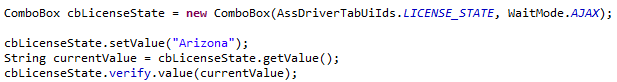
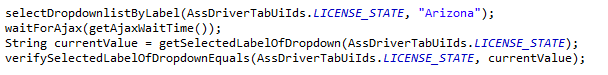
**Main differences between Automation frameworks for AAA Auto product and AAA Property product**

1. **UI primitives**

ISTF based framework for AAA Property interacts with UI elements with help of control objects with specified types: TextBox, Button, Link, ComboBox, CheckBox, etc. Each element type has its own set of operations that can be performed with it. You can create control object for your tests that can be used as any object in object-oriented programming:

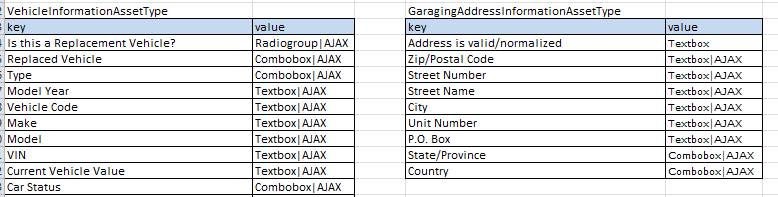
****

In tests for AAA Auto you haven’t control types and should decide what function is suitable for this control every time of interacting with it. You have multiple functions in class SeleniumUtils that you can access any time by passing control’s ID or XPath as parameter:

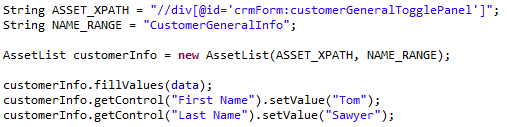


1. **UI composites: AssetList**

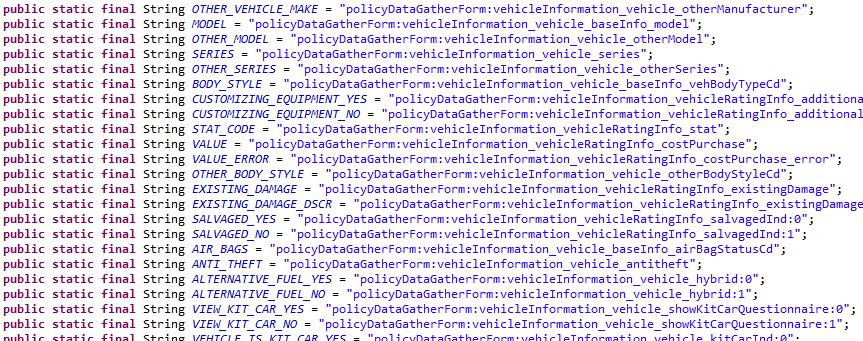
ISTF based tests use AssetLists for filling multiple controls on one page. Information about controls that are included in assetlist is stored in Excel file in form of table with control’s name, control’s type and WaitMode that would be used for delay after actions with this control:



For filling several fields on page you can use one object of AssetList class. You should know only XPath of page block that contains all controls. And also you can access certain control by its name, its type and WaitMode will be provided by assetlist:

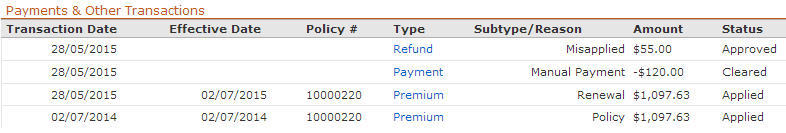


For Auto product field names are stored right in code and ID or XPath of every field should be known. So much more code needed for work with controls, see Appendix at the end of document for comparison of code amount.

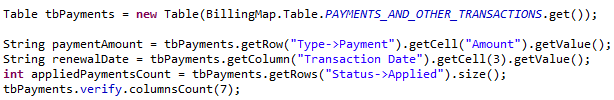


1. **UI composites: Table**

ISTF also have Table class that provides set of methods to manipulate with the table structures: Cell, Row, Column primitives. It’s very helpful because lot of data represented by tables in EIS-based applications. You can use it to interact with all information in table like this:



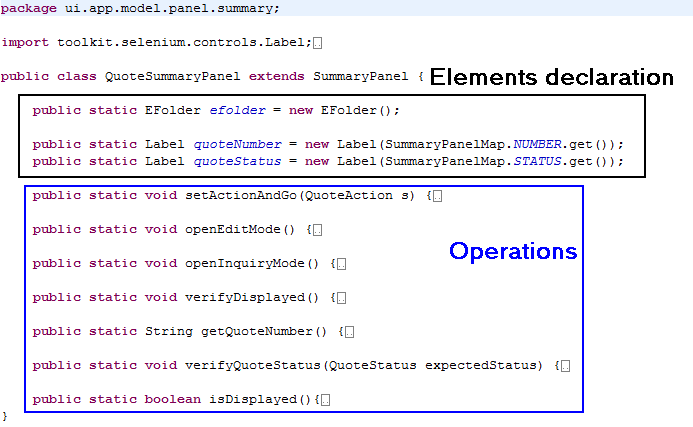
Known only ID or XPath of this table, no additional info about rows and columns needed:



For Auto tests similar with AssetList you should know XPath for every table’s cell to get value from it. A

1. **Panels objects**

Test framework for Property product has static instances of controls for every panel that you interact.



In test every control can be accessed in way PanelName.controlName.action():

PremiumTabEditorPanel .fillPremium(data);

PremiumTabEditorPanel.btnCalculatePremium.click();

Framework for Auto product has no control objects and you should create new instance of classes with filling methods or set/get functions before using them.

AssGeneralTabFilling assGeneralTabFilling = new AssGeneralTabFilling(getSelenium()); assGeneralTabFilling.insuredFilling(getDataSet(), "InsuredFillData6");

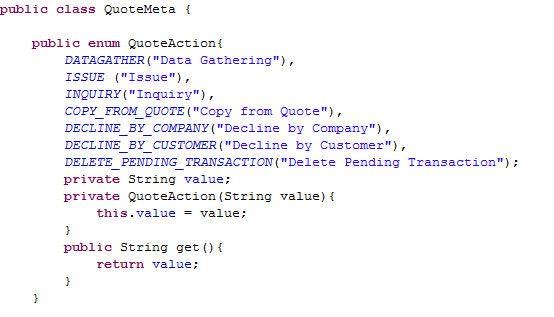
1. **Application model**

For Property tests you can access End User application, Admin application and connect SQL database in a single test method with help of separate Application objects or DBHelper utilities.

For Auto tests you can access only EU application or only Admin application or only Database in a single test class by inheritance of AppBaseClass, AdminBaseClass or DatabaseBaseClass. The simplest way to start Job in Admin application is starting utility method with specified parameter = JobName in test suite.

1. **MetaData**

One of the simplest, but efficient and convenient techniques that is used in test framework for Property product is UI metadata. Under the metadata term we mean all constants that you see on UI and use in tests many times (tab names, actions, etc.). It’s organized as a customized enumeration:



So during test creation you shouldn’t open application to copy exact writing of some string, you can find it in your project. And if this string will be changed in application, you can change it in one place in test framework, not in every test that uses it.

1. **Verification mechanism**

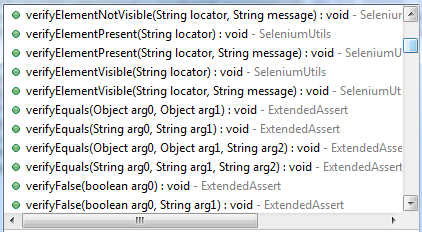
In tests for Property product common asserts, that can be used in any time, can be accessed thru class CustomAssert that supports both JUnit and TestNG testing frameworks:

CustomAssert.assertTrue("Message", condition);

Asserts for verifying some control’s states can be accessed in way control.verify.verifyAction(). Only methods that are supported by this control are available:

PolicySummaryPanel.status.verify.value(PolicyMeta.Status.ACTIVE);

In tests for Auto product all verification functions are available any time and you should pay attention to choose method that you need for specific control:



You should use method checkForVerificationErrors(); after all verifications or use methods starting with “check” to fail test in case of failed verification.

1. **WaitMode**

ISTF has enumeration WaitMode with values AJAX, PAGE, SLEEP, NONE that determines origin of delay before next action for every control (NONE is default). In time of control’s initialization you can assign wait mode to control and it would be applied for all actions with control if needed. Wait timeouts are configured in properties file.

Tests for Auto product uses different function for action with wait and action without wait:

click(locator)

clickAndWait(locator, seconds)

Or separate wait function waitForPage(seconds) or waitForAjax(seconds)after action.

Timeouts for wait are set in property file and accessed by functions

getMiddleWaitTime()

getLongWaitTime()

getAjaxWaitTime()

1. **DateTime**

ISTF has class DateTime for work with date and time in format:



So you have convenient way to change date and output it in any format.

Tests for Auto product uses functions like addDaysToSpecificDate(String specificDateString, int days) that works only with “MM/dd/yyyy” date format or you should use standard Java Date class to format date.

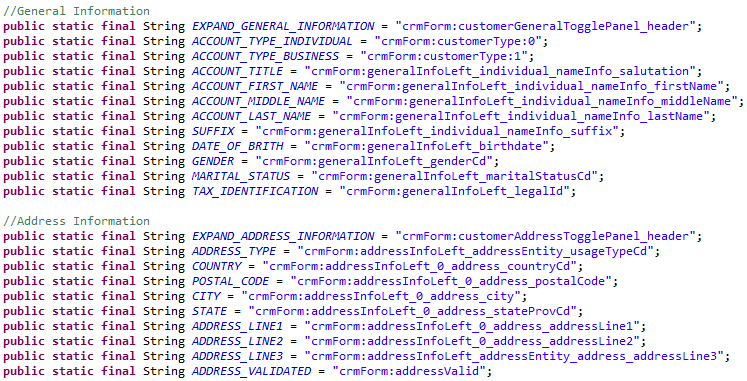
1. **HTTP utilities**

Tests for Property product use HTTP utilities from ISTF. These utilities can be used to get some data from application or perform some actions without opening application in browser. It significantly decreases time of performing some routine actions that are repeated many times. Of course they are used not for testing but for some test preconditions.

**Appendix: Comparison of code needed to create customer by different test frameworks**

1. **Test framework for Auto product**

IDs of fields that will be filled:



Functions for setting and getting value for every field:



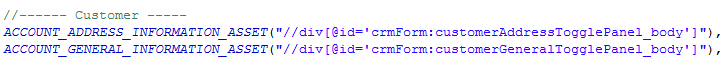
And so on for another 100 code lines…

Filling function that inputs values from Excel file with data to fields:

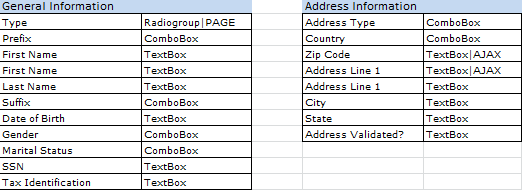


1. **Test framework for Property product**

XPath of 2 blocks that contains fields for assetlists:

****

Application metadata that contains field names, types and WaitModes:

****

Code of panel class that contains initialization of 2 assetlists and filling function:

