Week 4 Unit 4

Writing Integration Test with OPA

Please perform the exercises below in your app project as shown in the video.

Table of Contents

[1 Add a New OPA Test 2](#_Toc449333010)

[2 Add a New OPA Page Object 3](#_Toc449333011)

## Preview

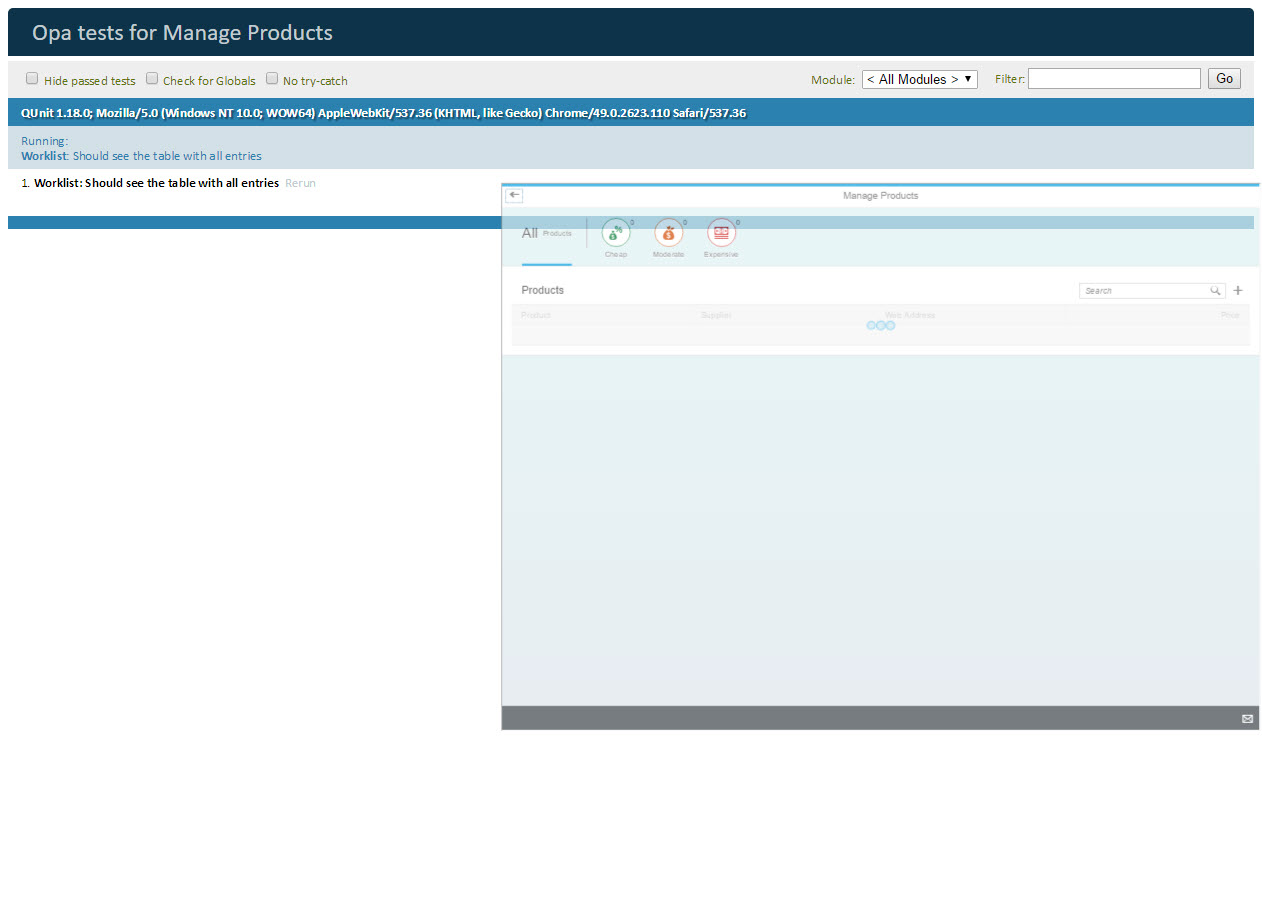


Figure 1 - Preview of the app after doing this unit’s exercises

# Add a New OPA Test

**Let us now add our own OPA test to the Worklist journey. You may have noticed that the products list displays a ‘plus’ button next to the search field in the header toolbar. We would like to write a simple OPA test to check whether this button opens the ‘Add’ view.**

## webapp/test/integration/pages/Worklist.js

|  |
| --- |
| …  var sViewName = "Worklist",  sTableId = "table",  **sAddButtonId = "addButton",**  sSearchFieldId = "searchField",  sSomethingThatCannotBeFound = "\*#-Q@@||",  … |

**First add the add button id to the variable declaration. You can find this id in the file ‘Worklist.view.xml’.**

## webapp/test/integration/pages/Worklist.js

|  |
| --- |
| sap.ui.define([  "sap/ui/test/Opa5",  "sap/ui/test/actions/Press",  "sap/ui/test/actions/EnterText",  "sap/ui/test/matchers/AggregationLengthEquals",  "sap/ui/test/matchers/AggregationFilled",  "sap/ui/test/matchers/PropertyStrictEquals",  "opensap/manageproducts/test/integration/pages/Common",  "opensap/manageproducts/test/integration/pages/shareOptions"  ], function(Opa5, Press, EnterText, AggregationLengthEquals, AggregationFilled, PropertyStrictEquals, Common, shareOptions) {  …  Opa5.createPageObjects({   onTheWorklistPage : {  baseClass : Common,  actions :  …  iSearchForSomethingWithNoResults : function () {  return this.iSearchForValue([new EnterText({text: sSomethingThatCannotBeFound}), new Press()]);  },   **iPressAdd : function () {  return this.waitFor({  id: sAddButtonId,  viewName : sViewName,  actions : new Press(),  errorMessage : "Add button not found"  });  },**  … |

**Now we can append our iPressAdd function to the ‘actions’ section of the Worklist page. This function consists of a waitFor statement that checks for controls with id ‘addButton’ inside the ‘worklist’ view. Once the button is found, waitFor executes the Press action for the ‘Add’ button.**

**The Press action will not only trigger the ‘tap’ event on the button. In addition it makes sure that the button gets the focus. If the button is not visible, which is a common source of errors, the Press action will log an error.**

# Add a New OPA Page Object

## webapp/test/integration/pages/NewProduct.js (NEW)

|  |
| --- |
| **sap.ui.require([  "sap/ui/test/Opa5",  "opensap/manageproducts/test/integration/pages/Common"  ], function(Opa5, Common) {  "use strict";   var sViewName = "Add",  sPageId = "page";    Opa5.createPageObjects({  onTheNewProductPage : {  baseClass : Common,   assertions : {  iShouldSeeThePage : function () {  return this.waitFor({  id : sPageId,  viewName : sViewName,  success: function () {  Opa5.assert.ok(true, "The 'New Product' title is shown.");  },  errorMessage : "Did not display the 'New Product' page."  });  }  }  }  });  } );** |

**Now we can proceed by adding a new page object for new products. OPA uses page objects to structure its journeys. Typically each page object contains actions and assertions which are for controls in one view. In our case, this would be the ‘Add’ view.**

**We implement our assertion iShouldSeeThePage which checks whether the ‘page’ is displayed . Again this is done by a waitFor statement which checks in the DOM tree whether the entry ‘page’ is available in the ‘Add’ view. In this case, the success function is called, which executes a QUnit assert.ok. If not, an error message is displayed and the test fails.**

## webapp/test/integration/AllJourneys.js

|  |
| --- |
| …  sap.ui.require([  "sap/ui/test/Opa5",  "opensap/manageproducts/test/integration/pages/Common",  "sap/ui/test/opaQunit",  **"opensap/manageproducts/test/integration/pages/NewProduct",**  "opensap/manageproducts/test/integration/pages/Worklist",  …  ],  … |

**To make sure our new page is loaded, we need to require it in ‘AllJourneys.js’**

## webapp/test/integration/ObjectJourney.js

|  |
| --- |
| sap.ui.define([  "sap/ui/test/opaQunit"  ], function (opaTest) {  "use strict";   QUnit.module("Object");   …  **opaTest("Should see the 'New Product' view after pressing the 'Add' button", function (Given, When, Then) {  *// Arrangements* Given.iStartMyApp();  *//Actions* When.onTheWorklistPage.iWaitUntilTheTableIsLoaded().and.iPressAdd();  *//Assertions* Then.onTheNewProductPage.iShouldSeeThePage().and.iTeardownMyAppFrame();  });**  } ); |

**We are all set now to write our OPA test. Let us add it to the ‘ObjectJourney’. A journey consists of a series of integration tests that belong to the same context such as navigating through the app. Similar to the QUnit test implementation, OPA5 uses QUnit, that's why we first set up a QUnit module ‘Object’ that will be displayed on our result page.**

**The function opaTest is the main aspect for defining integration tests with OPA. Its parameters define a test name and a callback function that gets executed with the following OPA5 helper objects to write meaningful tests that read like a user story.**

* **Given: on the given object we can call arrangement functions like iStartMyApp to load our app in a separate iFrame for integration testing.**
* **When: contains custom actions that we can execute to get the application in a state where we can test the expected behavior.**
* **Then: contains custom assertions that check a specific constellation in the application, and the teardown function that removes our iFrame again.**

**In our test, we first start the app and wait for the table to be populated on the worklist page. Then we press the ‘Add’ button on the worklist page. After that, we check whether the new product page is displayed. Finally, we remove the iFrame again from our test page.**

**As you can see, the test case reads like a user story. We actually do not need the implementation of the methods yet to understand the meaning of the test case. This approach is called "Behavior Driven Development" or simply BDD and is popular in "Agile Software Development".**

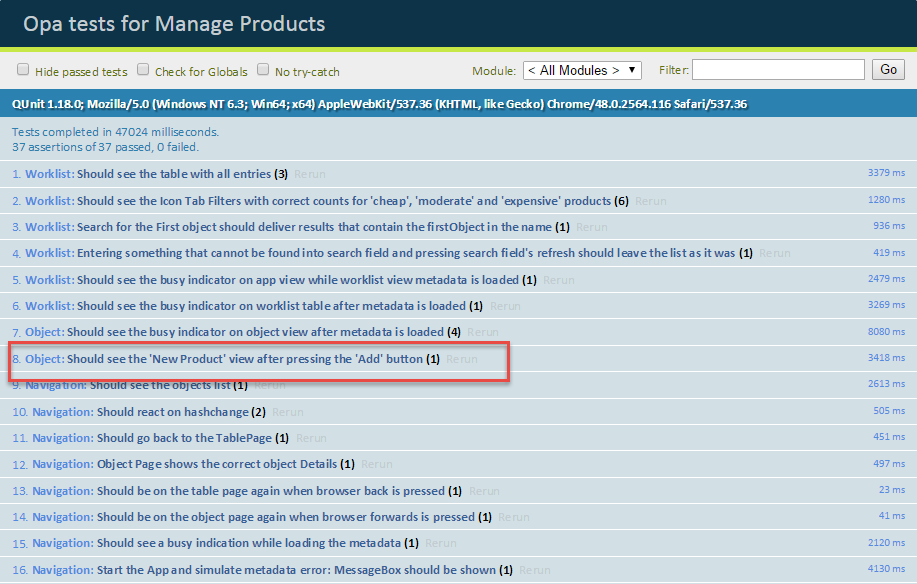


Figure 2 - The OPA test result page

**Let us execute our new OPA test: Select the file ‘opaTest.qunit.html’ in the folder test/integration of your SAP Web IDE project and run it, or select the run configuration ‘run OPA tests’ from the ‘run’ menu.**

**You will see a QUnit result page and a preview of the app where the tests are executed on. At the end of the ‘Worklist’ journey, the new test is executed.**

Related Information

[API Reference: sap.ui.test.OPA5](https://sapui5.hana.ondemand.com/#docs/api/symbols/sap.ui.test.Opa5.html)

[API Reference: sap.ui.test.matchers](https://sapui5.hana.ondemand.com/#docs/api/symbols/sap.ui.test.matchers.html)

**Coding Samples**

Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.