required.
     *
     * @param cmd
     * @throws Exception
     */
    public static int exec(String cmd) throws Exception {
        Runtime rt = Runtime.getRuntime();
        Process p = rt.exec(cmd);

        processMap.put(p, new Date());

        byte buf[] = new byte[512];
        InputStream is = p.getInputStream();

        if (is.available() > 0) {
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | SystemUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
        if (is.available() > 0) {
            while (is.read(buf) > 0) {
            }
        }
        is.close();

        is = p.getErrorStream();
        if (is.available() > 0) {
            while (is.read(buf) > 0) {
            }
        }
        is.close();

        // return p.waitFor();
        return 0;
    }

    /**
     * Executes a remote process. Clears the standard output and standard error to avoid process blocking. Also wait for the process to complete if required.
     *
     * @param psCommand
     * @param command
     * @param hostname
     * @param user
     * @param password
     * @param wait
     * @return
     */
    public static int exec2(PsToolsCommandEnum psCommand, String command, String hostname, String user, String password, boolean wait) throws Exception {
        return exec(psCommand, command, null, hostname, user, password, wait);
    }

    /**
     * Executes a native process
     *
     * @param cmd
     * @return
     * @throws Exception
     */
    public static String exec2(String cmd) throws Exception {
        Process process = Runtime.getRuntime().exec(cmd);

        processMap.put(process, new Date());
        // process.waitFor();
        InputStreamReader isr = new InputStreamReader(process.getInputStream());
        BufferedReader br = new BufferedReader(isr);
        String line = null;
        StringBuffer sb = new StringBuffer();
        while ((line = br.readLine()) != null) {
            sb.append(line).append("\n");
        }
        br.close();
        isr.close();
        return sb.toString();
    }

    /**
     * Works for Windows
     *
     * @return
     */
    public static String getPID() {
        if (isWindowsPlatform()) {
            RuntimeMXBean rmxb = ManagementFactory.getRuntimeMXBean();
            return rmxb.getName().substring(0, rmxb.getName().indexOf('@'));
        }
        return System.getProperty("PID");
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | SystemUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
        return System.getProperty("PID");
    }

    /**
     * Checks if Windows PlatForm
     *
     * @return
     */
    public static boolean isWindowsPlatform() {
        String os = System.getProperty("os.name");
        if (os != null && os.startsWith("Windows"))
            return true;
        else
            return false;
    }

    /**
     * @param args
     */
    public static void main(String[] args) {
        try {
            psListWithPID("wbqalab69", "java", "", "");
        } catch (Exception e) {
            e.printStackTrace();// KEEPME
        }
    }

    public static int psExec(final String command, final String workingFolder, final String hostname, final String username, final String password, final boolean wait) throws Exception {
        Runnable r = new Runnable() {
            public void run() {
                try {
                    execCPsToolsCommandEnum.psexec, command, workingFolder, hostname, username, password, wait);
                } catch (Exception e) {
                    logger.log(Level.WARN, "ex", e);
                }
            }
        };
        Thread t = new Thread(r);
        t.start();

        if (wait)
            t.join();

        return 0;
    }

    public static boolean psFound(String hostname, String pid, final String username, final String password) throws Exception {

        String[] output = psList(hostname, pid, username, password);
        boolean found = true;
        for (int i = 0; i < output.length; i++) {
            if (output[i].indexOf("not found") > -1) {
                found = false;
                break;
            }
        }
        return found;
    }

    public static String psInfo(String hostname, final String username, final String password) throws Exception {

        StringBuffer cmd = new StringBuffer(PSTOOLS_FOLDER);
        cmd.append(PsToolsCommandEnum.psinfo.name()).append(".exe -accepteula");

        cmd.append(" -d ");

        if (hostname != null && hostname.trim().length() > 0) {
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | SystemUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
            cmd.append(" \\\\").append(hostname);

            cmd.append(" -u ").append(username);
            cmd.append(" -p ").append(password);
        }

        return exec2(cmd.toString());
    }

    public static int psKill(String hostname, String process, final String username, final String password) throws Exception {
        return exec(PsToolsCommandEnum.pskill, process, null, hostname, username, password, false);
    }

    public static String[] psList(String hostname, String pName, final String username, final String password) throws Exception {

        StringBuffer cmd = new StringBuffer(PSTOOLS_FOLDER);
        cmd.append(PsToolsCommandEnum.pslist.name()).append(".exe -accepteula");

        if (hostname != null && hostname.trim().length() > 0) {
            cmd.append(" \\\\").append(hostname);

            cmd.append(" -u ").append(username);
            cmd.append(" -p ").append(password);
        }
        if (pName != null) {
            cmd.append(" ");
            cmd.append(pName);
        }

        String output = exec2(cmd.toString());
        // System.out.println(output);
        if (output.indexOf("Process information") > -1) {
            output = output.substring(output.indexOf("Process information"));
        }

        return output.split("\\n");
    }

    /**
     * Provide the process List
     *
     * <PRE>
     * Name                Pid Pri Thd  Hnd   Priv        CPU Time    Elapsed Time
     * java               3320   8   9  589  94284     0:00:57.312   100:44:12.127
     * java               3640   8   9  613  95100     0:00:50.921   100:39:10.849
     * </PRE>
     *
     * @param hostname
     * @param pName
     * @return
     * @throws Exception
     */
    public static List<String[]> psListWithPID(String hostname, String pName, final String username, final String password) throws Exception {
        String[] output = psList(hostname, pName, username, password);
        List<String[]> processList = new ArrayList<String[]>();
        // processList.add(new String[]{"Name","Pid","Pri","Thd","Hnd","Priv","CPU Time","Elapsed Time"});
        for (int i = 2; i < output.length; i++) {
            if (output[i].indexOf("not found") != -1)
                break;
            if (output[i].indexOf(pName) > -1) {
                String s[] = output[i].split("\\s");
                List<String> attrList = new ArrayList<String>();
                for (int k = 0; k < s.length; k++) {
                    if (s[k].length() > 0)
                        attrList.add(s[k]);
                }
                processList.add(attrList.toArray(new String[attrList.size()]));
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | SystemUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
            output = output.substring(output.indexOf("Process information"));
        }

        return output.split("\\n");
    }

    /**
     * Provide the process List
     *
     * <PRE>
     * Name                 Pid Pri Thd  Hnd   Priv        CPU Time    Elapsed Time
     * java                3320   8   9  589  94284     0:00:57.312   100:44:12.127
     * java                3640   8   9  613  95100     0:00:50.921   100:39:10.849
     * </PRE>
     *
     * @param hostname
     * @param pName
     * @return
     * @throws Exception
     */
    public static List<String[]> psListWithPID(String hostname, String pName, final String username, final String password) throws Exception {
        String[] output = psList(hostname, pName, username, password);
        List<String[]> processList = new ArrayList<String[]>();
        // processList.add(new String[]{"Name","Pid","Pri","Thd","Hnd","Priv","CPU Time","Elapsed Time"});
        for (int i = 2; i < output.length; i++) {
            if (output[i].indexOf("not found") != -1)
                break;
            if (output[i].indexOf(pName) > -1) {
                String s[] = output[i].split("\\s");
                List<String> attrList = new ArrayList<String>();
                for (int k = 0; k < s.length; k++) {
                    if (s[k].length() > 0)
                        attrList.add(s[k]);
                }
                processList.add(attrList.toArray(new String[attrList.size()]));
            }
        }
        return processList;
    }

    public static int psShutdown(String hostname, final String username, final String password) throws Exception {
        return exec(PsToolsCommandEnum.psshutdown, null, null, hostname, username, password, false);
    }

    public static String replace(String target, String from, String to) {
        // target is the original string
        // from is the string to be replaced
        // to is the string which will used to replace
        // returns a new String!
        int start = target.indexOf(from);
        if (start == -1)
            return target;
        int lf = from.length();
        char[] targetChars = target.toCharArray();
        StringBuffer buffer = new StringBuffer();
        int copyFrom = 0;
        while (start != -1) {
            buffer.append(targetChars, copyFrom, start - copyFrom);
            buffer.append(to);
            copyFrom = start + lf;
            start = target.indexOf(from, copyFrom);
        }
        buffer.append(targetChars, copyFrom, targetChars.length - copyFrom);
        return buffer.toString();
    }
}
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
package com.ebay.maui.util;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.List;

import junit.framework.Assert;
import net.jsourcerer.webdriver.jserrorcollector.JavaScriptError;

import org.apache.log4j.Logger;
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.Cookie;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebDriverException;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.remote.LocalFileDetector;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

import com.ebay.maui.controller.ContextManager;
import com.ebay.maui.controller.Logging;
import com.ebay.maui.driver.web.BrowserType;
import com.ebay.maui.driver.web.CustomEventFiringWebDriver;
import com.ebay.maui.driver.web.ScreenshotUtil;
import com.ebay.maui.driver.web.WebDriverMode;
import com.ebay.maui.driver.web.WebUXDriver;
import com.ebay.maui.driver.web.element.HtmlElement;
import com.ebay.maui.helper.FileHelper;
import com.ebay.maui.helper.OSHelper;
import com.ebay.maui.helper.URLHelper;
import com.thoughtworks.selenium.Wait;

/**
 * This class provides reusable Web testing methods based on runtime driver from
 * WebUXDriver.
 *
 */
public class WebUXUtil {

    private static final Logger logger = Logging.getLogger(WebUXUtil.class);

    /**
     * Move forward in the browser's history
     */
    public static void forward() {
        WebUXDriver.getWebDriver().navigate().forward();
    }

    /**
     * Move back in the browser's history
     */
    public static void back() {
        WebUXDriver.getWebDriver().navigate().back();
    }

    /**
     * Refresh the current page
     */
    public static void refresh() {
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
    public static void refresh() {
        WebUXDriver.getWebDriver().navigate().refresh();
    }

    /**
     * Close the current window
     */
    public static void close() {
        WebUXDriver.getWebDriver().close();
    }

    /**
     * Quit driver and close every associated window
     */
    public static void quit() {
        WebUXDriver.getWebDriver().quit();
    }

    /**
     * Get a string representing the current URL that the browser is looking at
     *
     * @return The URL of the page currently loaded in the browser
     */
    public static String getCurrentURL() {
        return WebUXDriver.getWebDriver().getCurrentUrl();
    }

    /**
     * Accept alert and switch to default content
     */
    public static void acceptAlert() {
        WebDriver driver = WebUXDriver.getWebDriver();
        Alert alert = driver.switchTo().alert();
        alert.accept();
        driver.switchTo().defaultContent();
    }

    /**
     * Dismiss alert and switch to default content
     */
    public static void dissmissAlert() {
        WebDriver driver = WebUXDriver.getWebDriver();
        Alert alert = driver.switchTo().alert();
        alert.dismiss();
        driver.switchTo().defaultContent();
    }

    /**
     * Get text of alert
     *
     * @return text of alert
     */
    public static String getAlert() {
        Alert alert = WebUXDriver.getWebDriver().switchTo().alert();
        return alert.getText();
    }

    /**
     * Select a frame
     *
     * @param by
     */
    public void selectFrame(By by) {
        WebUXDriver.getWebDriver().switchTo()
                .frame(WebUXDriver.getWebDriver().findElement(by));
    }
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
    /**
     * Switch the focus of future commands to the new window with the handle.
     */
    public static void selectNewWindow() {
        final WebDriver driver = WebUXDriver.getWebDriver();
        new Wait() {
            // driver = WebUXDriver.getWebDriver();

            @Override
            public boolean until() {
                try {
                    Object[] handles = driver.getWindowHandles().toArray();
                    String handle = WebUXDriver.getWindowHandle();
                    if (handle.equals((String) handles[0])) {
                        driver.switchTo()
                                .window((String) driver.getWindowHandles()
                                        .toArray()[1]);
                    } else
                        driver.switchTo()
                                .window((String) driver.getWindowHandles()
                                        .toArray()[0]);
                    return true;
                } catch (WebDriverException e) {
                } catch (ArrayIndexOutOfBoundsException ex) {
                }
                return false;
            }
        }.wait(String.format("Timed out waiting for page to load"), WebUXDriver
                .getWebUXDriver().getWebSessionTimeout());
        System.out.println(driver.getWindowHandles());
    }

    /**
     * Switch the focus of future commands to the window with the next handle.
     */
    public static void selectWindow() {
        // Set<String> handles = WebUXDriver.getWebDriver().getWindowHandles();
        // selectWindow(handles.iterator().next());
        WebUXDriver.getWebDriver().switchTo()
                .window(WebUXDriver.getWindowHandle());
    }

    /**
     * Switch the focus of future commands to the window with the given name.
     *
     * @param windowName
     *            - The name of the window
     */
    public static void selectWindow(String windowName) {
        WebUXDriver.getWebDriver().switchTo().window(windowName);
    }

    /**
     * Switch the focus of future commands to the popup with the given name. The
     * same as selectWindow(String)
     *
     * @param name
     *            - The name of the popup
     */
    public static void selectPopUp(String name) {
        selectWindow(name);
    }

    /**
     * Switch the focus of future commands to the window with the given name and
     * load url.
     *
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
     * @param url
     * @param windowName
     */
    public static void openWindow(String url, String windowName) {
        WebDriver driver = WebUXDriver.getWebDriver();
        driver.switchTo().window(windowName);
        driver.get(url);
    }

    /**
     * Add cookie
     *
     * @param cookie
     */
    public static void addCookie(Cookie cookie) {
        WebUXDriver.getWebDriver().manage().addCookie(cookie);
    }

    /**
     * Delete all the cookies
     *
     */
    public static void deleteAllCookies() {
        WebUXDriver.getWebDriver().manage().deleteAllCookies();
    }

    /**
     * Get Cookie by name
     *
     * @param name
     */
    public static Cookie getCookieNamed(String name) {
        return WebUXDriver.getWebDriver().manage().getCookieNamed(name);
    }

    /**
     * Execute JavaScript
     *
     * @param script
     */
    public static void executeJavaScript(String script) {
        ((JavascriptExecutor) WebUXDriver.getWebDriver()).executeScript(script);
    }

    /**
     * Execute JavaScript
     *
     * @param driver
     * @param script
     */
    public static void executeJavaScript(WebDriver driver, String script) {
        ((JavascriptExecutor) driver).executeScript(script);
    }

    /**
     * Get JS Error by JSErrorCollector which only supports Firefox browser.
     *
     * @return jsErrors in format "line number, error message, source name; "
     */
    public static String getJSErrors() {
        if (ContextManager.getThreadContext().getAddJSErrorCollectorExtension()
                .equalsIgnoreCase("true")) {
            List<JavaScriptError> jsErrorList = JavaScriptError
                    .readErrors(WebUXDriver.getWebDriver());
            if (!jsErrorList.isEmpty()) {
                String jsErrors = "";
                for (int i = 0; i < jsErrorList.size(); i++) {
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
                for (int i = 0; i < jsErrorList.size(); i++) {
                    jsErrors += jsErrorList.get(i).getLineNumber() + ", "
                            + jsErrorList.get(i).getErrorMessage() + ", "
                            + jsErrorList.get(i).getSourceName() + "; ";
                }
                return jsErrors;
            }
        }
        return null;
    }

    /**
     * Wait element to present until timeout
     *
     * @param by
     * @param timeout
     */
    public static void waitForElementPresent(By by, int timeout) {
        Logging.logWebStep(null, "wait for " + by.toString()
                + " to be present.", false);
        WebDriverWait wait = new WebDriverWait(WebUXDriver.getWebDriver(),
                timeout);
        wait.until(ExpectedConditions.presenceOfElementLocated(by));
    }

    /**
     * Wait element to present until EXPLICIT_WAIT_TIME_OUT (default 15s)
     *
     * @param by
     */
    public static void waitForElementPresent(By by) {
        waitForElementPresent(by, WebUXDriver.getWebUXDriver()
                .getExplicitWait());
    }

    /**
     * Wait element to present until EXPLICIT_WAIT_TIME_OUT (default 15s)
     *
     * @param element
     */
    public static void waitForElementPresent(HtmlElement element) {
        waitForElementPresent(element.getBy());
    }

    /**
     * Wait element to be visible until timeout
     *
     * @param by
     * @param timeout
     */
    public static void waitForElementToBeVisible(By by, int timeout) {
        Logging.logWebStep(null, "wait for " + by.toString()
                + " to be visible.", false);
        WebDriverWait wait = new WebDriverWait(WebUXDriver.getWebDriver(),
                timeout);
        wait.until(ExpectedConditions.visibilityOfElementLocated(by));
    }

    /**
     * Wait element to be visible until EXPLICIT_WAIT_TIME_OUT (default 15s)
     *
     * @param by
     */
    public static void waitForElementToBeVisible(By by) {
        waitForElementToBeVisible(by, WebUXDriver.getWebUXDriver()
                .getExplicitWait());
    }
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
            .getExplicitWait());
    }

    /**
     * Wait element to be visible until EXPLICIT_WAIT_TIME_OUT (default 15s)
     *
     * @param element
     */
    public static void waitForElementToBeVisible(HtmlElement element) {
        waitForElementToBeVisible(element.getBy());
    }

    /**
     * Wait element to be invisible until timeout
     *
     * @param by
     * @param timeout
     */
    public static void waitForElementToDisappear(By by, int timeout) {
        Logging.logWebStep(null,
                "wait for " + by.toString() + " to disappear.", false);
        WebDriverWait wait = new WebDriverWait(WebUXDriver.getWebDriver(),
                timeout);
        wait.until(ExpectedConditions.invisibilityOfElementLocated(by));
    }

    /**
     * Wait element to be invisible until EXPLICIT_WAIT_TIME_OUT (default 15s)
     *
     * @param by
     */
    public static void waitForElementToDisappear(By by) {
        waitForElementToDisappear(by, WebUXDriver.getWebUXDriver()
                .getExplicitWait());
    }

    /**
     * Wait element to be invisible until EXPLICIT_WAIT_TIME_OUT (default 15s)
     *
     * @param element
     */
    public static void waitForElementToDisappear(HtmlElement element) {
        waitForElementToDisappear(element.getBy(), WebUXDriver.getWebUXDriver()
                .getExplicitWait());
    }

    /**
     * Wait element to be clickable until EXPLICIT_WAIT_TIME_OUT (default 15s)
     *
     * @param by
     */
    public void waitForElementToBeClickable(By by) {
        Logging.logWebStep(null, "wait for " + by.toString()
                + " to be clickable.", false);
        WebDriverWait wait = new WebDriverWait(WebUXDriver.getWebDriver(),
                WebUXDriver.getWebUXDriver().getExplicitWait());
        wait.until(ExpectedConditions.elementToBeClickable(by));
    }

    /**
     * Wait element to be selected until EXPLICIT_WAIT_TIME_OUT (default 15s)
     *
     * @param by
     */
    public void waitForElementToBeSelected(By by) {
        Logging.logWebStep(null, "wait for " + by.toString()
                + " to be selected.", false);
        WebDriverWait wait = new WebDriverWait(WebUXDriver.getWebDriver(),
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
        WebDriverWait wait = new WebDriverWait(WebUXDriver.getWebDriver(),
                WebUXDriver.getWebUXDriver().getExplicitWait());
        wait.until(ExpectedConditions.elementToBeSelected(by));
    }

    /**
     * Wait element to be active until EXPLICIT_WAIT_TIME_OUT (default 15s)
     *
     * @param by
     */
    public static void waitForElementToActive(By by) {
        Logging.logWebStep(null, "wait for " + by.toString() + " to active.",
                false);
        WebDriverWait wait = new WebDriverWait(WebUXDriver.getWebDriver(),
                WebUXDriver.getWebUXDriver().getExplicitWait());
        wait.until(ExpectedConditionsUtil.activeOfElementLocated(by));
    }

    /**
     * Wait element to be inactive until EXPLICIT_WAIT_TIME_OUT (default 15s)
     *
     * @param by
     */
    public static void waitForElementToInactive(By by) {
        Logging.logWebStep(null,
                "wait for " + by.toString() + " to disappear.", false);
        WebDriverWait wait = new WebDriverWait(WebUXDriver.getWebDriver(),
                WebUXDriver.getWebUXDriver().getExplicitWait());
        wait.until(ExpectedConditionsUtil.inactiveOfElementLocated(by));
    }

    /**
     * Wait web page to be loaded until WebSessionTimeout (default 90s). This
     * method cannot make sure all the elements are loaded completely as it just
     * makes driver switch to default content.
     */
    public static void waitForPageToLoad() {
        new Wait() {
            WebDriver driver = WebUXDriver.getWebDriver();

            @Override
            public boolean until() {
                try {
                    driver.switchTo().defaultContent();
                    return true;
                } catch (WebDriverException e) {
                }
                return false;
            }
        }.wait(String.format("Timed out waiting for page to load"), WebUXDriver
                .getWebUXDriver().getWebSessionTimeout());
    }

    /**
     * Check if an alert on a web page is present or not
     *
     * @return true if present
     */
    public static boolean isAlertPresent() {
        WebDriver driver = WebUXDriver.getWebDriver();
        try {
            driver.switchTo().alert();
        } catch (Exception ex) {
            return false;
        }
        driver.switchTo().defaultContent();
        return true;
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
     * Check if an element on a web page is present or not
     *
     * @return true if present
     */
    public static boolean isElementPresent(By by) {
        try {
            return WebUXDriver.getWebDriver().findElements(by).size() != 0;
        } catch (Exception ex) { // For OperaDriver
            return false;
        }
    }

    /**
     * Assert if text in page source is present or not
     *
     * @return true if present
     */
    public void assertTextPresentInSource(String text) {
        Assert.assertTrue(WebUXDriver.getWebDriver().getPageSource()
                .contains(text));
    }

    /**
     * Get the number of elements with xPath
     *
     * @param xPath
     * @return the number of elements
     */
    public static Number getXpathCount(String xPath) {
        WebDriver driver = WebUXDriver.getWebDriver();
        return driver.findElements(By.xpath(xPath)).size();
    }

    /**
     * Capture snapshot
     *
     * @param messagePrefix
     */
    public static void captureSnapshot(String messagePrefix) {
        ScreenshotUtil.captureSnapshot(messagePrefix);
    }

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

```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
    /**
     * Download a file to browserDownloadDir, default is output folder +
     * downloads
     *
     * @param url
     * @param convert
     *            - true to convert
     */
    public static void downloadFile(String url, Boolean convert) {
        if (convert) {
            String tempURL = convert(url);
            url = tempURL;
        }
        WebUXDriver.getWebDriver(true).get(url);
        Logging.log("Download file to "
                + ContextManager.getThreadContext().getBrowserDownloadDir());
    }

    /**
     * Log content for HTML report
     *
     * @param content
     * @param title
     */
    public static void logContent(String content, String title) {
        if (content != null) {
            String outputDir = ContextManager.getGlobalContext()
                    .getOutputDirectory();
            String inputfilename = URLHelper.getRandomHashCode("open-url");
            String subFolder = ContextManager.getGlobalContext()
                    .getTestNGContext().getSuite().getName();
            try {
                FileHelper.writeToFile(outputDir + "/xmls/" + inputfilename
                        + ".xml", content);
            } catch (IOException e) {
                e.printStackTrace();
            }
            Logging.log(title + "  <a href='" + subFolder + "/xmls/"
                    + inputfilename + ".xml' target=content>content</a>");
        }
    }

    /**
     * Upload file works for both local and grid
     *
     * @param path
     *            - the path of the file
     * @param by
     */
    public static void uploadFile(String path, By by) {
        if (OSHelper.isWindows()) {
            path = path.replaceAll("/", "\\\\");
            if (path.startsWith("\\"))
                path = path.substring(1);
        }
        if (WebUXDriver.getWebUXDriver().getConfig().getMode() == WebDriverMode.ExistingGrid) {
            org.openqa.selenium.remote.RemoteWebDriver driver = (RemoteWebDriver) ((CustomEventFiringWebDriver) WebUXDriver
                    .getWebDriver()).getWebDriver();
            LocalFileDetector detector = new LocalFileDetector();
            File file = detector.getLocalFile(path);
            driver.setFileDetector(detector);
            driver.findElement(by).sendKeys(file.getAbsolutePath());
        } else {
            WebDriver driver = WebUXDriver.getWebDriver();
            driver.findElement(by).sendKeys(path);
        }
    }
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
    /**
     * Use to save a remote file into local output directory
     *
     * @Sample String file =
     *         WebUXUtil.saveFile("http://picturesplatform.qaebay.com/qa_images/3.jpg"
     *         );
     * @param url
     * @return the path of local file, such as
     *         C:\GitWorkSpace\QE-MAUI-UTIL\mauiutil\test-output\Default
     *         suite\downloads\3ddd8709e16e2e0cf89d6a6f78e2f75c-save.jpg
     * @throws Exception
     */
    public static String saveFile(String url) throws Exception {
        // url = URLHelper.encode(url);
        String fileNamePostFix = "save";
        InputStream inputStream = URLHelper.openAsStream(url);
        String fileType = url.substring(url.lastIndexOf(".") + 1);
        StringBuffer sbMessage = new StringBuffer("");
        String fileName = URLHelper.getRandomHashCode("download") + "-"
                + fileNamePostFix;
        if (fileType != null && fileType.length() < 5)
            fileName = fileName + "." + fileType;
        String outputDirectory = ContextManager.getGlobalContext()
                .getTestNGContext().getOutputDirectory();
        String suiteName = ContextManager.getGlobalContext().getTestNGContext()
                .getSuite().getName();
        try {

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

            OutputStream out = new FileOutputStream(downloadDir + fileName);
            int read = 0;
            byte[] bytes = new byte[1024];

            while ((read = inputStream.read(bytes)) != -1) {
                out.write(bytes, 0, read);
            }
            out.close();

            sbMessage.append("Downloading file <a href='" + suiteName
                    + "/downloads/" + fileName + " title='" + fileName + "'>"
                    + fileNamePostFix + "</a>");
        } catch (Throwable e) {
            logger.warn("Ex", e);
        }

        Logging.logWebOutput(url, sbMessage.toString(), false);
        fileName = outputDirectory + "/downloads/" + fileName;
        // correct file path format
        if (OSHelper.isWindows())
            fileName = fileName.replaceAll("/", "\\\\");

        return fileName;
    }

    /**
     * Get browser
     *
     */
    public static BrowserType getBrowserType() {
        return WebUXDriver.getWebUXDriver().getConfig().getBrowser();
    }

```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
        fileName = outputDirectory + "/downloads/" + fileName;
        // correct file path format
        if (OSHelper.isWindows())
            fileName = fileName.replaceAll("/", "\\\\");

        return fileName;
    }

    /**
     * Get browser
     *
     */
    public static BrowserType getBrowserType() {
        return WebUXDriver.getWebUXDriver().getConfig().getBrowser();
    }

    /**
     * Get runtime browser version
     *
     */
    public static String getBrowserVersion() {
        WebDriver driver = WebUXDriver.getNativeWebDriver();
        return ((RemoteWebDriver) (driver)).getCapabilities().getVersion();
    }

    /**
     * Get running mode
     *
     */
    public static WebDriverMode getRunningMode() {
        return WebUXDriver.getWebUXDriver().getConfig().getMode();
    }

    public static void scrollIntoView(WebElement element) {
        ((JavascriptExecutor) WebUXDriver.getWebDriver()).executeScript(
                "arguments[0].scrollIntoView(true);", element);
    }

    public static void scrollIntoViewBottom(WebElement element) {
        ((JavascriptExecutor) WebUXDriver.getWebDriver()).executeScript(
                "arguments[0].scrollIntoView(false);", element);
    }

    /**
     * Get all the matched URLs on current page according to the pattern
     *
     * @param patternList
     */
    public static List<String> getMatchedURLs(List<String> patternList) {
        List<String> urlList = new ArrayList<String>();
        List<WebElement> linksList = WebUXDriver.getWebDriver().findElements(
                By.tagName("a"));
        for (WebElement linkElement : linksList) {
            String url = linkElement.getAttribute("href");
            if (url != null) {
                Logging.log("Checking URL: " + url);
                for (String parttern : patternList) {
                    if (url.contains(parttern)) {
                        urlList.add(url);
                    }
                }
            }
        }
        return urlList;
    }
}
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
package com.ebay.maui.util;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.List;

import org.apache.commons.lang.StringUtils;
import org.dom4j.Attribute;
import org.dom4j.Document;
import org.dom4j.DocumentException;
import org.dom4j.Element;
import org.dom4j.io.SAXReader;
import org.testng.ITestContext;

import com.ebay.maui.controller.Context;
import com.ebay.maui.controller.ContextManager;
import com.ebay.maui.exception.MauiException;
import com.ebay.maui.util.internal.entity.PageElement;
import com.ebay.maui.util.internal.entity.TestObject;
import com.ebay.maui.util.internal.type.XmlTagName;

/**
 * Deal with data provider as XML file
 *
 * @author yimeng@ebay.com(Catherine
 *
 */
public class XmlUtil {

    private static final String VERTICAL = "V";
    private static final String HORIZONTAL = "H";

    /**
     * get page element from XML file
     *
     * @param clazz
     * @param fileName
     * @return
     * @throws MauiException
     */
    @SuppressWarnings("unchecked")
    public static LinkedHashMap<String, PageElement> getPageElementsFromXml(Class<?> clazz, String fileName)
            throws MauiException {
        LinkedHashMap<String, PageElement> elementMap = null;
        Element root = getRootElement(clazz, fileName);
        // deal with page elements
        if (XmlTagName.PAGE_ELEMENTS.getTagName().equals(root.getName())) {
            String type = root.attributeValue(XmlTagName.TYPE.getTagName());
            if (StringUtils.isEmpty(type)) {
                type = HORIZONTAL;
            }
            elementMap = new LinkedHashMap<String, PageElement>();
            for (Iterator<Element> i = root.elementIterator(); i.hasNext();) {
                Element xmlElement = i.next();
                PageElement pageElement = (PageElement) ReflectUtil.setDataToObject(PageElement.class,
                        getItemsMapFromXmlElement(type, xmlElement));
                elementMap.put(pageElement.getName(), pageElement);
            }
        }
        return elementMap;
    }
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
     * get test case information from XML
     *
     * @param clazz
     *            TestPlan
     * @param fileName
     *            XML file name
     * @param entityClazzMap
     *            java object
     * @return
     * @throws MauiException
     */
    public static Iterator<Object[]> getObjectsFromXml(Class<?> clazz, ITestContext testContex) throws Exception {
        return getObjectsFromXml(clazz, testContex, null);
    }

    /**
     * get test case information from XML
     *
     * @param clazz
     *            TestPlan
     * @param fileName
     *            XML file name
     * @param entityClazzMap
     *            java object
     * @return
     * @throws MauiException
     */
    public static Iterator<Object[]> getObjectsFromXml(Class<?> clazz, ITestContext testContex, String fileNames)
            throws Exception {

        if (StringUtils.isEmpty(fileNames)) {
            fileNames = System.getProperty(Context.TEST_DATA_FILE);
            if (StringUtils.isEmpty(fileNames)) {
                fileNames = ContextManager.getTestLevelContext(testContex).getTestDataFile();
            }
        }

        List<Object[]> mutiXmlData = new ArrayList<Object[]>();
        String[] fileNameList = fileNames.split(",");
        String siteCd = getSiteCode(testContex);
        for (String fm : fileNameList) {
            fm = fm.contains(".xml") ? fm.trim() : fm.trim() + ".xml";
            mutiXmlData.addAll(getObjectsFromXmlToList(clazz, fm, null, siteCd));
        }
        return mutiXmlData.iterator();
    }

    /**
     * get test case information from XML
     *
     * @param clazz
     *            TestPlan
     * @param fileName
     *            XML file name
     * @param testContex
     *            global context
     * @param method
     *            test method
     * @return
     * @throws MauiException
     */
    public static Iterator<Object[]> getObjectsFromXml(Class<?> clazz, ITestContext testContex, String fileNames,
            String method) throws Exception {
        if (StringUtils.isEmpty(fileNames)) {
            fileNames = System.getProperty(Context.TEST_DATA_FILE);
            if (StringUtils.isEmpty(fileNames)) {
                fileNames = ContextManager.getTestLevelContext(testContex).getTestDataFile();
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
136                 fileNames = ContextManager.getTestLevelContext(testContex).getTestDataFile();
137             }
138         }
139
140         List<Object[]> mutiXmlData = new ArrayList<Object[]>();
141         String[] fileNameList = fileNames.split(",");
142         String siteCd = getSiteCode(testContex);
143         for (String fm : fileNameList) {
144             fm = fm.contains(".xml") ? fm.trim() : fm.trim() + ".xml";
145             mutiXmlData.addAll(getObjectsFromXmlToList(clazz, fm, method, siteCd));
146         }
147         return mutiXmlData.iterator();
148     }
149
150     private static String getSiteCode(ITestContext ctx) {
151         String siteCd = ContextManager.getTestLevelContext(ctx).getSite();
152         if (StringUtils.isEmpty(siteCd)) {
153             siteCd = ContextManager.getGlobalContext().getSite();
154         }
155         return siteCd;
156     }
157
158     /**
159      * get test case information from XML
160      *
161      * @param clazz
162      *            TestPlan
163      * @param fileName
164      *            XML file name
165      * @param entityClazzMap
166      *            java object
167      * @return
168      * @throws MauiException
169      * @throws ClassNotFoundException
170      */
171     public static List<Object[]> getObjectsFromXmlToList(Class<?> clazz, String fileName, String method)
172             throws Exception {
173         return getObjectsFromXmlToList(clazz, fileName, method, null);
174     }
175
176     /**
177      * get test case information from XML
178      *
179      * @param clazz
180      *            TestPlan
181      * @param fileName
182      *            XML file name
183      * @param entityClazzMap
184      *            java object
185      * @return
186      * @throws MauiException
187      * @throws ClassNotFoundException
188      */
189     @SuppressWarnings("unchecked")
190     public static List<Object[]> getObjectsFromXmlToList(Class<?> clazz, String fileName, String method, String siteCd)
191             throws Exception {
192         List<Object[]> xmlData = null;
193
194         Element root = getRootElement(clazz, fileName);
195
196         // deal with test data
197         xmlData = new ArrayList<Object[]>();
198         String type = root.attributeValue(XmlTagName.TYPE.getTagName());
199
200         // class name map
201         HashMap<String, String> classMap = new HashMap<String, String>();
202         List<Element> classList = root.elements();
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
202         List<Element> classList = root.elements();
203         for (Element c : classList) {
204             String className = c.attributeValue(XmlTagName.ELEMENT_CLASS.getTagName());
205             if (StringUtils.isNotEmpty(className)) {
206                 classMap.put(c.getName(), className);
207             }
208         }
209         // test cases
210         TestObject testObj = null;
211         List<Attribute> caseAttributes = null;
212         Element testCase = null;
213         List<Object> objList = null;
214         for (Iterator<Element> i1 = root.elementIterator(XmlTagName.TEST_CASE.getTagName()); i1.hasNext();) {
215             testCase = i1.next();
216             objList = new ArrayList<Object>();
217             caseAttributes = testCase.attributes();
218             testObj = new TestObject();
219             for (Attribute attr : caseAttributes) {
220                 ReflectUtil.invokeSetter(testObj.getClass(), testObj, attr.getName(), attr.getValue());
221             }
222             if ((StringUtils.isEmpty(method) || method.equals(testObj.getTestMethod()))
223                 && (StringUtils.isEmpty(testObj.getTestSite()) || testObj.getTestSite().equals(siteCd))) {
224                 objList.add(testObj);
225
226                 List<Element> elements = testCase.elements();
227                 for (Element e : elements) {
228                     Class<?> c;
229                     try {
230                         String cName = e.attributeValue(XmlTagName.ELEMENT_CLASS.getTagName());
231                         if (StringUtils.isEmpty(cName)) {
232                             cName = classMap.get(e.getName());
233                         }
234                         if (cName == null) {
235                             throw new MauiException("No defined class object - [" + e.getName()
236                                     + "], Please check your xml file.");
237                         }
238                         c = Class.forName(cName);
239                     } catch (ClassNotFoundException e1) {
240                         e1.printStackTrace();
241                         throw new MauiException(e1);
242                     }
243                     String eType = e.attributeValue(XmlTagName.TYPE.getTagName());
244                     if (StringUtils.isNotEmpty(eType)) {
245                         eType = type;
246                     }
247                     // Object o = ReflectUtil.setDataToObject(c,
248                     // getItemsMapFromXmlElement(type, e));
249                     Object o = getXmlObject(c, type, e);
250                     objList.add(o);
251                 }
252
253                 if (objList != null && !objList.isEmpty()) {
254                     xmlData.add(objList.toArray(new Object[] { objList.size() }));
255                 }
256             }
257         }
258
259         return xmlData;
260     }
261
262     /**
263      * put the XML element to linked hash map
264      *
265      * @param type
266      *            Horizontal/Vertical
267      * @param xmlElement
268      * @return
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
270     @SuppressWarnings("unchecked")
271     public static LinkedHashMap<String, String> getItemsMapFromXmlElement(String type, Element xmlElement) {
272         LinkedHashMap<String, String> itemsMap = new LinkedHashMap<String, String>();
273         if (VERTICAL.equals(type)) {
274             for (Iterator<Element> i = xmlElement.elementIterator(); i.hasNext();) {
275                 Element item = i.next();
276                 itemsMap.put(item.getName(), String.valueOf(item.getData()));
277             }
278         } else {
279             for (Iterator<Attribute> i = xmlElement.attributeIterator(); i.hasNext();) {
280                 Attribute a = i.next();
281                 if (StringUtils.isNotEmpty(a.getName()) && !XmlTagName.ELEMENT_CLASS.getTagName().equals(a.getName())) {
282                     itemsMap.put(a.getName(), String.valueOf(a.getData()));
283                 }
284             }
285         }
286         return itemsMap;
287     }
288
289     private static Object getXmlObject(Class<?> clazz, String type, Element xmlElement) throws Exception {
290         if (VERTICAL.equals(type)) {
291             return getVerticalElement(clazz, xmlElement);
292         } else {
293             return getHerizentalElement(clazz, xmlElement);
294         }
295     }
296
297     @SuppressWarnings("unchecked")
298     private static Object getVerticalElement(Class<?> clazz, Element xmlElement) throws Exception {
299         Object objInstance = clazz.newInstance();
300
301         for (Iterator<Element> i = xmlElement.elementIterator(); i.hasNext();) {
302             Element item = i.next();
303             if (item.isTextOnly()) {
304                 ReflectUtil.invokeSetter(clazz, objInstance, item.getName(), item.getText().trim());
305             } else {
306                 Class<?> subClazz = clazz.getDeclaredField(item.getName()).getType();
307                 Object o = getVerticalElement(subClazz, item);
308                 ReflectUtil.invokeSetter(clazz, objInstance, item.getName(), o);
309             }
310         }
311         return objInstance;
312     }
313
314     @SuppressWarnings("unchecked")
315     private static Object getHerizentalElement(Class<?> clazz, Element xmlElement) throws Exception {
316         Object objInstance = clazz.newInstance();
317
318         for (Iterator<Attribute> i = xmlElement.attributeIterator(); i.hasNext();) {
319             Attribute a = i.next();
320             if (StringUtils.isNotEmpty(a.getName()) && !XmlTagName.ELEMENT_CLASS.getTagName().equals(a.getName())) {
321                 ReflectUtil.invokeSetter(clazz, objInstance, a.getName(), String.valueOf(a.getData()));
322             }
323         }
324         return objInstance;
325     }
326
327     public static Element getRootElement(Class<?> clazz, String fileName) {
328         Element root = null;
329         InputStream is = null;
330         try {
331             if (clazz != null)
332                 is = clazz.getResourceAsStream(fileName);
333             else
334                 is = new FileInputStream(fileName);
335
336             if (is == null) {
```

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XmlUtil.class | XPathUtil.class

```java
294         }
295     }
296
297     @SuppressWarnings("unchecked")
298     private static Object getVerticalElement(Class<?> clazz, Element xmlElement) throws Exception {
299         Object objInstance = clazz.newInstance();
300
301         for (Iterator<Element> i = xmlElement.elementIterator(); i.hasNext();) {
302             Element item = i.next();
303             if (item.isTextOnly()) {
304                 ReflectUtil.invokeSetter(clazz, objInstance, item.getName(), item.getText().trim());
305             } else {
306                 Class<?> subClazz = clazz.getDeclaredField(item.getName()).getType();
307                 Object o = getVerticalElement(subClazz, item);
308                 ReflectUtil.invokeSetter(clazz, objInstance, item.getName(), o);
309             }
310         }
311         return objInstance;
312     }
313
314     @SuppressWarnings("unchecked")
315     private static Object getHerizentalElement(Class<?> clazz, Element xmlElement) throws Exception {
316         Object objInstance = clazz.newInstance();
317
318         for (Iterator<Attribute> i = xmlElement.attributeIterator(); i.hasNext();) {
319             Attribute a = i.next();
320             if (StringUtils.isNotEmpty(a.getName()) && !XmlTagName.ELEMENT_CLASS.getTagName().equals(a.getName())) {
321                 ReflectUtil.invokeSetter(clazz, objInstance, a.getName(), String.valueOf(a.getData()));
322             }
323         }
324         return objInstance;
325     }
326
327     public static Element getRootElement(Class<?> clazz, String fileName) {
328         Element root = null;
329         InputStream is = null;
330         try {
331             if (clazz != null)
332                 is = clazz.getResourceAsStream(fileName);
333             else
334                 is = new FileInputStream(fileName);
335
336             if (is == null) {
337                 throw new FileNotFoundException("File is not exsited, please check your file name. FileName - "
338                         + fileName);
339             }
340
341             SAXReader reader = new SAXReader();
342             Document document = reader.read(is);
343             root = document.getRootElement();
344         } catch (FileNotFoundException e) {
345             throw new RuntimeException(e.getMessage());
346         } catch (DocumentException e) {
347             throw new RuntimeException(e.getMessage());
348         } finally {
349             try {
350                 if (is != null) {
351                     is.close();
352                 }
353             } catch (IOException e) {
354                 e.printStackTrace();
355             }
356         }
357         return root;
358     }
359 }
360
```

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XPathUtil.class

```java
package com.ebay.maui.util;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.io.StringWriter;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.apache.xpath.XPathAPI;
import org.jdom.Element;
import org.jdom.Namespace;
import org.jdom.input.SAXBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.w3c.tidy.Tidy;

import com.ebay.maui.exception.XMLException;

public class XPathUtil {

    /**
     * Return the document node of a HTML
     *
     * This method has some limitation, it can't always work as expected, so far, we haven't fixed it.
     *
     * @param htmlContent
     * @return
     * @throws XMLException
     * @deprecated
     */
    public static Document ~~getHTMLDocumentNode~~(String htmlContent) throws XMLException {

        Tidy tidy = new Tidy(); // obtain a new Tidy instance
        tidy.setXHTML(false); // set desired config options using tidy setters
        tidy.setFixBackslash(true);
        StringWriter sw = new StringWriter();
        tidy.setErrout(new PrintWriter(sw));
        tidy.setShowErrors(5);
        tidy.setShowWarnings(true);

        try {
            byte[] bytes = htmlContent.getBytes("UTF-8");
            InputStream input = new ByteArrayInputStream(bytes);
            return tidy.parseDOM(input, (OutputStream) null);
        } catch (Exception e) {
            throw new XMLException(e);
        }
    }

    /***
     * Get xml content for the node
     * @author junjshi
     * @param node
     * @return
     */
    public static String getNodeXMLContent(Node node)
    {
        return nodeToString(node,"",new StringBuilder()).toString();
    }

```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XPathUtil.class

```java
    /**
     * Return the document node of a XML
     *
     * @param xmlContent
     * @return
     * @throws XMLException
     */
    public static Document getXMLDocumentNode(String xmlContent) throws XMLException {

        try {
            DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
            byte[] bytes = xmlContent.getBytes("UTF-8");
            InputStream input = new ByteArrayInputStream(bytes);
            return builder.parse(input);
        } catch (Exception e) {
            throw new XMLException(e);
        }
    }

    public static String getXMLRootName(String xmlContent) throws XMLException {
        try {
            SAXBuilder builder = new SAXBuilder();
            org.jdom.Document document = builder.build(new StringInputStream(xmlContent));
            Element root = document.getRootElement();

            String name = root.getName();
            if (name.indexOf("Envelope") > -1) {
                org.jdom.Element body = root.getChild("Body", Namespace.getNamespace("http://schemas.xmlsoap.org/soap/envelope/"));
                name = ((org.jdom.Element) body.getChildren().get(0)).getName();
            }
            return name;

        } catch (Exception e) {
            e.printStackTrace(); // KEEPME
        }
        return "";
    }

    /**
     * Returns XPath Node content from XML Document.
     *
     * @param doc
     * @param xpathExpression
     * @return
     * @throws XMLException
     */
    public static String[] getXPathContent(Document doc, String xpathExpression) throws XMLException {

        try {
            String[] nodeAndAttributes = new String[] { xpathExpression };
            if (xpathExpression.contains("/@")) {
                nodeAndAttributes = xpathExpression.split("/@");
            }

            NodeList nList = XPathAPI.selectNodeList(doc.getDocumentElement(), nodeAndAttributes[0]);

            List<String> valueList = new ArrayList<String>();
            for (int i = 0; i < nList.getLength(); i++) {
                if (nodeAndAttributes.length == 1) {
                    if (nList.item(i).getTextContent() != null)
                        valueList.add(nList.item(i).getTextContent());
                    else
                        valueList.add(nList.item(i).getFirstChild().getNodeValue());
                } else {
                    valueList.add(nList.item(i).getAttributes().getNamedItem(nodeAndAttributes[1]).getNodeValue());
                }
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XPathUtil.class

```java
            }

            if (valueList.size() > 0) {
                return valueList.toArray(new String[valueList.size()]);
            }
        } catch (Exception e) {
            throw new XMLException(e);
        }

        return null;
    }

    /**
     * Returns XPath Node content from a XML.
     *
     * @param xmlContent
     * @param xpathExpression
     * @return
     * @throws XMLException
     */
    public static String[] getXPathContent(String xmlContent, String xpathExpression) throws XMLException {

        return getXPathContent(getXMLDocumentNode(xmlContent), xpathExpression);
    }


    /**
     * Returns XPath Node content from a HTML.
     *
     * @param htmlContent
     * @param xpathExpression
     * @return
     * @throws XMLException
     */
    public static String[] getXPathContentInHTML(String htmlContent, String xpathExpression) throws XMLException {

        return getXPathContent(getHTMLDocumentNode(htmlContent), xpathExpression);
    }

    public static NodeList getXPathNodeInHTML(String htmlContent, String xpathExpression) throws XMLException {
        Document doc = getHTMLDocumentNode(htmlContent);

        try {
            String[] nodeAndAttributes = new String[] { xpathExpression };
            if (xpathExpression.contains("/@")) {
                nodeAndAttributes = xpathExpression.split("/@");
            }

            NodeList nList = XPathAPI.selectNodeList(doc.getDocumentElement(), nodeAndAttributes[0]);

            return nList;
        } catch (Exception e) {
            throw new XMLException(e);
        }
    }

    /**
     * get nodes from the xpath query in the doc
     * @param doc
     * @param xpathExpression
     * @return
     * @throws XMLException
     */
    private static NodeList getXPathNodes(Document doc, String xpathExpression) throws XMLException {

        try {
            String[] nodeAndAttributes = new String[] { xpathExpression };
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XPathUtil.class

```java
 try {
     String[] nodeAndAttributes = new String[] { xpathExpression };
     if (xpathExpression.contains("/@")) {
         nodeAndAttributes = xpathExpression.split("/@");
     }

     NodeList nList = XPathAPI.selectNodeList(doc.getDocumentElement(), nodeAndAttributes[0]);

     return nList;
 } catch (Exception e) {
     throw new XMLException(e);
 }


 Get nodes from the xpath query in the xml content
 @param xmlContent
 @param xpathExpression
 @return
 @throws XMLException

lic static NodeList getXPathNodes(String xmlContent, String xpathExpression) throws XMLException {

 return getXPathNodes(getXMLDocumentNode(xmlContent), xpathExpression);


 Returns XML format XPath Node content from a XML.

 @param xmlContent
 @param xpathExpression
 @return
 @throws XMLException

lic static String[] getXPathXMLContent(String xmlContent, String xpathExpression) throws XMLException {

 NodeList nodes = getXPathNodes(getXMLDocumentNode(xmlContent), xpathExpression);

 if(nodes.getLength()>0)
 {
     String[] contents = new String[(nodes.getLength())];
     for(int i=0;i<nodes.getLength();i++)
     {
         contents[i] = getNodeXMLContent(nodes.item(i));
     }
     return contents;
 }else
     return null;


 Convert node to xml content
 @author junjshi
 @param node
 @param blank
 @param builder
 @return

vate static StringBuilder nodeToString(Node node,String blank,StringBuilder builder)


 if(node.getNodeType()==3){
     if(node.getParentNode().getChildNodes().getLength()==1 )
     {
         builder.append(node.getParentNode().getTextContent());
     }
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XPathUtil.class

```java
     * @param blank
     * @param builder
     * @return
     */
    private static StringBuilder nodeToString(Node node,String blank,StringBuilder builder)
    {

        if(node.getNodeType()==3){
            if(node.getParentNode().getChildNodes().getLength()==1 )
            {
                builder.append(node.getParentNode().getTextContent());
            }
            return builder;
        }else if (node.getNodeType()==4)
        {
            builder.append("<![CDATA["+node.getTextContent()+"]]>");
            return builder;
        }
        builder.append("\n");
        builder.append(blank+"<"+node.getNodeName());


        if(node.hasAttributes())
        {
            NamedNodeMap attrs = node.getAttributes();
            for(int j=0;j<attrs.getLength();j++){
                builder.append(" "+attrs.item(j).getNodeName()+"=\""+attrs.item(j).getNodeValue()+"\"");
            }
        }
        builder.append(">");
        NodeList list = node.getChildNodes();
        for(int i=0;i<list.getLength();i++)
        {

            Node item = list.item(i);
            nodeToString(item,blank+"\t",builder);
        }

        if(node.getChildNodes().getLength()>1)
            builder.append(blank);
        builder.append("</"+node.getNodeName()+">\n");
        return builder;
    }

    /**
     * This method ensures that the output String has only valid XML unicode characters as specified by the XML 1.0 standard. For reference, please see <a href="http://www.w3.org/TR/2000/REC-xml-20001006#NT-Char">the s
     * an empty String if the input is null or empty.
     *
     * @param in
     *          The String whose non-valid characters we want to remove.
     * @return The in String, stripped of non-valid characters.
     */
    public static String stripNonValidXMLCharacters(String in) {
        StringBuffer out = new StringBuffer(); // Used to hold the output.
        char current; // Used to reference the current character.

        if (in == null || ("".equals(in)))
            return ""; // vacancy test.
        for (int i = 0; i < in.length(); i++) {
            current = in.charAt(i); // NOTE: No IndexOutOfBoundsException caught here; it should not happen.
            if ((current == 0x9) || (current == 0xA) || (current == 0xD) || ((current >= 0x20) && (current <= 0xD7FF)) || ((current >= 0xE000) && (current <= 0xFFFD)) || ((current >= 0x10000) && (current <= 0x10FFFF)))
                out.append(current);
        }
        return out.toString();
    }
}
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | XPathUtil.class

```
eToString(Node node,String blank,StringBuilder builder)



getChildNodes().getLength()==1 )

getParentNode().getTextContent());


==4)

["+node.getTextContent()+"]]>");



e.getNodeName());



e.getAttributes();
ength();j++){
ttrs.item(j).getNodeName()+"=\""+attrs.item(j).getNodeValue()+"\"");



dNodes();
();i++)

;
"\t",builder);


ength()>1)

NodeName()+">\n");



output String has only valid XML unicode characters as specified by the XML 1.0 standard. For reference, please see <a href="http://www.w3.org/TR/2000/REC-xml-20001006#NT-Char">the standard</a>. This method will return
is null or empty.


on-valid characters we want to remove.
ed of non-valid characters.

ldXMLCharacters(String in) {
gBuffer(); // Used to hold the output.
erence the current character.

(in)))
st.
h(); i++) {
// NOTE: No IndexOutOfBoundsException caught here; it should not happen.
(current == 0xA) || (current == 0xD) || ((current >= 0x20) && (current <= 0xD7FF)) || ((current >= 0xE000) && (current <= 0xFFFD)) || ((current >= 0x10000) && (current <= 0x10FFFF)))
```

Read-Only | Smart Insert | 1 : 1

```java
package com.ebay.maui.util.internal.context;

import org.testng.ITestContext;

import com.ebay.maui.controller.Context;
import com.ebay.maui.controller.IContexAttributeListener;

public class APIContext implements IContexAttributeListener {

    /*---------------------API Context----------------------------*/
    public static final String API_SCHEMA_END_POINT = "apiSchmaEndPoint";

    public static final String API_SOAP_END_POINT = "apiSoapEndPoint";

    public static final String API_VERSION = "apiVersion";
    public static final String API_CERT_NAME = "apiCertName";
    public static final String API_APP_NAME = "apiAppName";
    public static final String API_DEV_NAME = "apiDevName";
    public static final String API_DETAIL_LEVEL = "apiDetailLevel";
    public static final String API_CS_END_POINT = "apiCSEndPoint";
    public static final String API_CS1_END_POINT = "apiCS1EndPoint";
    public static final String API_CS_APP_NAME = "apiCSAppName";

    public static final String API_CS_DEV_NAME = "apiCSDevName";
    public static final String API_CS_CERT_NAME = "apiCSCertName";
    public static final String API_RTM_END_POINT = "apiRtmEndPoint";
    /*------------------API Header Constant-----------------------*/
    public static final String SCHEMA_COMPATIBILITY_LEVEL = "X-EBAY-API-COMPATIBILITY-LEVEL";
    public static final String SCHEMA_CALL_NAME = "X-EBAY-API-CALL-NAME";
    public static final String SCHEMA_CERT_NAME = "X-EBAY-API-CERT-NAME";
    public static final String SCHEMA_APP_NAME = "X-EBAY-API-APP-NAME";
    public static final String SCHEMA_DEV_NAME = "X-EBAY-API-DEV-NAME";
    public static final String SCHEMA_DETAIL_LEVEL = "X-EBAY-API-DETAIL-LEVEL";
    public static final String SCHEMA_SITEID = "X-EBAY-API-SITEID";

    public static final String RTM_CALL_NAME = "api";
    public static final String RTM_CALL_VERSION = "ver";

    public static final String SOAP_CALL_NAME = "callname";
    public static final String SOAP_SITEID = "siteid";
    public static final String SOAP_APP_NAME = "appid";
    public static final String SOAP_VERSION = "version";
    public static final String SOAP_ROUTING = "routing";
    public static final String SOAP_CLIENT = "client";
    public static final String SOAP_ACTION = "SOAPAction";

    public static final String CS_CALL_NAME = "callname";
    public static final String CS_SITEID = "siteid";
    public static final String CS_APP_NAME = "appid";
    public static final String CS_VERSION = "version";
    public static final String CS_ROUTING = "routing";
    public static final String CS_CLIENT = "client";
    public static final String CS_DEV_NAME = "devname";
    public static final String CS_CERT_NAME = "certname";
    public static final String CS_SOAP_ACTION = "SOAPAction";
```

```java
    public static final String CS_DEV_NAME = "devname";
    public static final String CS_CERT_NAME = "certname";
    public static final String CS_SOAP_ACTION = "SOAPAction";

    public static final String API_IS_INTERNAL_CLIENT = "X-EBAY-API-IS-INTERNAL-CLIENT";
    public static final String SOA_SEND_RLOGID = "X-EBAY-SOA-SEND-RLOGID";
    /*-----------API EndPoing----------------- */
    private String schemaEndPoint;
    private String soapEndPoint;

    private String csEndPoint;
    private String cs1EndPoint;

    private String rtmEndPoint;
    /*-----------Schema Header------------------ */
    private String schemaCompatibilityLevel;
    private String schemaDetailLevel;
    private String schemaCertName;
    private String schemaAppName;
    private String schemaDevName;

    /*-----------RTM Header----------------- */
    private String rtmVersion;
    /*-----------SOAP Header---------------- */
    private String soapAppName;

    private String soapVersion;

    private String soapRouting;
    private String soapClient;
    /*-----------cs Header---------------- */
    private String csAppName;
    private String csVersion;
    private String csRouting;
    private String csClient;
    private String csDevName;
    private String csCertName;
    /*-----------RLogId Header------------- */
    private String apiIsInternalClient;
    private String soaSendRlogid;

    public String getCsAppName() {
        return csAppName;
    }
    public String getCsCertName() {
        return csCertName;
    }

    public String getCsClient() {
        return csClient;
    }

    public String getCsDevName() {
        return csDevName;
    }

    public String getCsEndPoint() {
        return csEndPoint;
    }

    public String getCs1EndPoint() {
        return cs1EndPoint;
    }

    public String getCsRouting() {
        return csRouting;
    }
```

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | APIContext.class

```java
    public String getCsVersion() {
        return csVersion;
    }

    public String getRtmEndPoint() {
        return rtmEndPoint;
    }

    public String getRtmVersion() {
        return rtmVersion;
    }

    public String getSchemaAppName() {
        return schemaAppName;
    }

    public String getSchemaCertName() {
        return schemaCertName;
    }

    public String getSchemaCompatibilityLevel() {
        return schemaCompatibilityLevel;
    }

    public String getSchemaDetailLevel() {
        return schemaDetailLevel;
    }

    public String getSchemaDevName() {
        return schemaDevName;
    }

    public String getSchemaEndPoint() {
        return schemaEndPoint;
    }

    public String getSoapAppName() {
        return soapAppName;
    }

    public String getSoapClient() {
        return soapClient;
    }

    public String getSoapEndPoint() {
        return soapEndPoint;
    }

    public String getSoapRouting() {
        return soapRouting;
    }

    public String getSoapVersion() {
        return soapVersion;
    }

    public void load(ITestContext testNGCtx, Context mauiCtx) {
        String schemaEndPoint = "http://eazye.qa.ebay.com:8080/ws/api.dll";
        if (System.getProperty(APIContext.API_SCHEMA_END_POINT.toUpperCase()) != null) {
            schemaEndPoint = System.getProperty(APIContext.API_SCHEMA_END_POINT.toUpperCase());
        } else if (testNGCtx!=null && testNGCtx.getSuite().getParameter(APIContext.API_SCHEMA_END_POINT) != null) {
            schemaEndPoint = testNGCtx.getSuite().getParameter(APIContext.API_SCHEMA_END_POINT);
        }

        String soapEndPoint = "http://eazye.qa.ebay.com:8080/wsapi";
        if (System.getProperty(APIContext.API_SOAP_END_POINT.toUpperCase()) != null) {
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | APIContext.class

```java
        if (System.getProperty(APIContext.API_SOAP_END_POINT.toUpperCase()) != null) {
            soapEndPoint = System.getProperty(APIContext.API_SOAP_END_POINT.toUpperCase());
        } else if (testNGCtx!=null && testNGCtx.getSuite().getParameter(APIContext.API_SOAP_END_POINT) != null) {
            soapEndPoint = testNGCtx.getSuite().getParameter(APIContext.API_SOAP_END_POINT);
        }

        //String csEndPoint = "http://tv3apicore.qa.ebay.com:8080/ws/websvc/eBayCSAPI";
        String csEndPoint = "http://apics.vip.qa.ebay.com/ws/websvc/eBayCSAPI";
        if (System.getProperty(APIContext.API_CS_END_POINT.toUpperCase()) != null) {
            csEndPoint = System.getProperty(APIContext.API_CS_END_POINT.toUpperCase());
        } else if (testNGCtx!=null && testNGCtx.getSuite().getParameter(APIContext.API_CS_END_POINT) != null) {
            csEndPoint = testNGCtx.getSuite().getParameter(APIContext.API_CS_END_POINT);
        }

        String cs1EndPoint = "http://apics1.qa.ebay.com/ws/websvc/eBayCSAPI";
        if (System.getProperty(APIContext.API_CS1_END_POINT.toUpperCase()) != null) {
            cs1EndPoint = System.getProperty(APIContext.API_CS1_END_POINT.toUpperCase());
        } else if (testNGCtx!=null && testNGCtx.getSuite().getParameter(APIContext.API_CS1_END_POINT) != null) {
            cs1EndPoint = testNGCtx.getSuite().getParameter(APIContext.API_CS1_END_POINT);
        }

        String rtmEndPoint = "http://srv.main.qa.ebayrtm.com/rtm?RtmAdminApi";
        if (System.getProperty(APIContext.API_RTM_END_POINT.toUpperCase()) != null) {
            rtmEndPoint = System.getProperty(APIContext.API_RTM_END_POINT.toUpperCase());
        } else if (testNGCtx!=null && testNGCtx.getSuite().getParameter(APIContext.API_RTM_END_POINT) != null) {
            rtmEndPoint = testNGCtx.getSuite().getParameter(APIContext.API_RTM_END_POINT);
        }

        String apiVersion = "715";
        if (System.getProperty(APIContext.API_VERSION.toUpperCase()) != null) {
            apiVersion = System.getProperty(APIContext.API_VERSION.toUpperCase());
        } else if (testNGCtx!=null && testNGCtx.getSuite().getParameter(APIContext.API_VERSION) != null) {
            apiVersion = testNGCtx.getSuite().getParameter(APIContext.API_VERSION);
        }

        String certName = "AdminCertificate";
        if (System.getProperty(APIContext.API_CERT_NAME.toUpperCase()) != null) {
            certName = System.getProperty(APIContext.API_CERT_NAME.toUpperCase());
        } else if (testNGCtx!=null && testNGCtx.getSuite().getParameter(APIContext.API_CERT_NAME) != null) {
            certName = testNGCtx.getSuite().getParameter(APIContext.API_CERT_NAME);
        }

        String appName = "AdminApp";
        if (System.getProperty(APIContext.API_APP_NAME.toUpperCase()) != null) {
            appName = System.getProperty(APIContext.API_APP_NAME.toUpperCase());
        } else if (testNGCtx!=null && testNGCtx.getSuite().getParameter(APIContext.API_APP_NAME) != null) {
            appName = testNGCtx.getSuite().getParameter(APIContext.API_APP_NAME);
        }

        String devName = "Admintest1";
        if (System.getProperty(APIContext.API_DEV_NAME.toUpperCase()) != null) {
            devName = System.getProperty(APIContext.API_DEV_NAME.toUpperCase());
        } else if (testNGCtx!=null && testNGCtx.getSuite().getParameter(APIContext.API_DEV_NAME) != null) {
            devName = testNGCtx.getSuite().getParameter(APIContext.API_DEV_NAME);
        }

        String detailLevel = "4";
        if (System.getProperty(APIContext.API_DETAIL_LEVEL.toUpperCase()) != null) {
            detailLevel = System.getProperty(APIContext.API_DETAIL_LEVEL.toUpperCase());
        } else if (testNGCtx!=null && testNGCtx.getSuite().getParameter(APIContext.API_DETAIL_LEVEL) != null) {
            detailLevel = testNGCtx.getSuite().getParameter(APIContext.API_DETAIL_LEVEL);
        }

        String csCertName = "sttiCertificate";
        if (System.getProperty(APIContext.API_CS_CERT_NAME.toUpperCase()) != null) {
            csCertName = System.getProperty(APIContext.API_CS_CERT_NAME.toUpperCase());
        } else if (testNGCtx!=null && testNGCtx.getSuite().getParameter(APIContext.API_CS_CERT_NAME) != null) {
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | APIContext.class

```java
        } else if (testNGCtx!=null && testNGCtx.getSuite().getParameter(APIContext.API_CS_CERT_NAME) != null) {
            csCertName = testNGCtx.getSuite().getParameter(APIContext.API_CS_CERT_NAME);
        }

        String csAppName = "sttiApplication1";
        if (System.getProperty(APIContext.API_CS_APP_NAME.toUpperCase()) != null) {
            csAppName = System.getProperty(APIContext.API_CS_APP_NAME.toUpperCase());
        } else if (testNGCtx!=null && testNGCtx.getSuite().getParameter(APIContext.API_CS_APP_NAME) != null) {
            csAppName = testNGCtx.getSuite().getParameter(APIContext.API_CS_APP_NAME);
        }

        String csDevName = "sttiDeveloper";
        if (System.getProperty(APIContext.API_CS_DEV_NAME.toUpperCase()) != null) {
            csDevName = System.getProperty(APIContext.API_CS_DEV_NAME.toUpperCase());
        } else if (testNGCtx!=null && testNGCtx.getSuite().getParameter(APIContext.API_CS_DEV_NAME) != null) {
            csDevName = testNGCtx.getSuite().getParameter(APIContext.API_CS_DEV_NAME);
        }

        APIContext config = new APIContext();
        config.setSchemaEndPoint(schemaEndPoint);
        config.setSchemaCompatibilityLevel(apiVersion);
        config.setSchemaCertName(certName);
        config.setSchemaAppName(appName);
        config.setSchemaDevName(devName);
        config.setSchemaDetailLevel(detailLevel);

        config.setRtmEndPoint(rtmEndPoint);
        config.setRtmVersion("0");

        config.setSoapEndPoint(soapEndPoint);
        config.setSoapVersion(apiVersion);
        config.setSoapAppName(appName);
        config.setSoapRouting("new");
        config.setSoapClient("net");

        config.setCsEndPoint(csEndPoint);
        config.setCs1EndPoint(cs1EndPoint);
        config.setCsVersion(apiVersion);
        config.setCsAppName(csAppName);
        config.setCsDevName(csDevName);
        config.setCsCertName(csCertName);
        config.setCsRouting("new");
        config.setCsClient("net");

        config.setApiIsInternalClient("true");
        config.setSoapSendRlogid("1");

        if(mauiCtx!=null)mauiCtx.setAttribute(Context.API_CONTEXT, config);
    }

    public void setCsAppName(String csAppName) {
        this.csAppName = csAppName;
    }

    public void setCsCertName(String csCertName) {
        this.csCertName = csCertName;
    }

    public void setCsClient(String csClient) {
        this.csClient = csClient;
    }

    public void setCsDevName(String csDevName) {
        this.csDevName = csDevName;
    }

    public void setCsEndPoint(String csEndPoint) {
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | APIContext.class

```java
        this.csEndPoint = csEndPoint;
    }

    public void setCs1EndPoint(String cs1EndPoint) {
        this.cs1EndPoint = cs1EndPoint;
    }
    public void setCsRouting(String csRouting) {
        this.csRouting = csRouting;
    }

    public void setCsVersion(String csVersion) {
        this.csVersion = csVersion;
    }

    public void setRtmEndPoint(String rtmEndPoint) {
        this.rtmEndPoint = rtmEndPoint;
    }

    public void setRtmVersion(String rtmVersion) {
        this.rtmVersion = rtmVersion;
    }

    public void setSchemaAppName(String schemaAppName) {
        this.schemaAppName = schemaAppName;
    }

    public void setSchemaCertName(String schemaCertName) {
        this.schemaCertName = schemaCertName;
    }

    public void setSchemaCompatibilityLevel(String schemaCompatibilityLevel) {
        this.schemaCompatibilityLevel = schemaCompatibilityLevel;
    }

    public void setSchemaDetailLevel(String schemaDetailLevel) {
        this.schemaDetailLevel = schemaDetailLevel;
    }

    public void setSchemaDevName(String schemaDevName) {
        this.schemaDevName = schemaDevName;
    }

    public void setSchemaEndPoint(String schemaEndPoint) {
        this.schemaEndPoint = schemaEndPoint;
    }

    public void setSoapAppName(String soapAppName) {
        this.soapAppName = soapAppName;
    }

    public void setSoapClient(String soapClient) {
        this.soapClient = soapClient;
    }

    public void setSoapEndPoint(String soapEndPoint) {
        this.soapEndPoint = soapEndPoint;
    }

    public void setSoapRouting(String soapRouting) {
        this.soapRouting = soapRouting;
    }

    public void setSoapVersion(String soapVersion) {
        this.soapVersion = soapVersion;
    }
    public void setApiIsInternalClient(String apiIsInternalClient) {
        this.apiIsInternalClient = apiIsInternalClient;
```

Read-Only | Smart Insert | 1 : 1

RESTResponse.class | SiteUtil.class | SpreadSheetUtil.class | StringInputStream.class | SSHUtil.class | WebUXUtil.class | APIContext.class

```java
    }

    public void setRtmEndPoint(String rtmEndPoint) {
        this.rtmEndPoint = rtmEndPoint;
    }

    public void setRtmVersion(String rtmVersion) {
        this.rtmVersion = rtmVersion;
    }

    public void setSchemaAppName(String schemaAppName) {
        this.schemaAppName = schemaAppName;
    }

    public void setSchemaCertName(String schemaCertName) {
        this.schemaCertName = schemaCertName;
    }

    public void setSchemaCompatibilityLevel(String schemaCompatibilityLevel) {
        this.schemaCompatibilityLevel = schemaCompatibilityLevel;
    }

    public void setSchemaDetailLevel(String schemaDetailLevel) {
        this.schemaDetailLevel = schemaDetailLevel;
    }

    public void setSchemaDevName(String schemaDevName) {
        this.schemaDevName = schemaDevName;
    }

    public void setSchemaEndPoint(String schemaEndPoint) {
        this.schemaEndPoint = schemaEndPoint;
    }

    public void setSoapAppName(String soapAppName) {
        this.soapAppName = soapAppName;
    }

    public void setSoapClient(String soapClient) {
        this.soapClient = soapClient;
    }

    public void setSoapEndPoint(String soapEndPoint) {
        this.soapEndPoint = soapEndPoint;
    }

    public void setSoapRouting(String soapRouting) {
        this.soapRouting = soapRouting;
    }

    public void setSoapVersion(String soapVersion) {
        this.soapVersion = soapVersion;
    }
    public void setApiIsInternalClient(String apiIsInternalClient) {
        this.apiIsInternalClient = apiIsInternalClient;
    }
    public String getApiIsInternalClient() {
        return apiIsInternalClient;
    }
    public void setSoaSendRlogid(String soaSendRlogid) {
        this.soaSendRlogid = soaSendRlogid;
    }
    public String getSoaSendRlogid() {
        return soaSendRlogid;
    }
}
```

Read-Only | Smart Insert | 1 : 1

Projec | Navig | Type

- itemtemplates.csv
- com.ebay.maui.util.internal.bes
- com.ebay.maui.util.internal.bes.consum
- com.ebay.maui.util.internal.bes.event
- com.ebay.maui.util.internal.cal
- com.ebay.maui.util.internal.context
  - APIContext.class
    - APIContext
  - APIContextListener.class
- com.ebay.maui.util.internal.db
- com.ebay.maui.util.internal.entity
- com.ebay.maui.util.internal.log
- com.ebay.maui.util.internal.pc
- com.ebay.maui.util.internal.site
- com.ebay.maui.util.internal.type
- com.ebay.maui.util.internal.vi
- com.ebay.maui.util.internal.web
  - Fees.class
  - IWaitable.class
  - LinksChecker.class
  - MouseActionUtility.class
  - ProductHelper.class
  - SessionUtils.class
  - SmartRobot.class
  - URLConverter.class
  - URLConverterV1.class
  - WebElementWaiter.class
- com.ebay.maui.util.jserror
- com.ebay.maui.util.paymente2e
- com.ebay.maui.util.proxy
- META-INF
- tmp
- mauiplatform-1.1.6.jar - /Users/gaurshukla
- log4j-1.2.17.jar - /Users/gaurshukla/.m2/r
- velocity-1.7.jar - /Users/gaurshukla/.m2/ra
- sequalite-1.0.6.jar - /Users/gaurshukla/.m2
- mail-1.4.5.jar - /Users/gaurshukla/.m2/rap
- activation-1.1.jar - /Users/gaurshukla/.m2/
- jsch-0.1.49.jar - /Users/gaurshukla/.m2/ra
- selenium-java-2.41.0.jar - /Users/gaurshui
- selenium-chrome-driver-2.41.0.jar - /User
- selenium-remote-driver-2.41.0.jar - /Users
- cglib-nodep-2.1_3.jar - /Users/gaurshukla/
- selenium-htmlunit-driver-2.41.0.jar - /Use
- htmlunit-2.13.jar - /Users/gaurshukla/.m2/
- httpmime-4.3.1.jar - /Users/gaurshukla/.m

WebUXUtil.class | IWaitable.class | LinksChecker.class | MouseActionUtility.class

```java
package com.ebay.maui.util.internal.web;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

/**
 * This class contains methods to perform mouseactions on a webpage using
 * webdriver.
 *
 * @author arkannan@ebay.com (Aravind Kannan)
 * @author sumitkumar@ebay.com (Sumit Kumar)
 */
public class MouseActionUtility {

    /**
     * Hover mouse over the WebElement identified by 'hoverOnElement' and polls
     * for 'locatorToFindOnPageAfterHovering' and logs the event.
     *
     * @param hoverOnElement
     * @param locatorToFindOnPageAfterHovering
     * @param driver
     */
    public static void mouseOver(WebElement hoverOnElement,
            final WebDriver driver) {
        doMouseActionUsingJS("mouseover", hoverOnElement, driver);
    }

    public static void doMouseActionUsingJS(String mouseEvent, By hoverOn,
            WebDriver webDriver) {
        WebElement element = webDriver.findElement(hoverOn);
        ((JavascriptExecutor) webDriver).executeScript(
                "var event = document.createEvent('MouseEvents');"
                        + "event.initEvent('" + mouseEvent + "', true, true);"
                        + "var element = arguments[0];"
                        + "element.dispatchEvent(event);", element);
    }

    public static void doMouseActionUsingJS(String mouseEvent,
            WebElement hoverOnElement, WebDriver webDriver) {
        ((JavascriptExecutor) webDriver).executeScript(
                "var event = document.createEvent('MouseEvents');"
                        + "event.initEvent('" + mouseEvent + "', true, true);"
                        + "var element = arguments[0];"
                        + "element.dispatchEvent(event);", hoverOnElement);
    }
}
```

Outline | Task Li

- MouseActionUtility
  - MouseActionUtility()
  - mouseOver(WebElement, We
  - doMouseActionUsingJS(Stri
  - doMouseActionUsingJS(Stri

Markers | Properties | Servers | Data Source Explorer | Snippets | Console | Progress | Results of running class RaiseClaimTestPlan | Package Explorer | Call Hierarchy

No operations to display at this time.

Read-Only | Smart Insert | 1 : 1

```java
package com.ebay.maui.util.internal.web;

import java.awt.AWTException;
import java.awt.Robot;
import java.awt.Toolkit;
import java.awt.datatransfer.Clipboard;
import java.awt.datatransfer.StringSelection;
import java.awt.datatransfer.Transferable;
import java.awt.event.KeyEvent;


class SmartRobot extends Robot
{

public SmartRobot() throws AWTException
{
super();
}

public void pasteClipboard()
{
keyPress(KeyEvent.VK_CONTROL);
keyPress(KeyEvent.VK_V);
delay(50);
keyRelease(KeyEvent.VK_V);
keyRelease(KeyEvent.VK_CONTROL);
}

public void type(String text)
{
writeToClipboard(text);
pasteClipboard();
}
public void keyType(int keyCode) {
      keyPress(keyCode);
      delay(50);
      keyRelease(keyCode);
     }

private void writeToClipboard(String s)
{
Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();
Transferable transferable = new StringSelection(s);
clipboard.setContents(transferable, null);
}
}
```

```java
package com.ebay.maui.util.internal.web;

import com.ebay.maui.controller.Logging;
import com.ebay.maui.driver.web.element.HtmlElement;
import com.ebay.maui.driver.web.element.WebPageSection;
import com.thoughtworks.selenium.Wait;
import com.thoughtworks.selenium.Wait.WaitTimedOutException;

/**
 * This class handles the waiting behaviors with the events definable by users in {@link IWaitable#meetIt(Object)}, capturing screenshots when the waitings fail.
 *
 * @author Michael Wu (Ziyang.Wu@ebay.com)
 *
 */
public class WebElementWaiter extends WebPageSection {
    private String defaultElementDisplayName = "Unnamed Element";

    public WebElementWaiter() {
        super("Web Element Waiter");
    }

    public WebElementWaiter(String defaultElementDisplayName) {
        this();
        this.defaultElementDisplayName = defaultElementDisplayName;
    }

    /**
     * To get the default element display name of the waited element, if the name is not set for it.
     *
     * @return default element display name
     */
    public String getDefaultElementDisplayName() {
        return defaultElementDisplayName;
    }

    /**
     * <pre>
     * To find if the instance of HtmlElement is displayed in given time period, once it's displayed, the method will end the waiting to save time.
     *
     * This method will take the label field of the given element as the element display name to show in report, if the label is set null or "",
     * {@link #getDefaultElementDisplayName()} would be used.
     * </pre>
     *
     * @param element the waited instance of HtmlElement
     * @param timeoutInSeconds longest waiting time in seconds
     * @return is the element displayed
     */
    public boolean isElementDisplayed(HtmlElement element, int timeoutInSeconds) {
        String elementDisplayName = element.getLabel();
        if (elementDisplayName == null || "".equals(elementDisplayName))
            elementDisplayName = getDefaultElementDisplayName();
        return isElementDisplayed(element, elementDisplayName, timeoutInSeconds);
    }

    /**
     * To find if the instance of HtmlElement is displayed in given time period, once it's displayed, the method will end the waiting to save time.
     *
     * @param element the waited instance of HtmlElement
     * @param elementDisplayName the element display name in report
     * @param timeoutInSeconds longest waiting time in seconds
     * @return is the element displayed
     */
    public boolean isElementDisplayed(HtmlElement element, String elementDisplayName, int timeoutInSeconds) {
        return isElementDisplayed(element, elementDisplayName, timeoutInSeconds, new IWaitable<HtmlElement>(){
            public boolean meetIt(HtmlElement e) {
                return e.isDisplayed();
            }
```

WebUXUtil.class | IWaitable.class | LinksChecker.class | MouseActionUtility.class | SmartRobot.class | URLConverter.class | WebElementWaiter.class

```
72      * To find if the instance of HtmlElement is displayed in given time period, once it's displayed, the method will end the waiting to save time.
73      *
74      * @param element the waited instance of HtmlElement
75      * @param elementDisplayName the element display name in report
76      * @param timeoutInSeconds longest waiting time in seconds
77      * @param waitable the instance implementing {@link IWaitable} interface to define what event to wait for
78      * @return is the element displayed
79      */
80     public boolean isElementDisplayed(final HtmlElement element, String elementDisplayName, int timeoutInSeconds, final IWaitable<HtmlElement> waitable) {
81         final int validatedTimeoutInSeconds = timeoutInSeconds <= 0 ? timeoutInSeconds * -1 : timeoutInSeconds;
82         Logging.info(String.format("Waiting at longest %s seconds for %s to appear...", timeoutInSeconds, elementDisplayName));
83         try {
84             new Wait() {
85                 public boolean until() {
86                     return waitable.meetIt(element);
87                 }
88             }.wait(String.format("The %s is not shown in %s seconds.", elementDisplayName, validatedTimeoutInSeconds), validatedTimeoutInSeconds * 1000, 1000);
89             return true;
90         }
91         catch (WaitTimedOutException e) {
92             capturePageSnapshot();
93             return false;
94         }
95     }
96
97     /**
98      * <pre>
99      * To find if the instance of HtmlElement is hidden in given time period, once it's hidden, the method will end the waiting to save time.
100     *
101     * This method will take the label field of the given element as the element display name to show in report, if the label is set null or "",
102     * {@link #getDefaultElementDisplayName()} would be used.
103     * </pre>
104     *
105     * @param element the waited instance of HtmlElement
106     * @param timeoutInSeconds longest waiting time in seconds
107     * @return is the element hidden
108     */
109    public boolean isElementHidden(HtmlElement element, int timeoutInSeconds) {
110        String elementDisplayName = element.getLabel();
111        if (elementDisplayName == null || "".equals(elementDisplayName))
112            elementDisplayName = getDefaultElementDisplayName();
113        return isElementHidden(element, elementDisplayName, timeoutInSeconds);
114    }
115
116    /**
117     * To find if the instance of HtmlElement is hidden in given time period, once it's hidden, the method will end the waiting to save time.
118     *
119     * @param element the waited instance of HtmlElement
120     * @param elementDisplayName the element display name in report
121     * @param timeoutInSeconds longest waiting time in seconds
122     * @return is the element hidden
123     */
124    public boolean isElementHidden(HtmlElement element, String elementDisplayName, int timeoutInSeconds) {
125        return isElementHidden(element, elementDisplayName, timeoutInSeconds, new IWaitable<HtmlElement>(){
126            public boolean meetIt(HtmlElement e) {
127                return !e.isDisplayed();
128            }
129        });
130    }
131
132    /**
133     * To find if the instance of HtmlElement is hidden in given time period, once it's hidden, the method will end the waiting to save time.
134     *
135     * @param element the waited instance of HtmlElement
136     * @param elementDisplayName the element display name in report
137     * @param timeoutInSeconds longest waiting time in seconds
138     * @param waitable the instance implementing {@link IWaitable} interface to define what event to wait for
```

WebUXUtil.class | IWaitable.class | LinksChecker.class | MouseActionUtility.class | SmartRobot.class | URLConverter.class | WebElementWaiter.class

```
138     * @param waitable the instance implementing {@link IWaitable} interface to define what event to wait for
139     * @return is the element hidden
140     */
141    public boolean isElementHidden(final HtmlElement element, String elementDisplayName, int timeoutInSeconds, final IWaitable<HtmlElement> waitable) {
142        final int validatedTimeoutInSeconds = timeoutInSeconds <= 0 ? timeoutInSeconds * -1 : timeoutInSeconds;
143        Logging.info(String.format("Waiting at longest %s seconds for %s to disappear...", timeoutInSeconds, elementDisplayName));
144
145        try {
146            new Wait() {
147                public boolean until() {
148                    return waitable.meetIt(element);
149                }
150            }.wait(String.format("The %s is still shown after %s seconds.", elementDisplayName, validatedTimeoutInSeconds),
151                    validatedTimeoutInSeconds * 1000,
152                    1000);
153            return true;
154        }
155        catch (RuntimeException e) {
156            capturePageSnapshot();
157            return false;
158        }
159    }
160
161    /**
162     * <pre>
163     * Wait until the the instance of HtmlElement to be displayed in given time period, once it's displayed, the method will end the waiting to save time.
164     *
165     * This method will take the label field of the given element as the element display name to show in report, if the label is set null or "",
166     * {@link #getDefaultElementDisplayName()} would be used.
167     * </pre>
168     *
169     * @param element the waited instance of HtmlElement
170     * @param timeoutInSeconds longest waiting time in seconds
171     */
172    public void waitUntilElementToBeDisplayed(HtmlElement element, int timeoutInSeconds) {
173        waitUntilElementToBeDisplayed(element, timeoutInSeconds, 1);
174    }
175
176    /**
177     * <pre>
178     * Wait until the the instance of HtmlElement to be displayed in given time period, once it's displayed, the method will end the waiting to save time.
179     *
180     * This method will take the label field of the given element as the element display name to show in report, if the label is set null or "",
181     * {@link #getDefaultElementDisplayName()} would be used.
182     * </pre>
183     *
184     * @param element the waited instance of HtmlElement
185     * @param timeoutInSeconds longest waiting time in seconds
186     * @param intervalInSeconds the interval seconds between each checking loop
187     */
188    public void waitUntilElementToBeDisplayed(HtmlElement element, int timeoutInSeconds, int intervalInSeconds) {
189        String elementDisplayName = element.getLabel();
190        if (elementDisplayName == null || "".equals(elementDisplayName))
191            elementDisplayName = getDefaultElementDisplayName();
192        waitUntilElementToBeDisplayed(element, elementDisplayName, timeoutInSeconds, intervalInSeconds);
193    }
194
195    /**
196     * Wait until the the instance of HtmlElement to be displayed in given time period, once it's displayed, the method will end the waiting to save time.
197     *
198     * @param element the waited instance of HtmlElement
199     * @param elementDisplayName the element display name in report
200     * @param timeoutInSeconds longest waiting time in seconds
201     */
202    public void waitUntilElementToBeDisplayed(HtmlElement element, String elementDisplayName, int timeoutInSeconds) {
203        waitUntilElementToBeDisplayed(element, elementDisplayName, timeoutInSeconds, 1);
204    }
```

WebUXUtil.class | IWaitable.class | LinksChecker.class | MouseActionUtility.class | SmartRobot.class | URLConverter.class | WebElementWaiter.class

```java
    /**
     * Wait until the the instance of HtmlElement to be displayed in given time period, once it's displayed, the method will end the waiting to save time.
     *
     * @param element the waited instance of HtmlElement
     * @param elementDisplayName the element display name in report
     * @param timeoutInSeconds longest waiting time in seconds
     * @param intervalInSeconds the interval seconds between each checking loop
     */
    public void waitUntilElementToBeDisplayed(HtmlElement element, String elementDisplayName, int timeoutInSeconds, int intervalInSeconds) {
        waitUntilElementToBeDisplayed(element, elementDisplayName, timeoutInSeconds, intervalInSeconds, new IWaitable<HtmlElement>(){
            public boolean meetIt(HtmlElement e) {
                return e.isDisplayed();
            }
        });
    }

    /**
     * Wait until the the instance of HtmlElement to be displayed in given time period, once it's displayed, the method will end the waiting to save time.
     *
     * @param element the waited instance of HtmlElement
     * @param elementDisplayName the element display name in report
     * @param timeoutInSeconds longest waiting time in seconds
     * @param intervalInSeconds the interval seconds between each checking loop
     * @param waitable the instance implementing {@link IWaitable} interface to define what event to wait for
     */
    public void waitUntilElementToBeDisplayed(final HtmlElement element,
                                              String elementDisplayName,
                                              int timeoutInSeconds,
                                              int intervalInSeconds,
                                              final IWaitable<HtmlElement> waitable) {
        final int validatedTimeoutInSeconds = timeoutInSeconds <= 0 ? timeoutInSeconds * -1 : timeoutInSeconds;
        final int validatedIntervalInSeconds = intervalInSeconds <= 0 ? (intervalInSeconds == 0 ? 1 : intervalInSeconds * -1) : intervalInSeconds;
        Logging.info(String.format("Waiting at longest %s seconds for %s to be displayed...",
                                   timeoutInSeconds,
                                   elementDisplayName));

        try {
            new Wait() {
                public boolean until() {
                    return waitable.meetIt(element);
                }
            }.wait(String.format("The %s is not displayed in %s seconds.", elementDisplayName, validatedTimeoutInSeconds),
                   validatedTimeoutInSeconds * 1000,
                   validatedIntervalInSeconds * 1000);
        }
        catch (RuntimeException e) {
            capturePageSnapshot();
            throw e;
        }
    }

    /**
     * <pre>
     * Wait until the the instance of HtmlElement to be absent in given time period, once it's absent, the method will end the waiting to save time.
     *
     * This method will take the label field of the given element as the element display name to show in report, if the label is set null or "",
     * {@link #getDefaultElementDisplayName()} would be used.
     * </pre>
     *
     * @param element the waited instance of HtmlElement
     * @param timeoutInSeconds longest waiting time in seconds
     */
    public void waitUntilElementToBeAbsent(HtmlElement element, int timeoutInSeconds) {
        waitUntilElementToBeAbsent(element, timeoutInSeconds, 1);
    }

    /**
     * <pre>
```

WebUXUtil.class | IWaitable.class | LinksChecker.class | MouseActionUtility.class | SmartRobot.class | URLConverter.class | WebElementWaiter.class

```java
     * <pre>
     * Wait until the the instance of HtmlElement to be absent in given time period, once it's absent, the method will end the waiting to save time.
     *
     * This method will take the label field of the given element as the element display name to show in report, if the label is set null or "",
     * {@link #getDefaultElementDisplayName()} would be used.
     * </pre>
     *
     * @param element the waited instance of HtmlElement
     * @param timeoutInSeconds longest waiting time in seconds
     * @param intervalInSeconds the interval seconds between each checking loop
     */
    public void waitUntilElementToBeAbsent(HtmlElement element, int timeoutInSeconds, int intervalInSeconds) {
        String elementDisplayName = element.getLabel();
        if (elementDisplayName == null || "".equals(elementDisplayName))
            elementDisplayName = getDefaultElementDisplayName();
        waitUntilElementToBeAbsent(element, elementDisplayName, timeoutInSeconds, intervalInSeconds);
    }

    /**
     * Wait until the the instance of HtmlElement to be absent in given time period, once it's absent, the method will end the waiting to save time.
     *
     * @param element the waited instance of HtmlElement
     * @param elementDisplayName the element display name in report
     * @param timeoutInSeconds longest waiting time in seconds
     */
    public void waitUntilElementToBeAbsent(HtmlElement element, String elementDisplayName, int timeoutInSeconds) {
        waitUntilElementToBeAbsent(element, elementDisplayName, timeoutInSeconds, 1);
    }

    /**
     * Wait until the the instance of HtmlElement to be absent in given time period, once it's absent, the method will end the waiting to save time.
     *
     * @param element the waited instance of HtmlElement
     * @param elementDisplayName the element display name in report
     * @param timeoutInSeconds longest waiting time in seconds
     * @param intervalInSeconds the interval seconds between each checking loop
     */
    public void waitUntilElementToBeAbsent(HtmlElement element, String elementDisplayName, int timeoutInSeconds, int intervalInSeconds) {
        waitUntilElementToBeAbsent(element, elementDisplayName, timeoutInSeconds, intervalInSeconds, new IWaitable<HtmlElement>(){
            public boolean meetIt(HtmlElement e) {
                return !e.isElementPresent();
            }
        });
    }

    /**
     * Wait until the the instance of HtmlElement to be absent in given time period, once it's absent, the method will end the waiting to save time.
     *
     * @param element the waited instance of HtmlElement
     * @param elementDisplayName the element display name in report
     * @param timeoutInSeconds longest waiting time in seconds
     * @param intervalInSeconds the interval seconds between each checking loop
     * @param waitable the instance implementing {@link IWaitable} interface to define what event to wait for
     */
    public void waitUntilElementToBeAbsent(final HtmlElement element,
                                           String elementDisplayName,
                                           int timeoutInSeconds,
                                           int intervalInSeconds,
                                           final IWaitable<HtmlElement> waitable) {
        final int validatedTimeoutInSeconds = timeoutInSeconds <= 0 ? timeoutInSeconds * -1 : timeoutInSeconds;
        final int validatedIntervalInSeconds = intervalInSeconds <= 0 ? (intervalInSeconds == 0 ? 1 : intervalInSeconds * -1) : intervalInSeconds;
        Logging.info(String.format("Waiting at longest %s seconds for %s to be absent...", timeoutInSeconds, elementDisplayName));

        try {
            new Wait() {
                public boolean until() {
                    return waitable.meetIt(element);
```

WebUXUtil.class | IWaitable.class | LinksChecker.class | MouseActionUtility.class | SmartRobot.class | URLConverter.class | WebElementWaiter.class

```java
                return waitable.meetIt(element);
            }
        }.wait(String.format("The %s is still present after %s seconds.", elementDisplayName, validatedTimeoutInSeconds),
                validatedTimeoutInSeconds * 1000,
                validatedIntervalInSeconds * 1000);
    }
    catch (RuntimeException e) {
        capturePageSnapshot();
        throw e;
    }
}

/**
 * <pre>
 * Wait until the the instance of HtmlElement to be hidden in given time period, once it's hidden, the method will end the waiting to save time.
 *
 * This method will take the label field of the given element as the element display name to show in report, if the label is set null or "",
 * {@link #getDefaultElementDisplayName()} would be used.
 * </pre>
 *
 * @param element the waited instance of HtmlElement
 * @param timeoutInSeconds longest waiting time in seconds
 */
public void waitUntilElementToBeHidden(HtmlElement element, int timeoutInSeconds) {
    waitUntilElementToBeHidden(element, timeoutInSeconds, 1);
}

/**
 * <pre>
 * Wait until the the instance of HtmlElement to be hidden in given time period, once it's hidden, the method will end the waiting to save time.
 *
 * This method will take the label field of the given element as the element display name to show in report, if the label is set null or "",
 * {@link #getDefaultElementDisplayName()} would be used.
 * </pre>
 *
 * @param element the waited instance of HtmlElement
 * @param timeoutInSeconds longest waiting time in seconds
 * @param intervalInSeconds the interval seconds between each checking loop
 */
public void waitUntilElementToBeHidden(HtmlElement element, int timeoutInSeconds, int intervalInSeconds) {
    String elementDisplayName = element.getLabel();
    if (elementDisplayName == null || "".equals(elementDisplayName))
        elementDisplayName = getDefaultElementDisplayName();
    waitUntilElementToBeHidden(element, elementDisplayName, timeoutInSeconds, intervalInSeconds);
}

/**
 * Wait until the the instance of HtmlElement to be hidden in given time period, once it's hidden, the method will end the waiting to save time.
 *
 * @param element the waited instance of HtmlElement
 * @param elementDisplayName the element display name in report
 * @param timeoutInSeconds longest waiting time in seconds
 */
public void waitUntilElementToBeHidden(HtmlElement element, String elementDisplayName, int timeoutInSeconds) {
    waitUntilElementToBeHidden(element, elementDisplayName, timeoutInSeconds, 1);
}

/**
 * Wait until the the instance of HtmlElement to be hidden in given time period, once it's hidden, the method will end the waiting to save time.
 *
 * @param element the waited instance of HtmlElement
 * @param elementDisplayName the element display name in report
 * @param timeoutInSeconds longest waiting time in seconds
 * @param intervalInSeconds the interval seconds between each checking loop
 */
public void waitUntilElementToBeHidden(HtmlElement element, String elementDisplayName, int timeoutInSeconds, int intervalInSeconds) {
    waitUntilElementToBeHidden(element, elementDisplayName, timeoutInSeconds, intervalInSeconds, new IWaitable<HtmlElement>(){
```

Read-Only | Smart Insert | 1 : 1

WebUXUtil.class | IWaitable.class | LinksChecker.class | MouseActionUtility.class | SmartRobot.class | URLConverter.class | WebElementWaiter.class

```java
public void waitUntilElementToBeHidden(HtmlElement element, int timeoutInSeconds, int intervalInSeconds) {
    String elementDisplayName = element.getLabel();
    if (elementDisplayName == null || "".equals(elementDisplayName))
        elementDisplayName = getDefaultElementDisplayName();
    waitUntilElementToBeHidden(element, elementDisplayName, timeoutInSeconds, intervalInSeconds);
}

/**
 * Wait until the the instance of HtmlElement to be hidden in given time period, once it's hidden, the method will end the waiting to save time.
 *
 * @param element the waited instance of HtmlElement
 * @param elementDisplayName the element display name in report
 * @param timeoutInSeconds longest waiting time in seconds
 */
public void waitUntilElementToBeHidden(HtmlElement element, String elementDisplayName, int timeoutInSeconds) {
    waitUntilElementToBeHidden(element, elementDisplayName, timeoutInSeconds, 1);
}

/**
 * Wait until the the instance of HtmlElement to be hidden in given time period, once it's hidden, the method will end the waiting to save time.
 *
 * @param element the waited instance of HtmlElement
 * @param elementDisplayName the element display name in report
 * @param timeoutInSeconds longest waiting time in seconds
 * @param intervalInSeconds the interval seconds between each checking loop
 */
public void waitUntilElementToBeHidden(HtmlElement element, String elementDisplayName, int timeoutInSeconds, int intervalInSeconds) {
    waitUntilElementToBeHidden(element, elementDisplayName, timeoutInSeconds, intervalInSeconds, new IWaitable<HtmlElement>(){
        public boolean meetIt(HtmlElement e) {
            return !e.isDisplayed();
        }
    });
}

/**
 * Wait until the the instance of HtmlElement to be hidden in given time period, once it's hidden, the method will end the waiting to save time.
 *
 * @param element the waited instance of HtmlElement
 * @param elementDisplayName the element display name in report
 * @param timeoutInSeconds longest waiting time in seconds
 * @param intervalInSeconds the interval seconds between each checking loop
 * @param waitable the instance implementing {@link IWaitable} interface to define what event to wait for
 */
public void waitUntilElementToBeHidden(final HtmlElement element,
                                       String elementDisplayName,
                                       int timeoutInSeconds,
                                       int intervalInSeconds,
                                       final IWaitable<HtmlElement> waitable) {
    final int validatedTimeoutInSeconds = timeoutInSeconds <= 0 ? timeoutInSeconds * -1 : timeoutInSeconds;
    final int validatedIntervalInSeconds = intervalInSeconds <= 0 ? (intervalInSeconds == 0 ? 1 : intervalInSeconds * -1) : intervalInSeconds;
    Logging.info(String.format("Waiting at longest %s seconds for %s to be hidden...", timeoutInSeconds, elementDisplayName));

    try {
        new Wait() {
            public boolean until() {
                return waitable.meetIt(element);
            }
        }.wait(String.format("The %s is still displayed after %s seconds.", elementDisplayName, validatedTimeoutInSeconds),
                validatedTimeoutInSeconds * 1000,
                validatedIntervalInSeconds * 1000);
    }
    catch (RuntimeException e) {
        capturePageSnapshot();
        throw e;
    }
}
}
```

Read-Only | Smart Insert | 1 : 1