TRICENTIS

Automation Specialist Level 2

STUDENT EXERCISE WORKBOOK

SURVEY

Tricentis Automation Specialist | Level 2

Student Exercise Workbook

- Version 2018_12
- Designed to be used with Tricentis Tosca version 12.x

Student Exercise Workbook

This exercise workbook is designed to provide a collection of exercises on the methods and concepts covered in the Tricentis Automation Specialist Level 2 training.

Legal Notice

- Tricentis GmbH
- Leonard-Bernstein-Straße 10
- 1220 Vienna
- Austria
- Tel.: +43 (1) 263 24 09
- Fax: +43 (1) 263 24 09-15
- Email: academy@tricentis.com

TABLE OF CONTENTS

PREFACE

About this workbook

This workbook is specifically designed to supplement training of the **Tricentis Automation Specialist Level 2**.

The workbook is arranged in sections. Each section contains a number of exercises which give detailed instructions on how to perform certain functions in Tosca.

Tricentis recommends completing the exercises before continuing to the next section and to take the related online exams in order to achieve high impact learning.

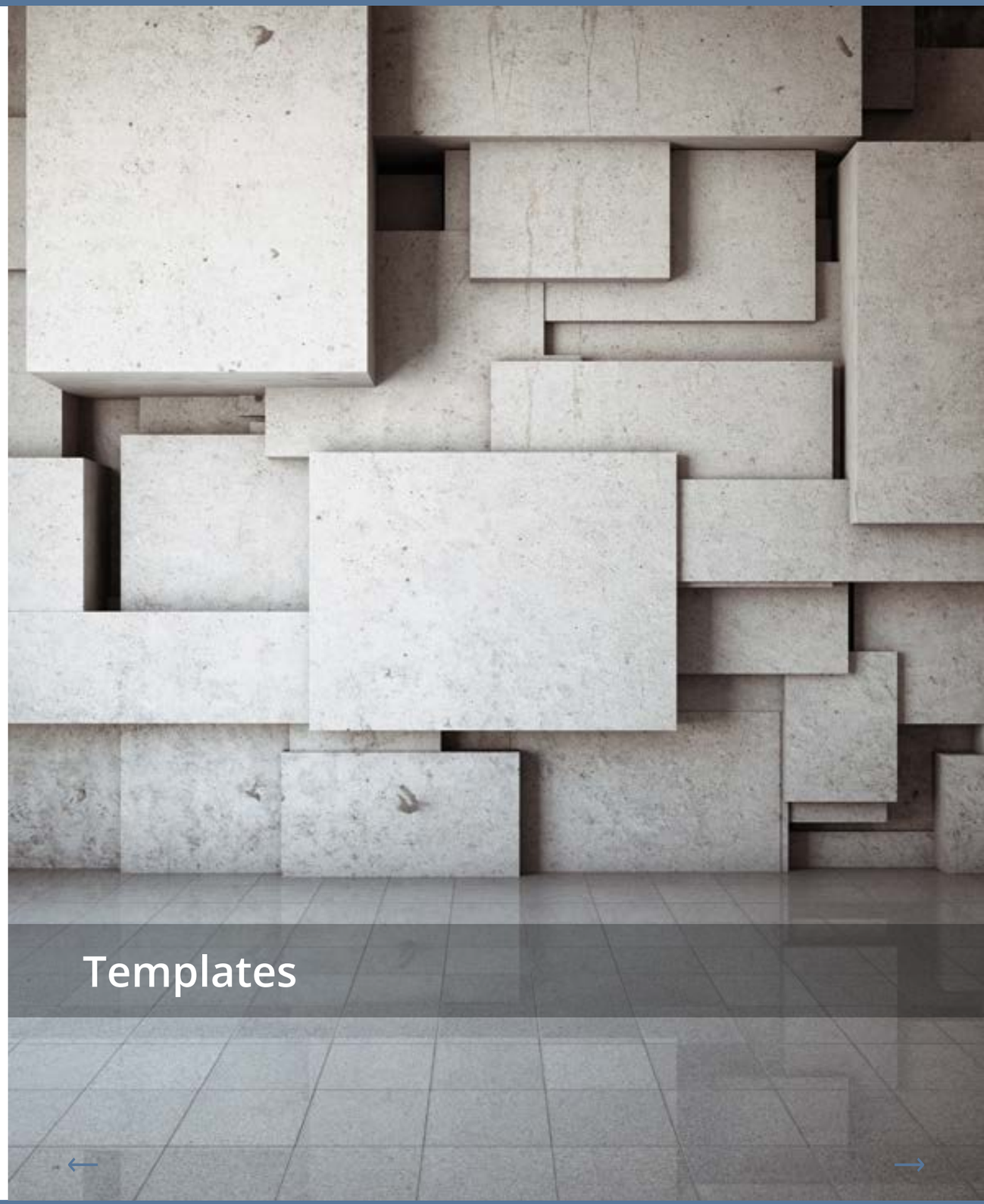
For each exercise there will be a lesson video that explains how to complete the exercise, most exercises will also have a solution video that explains how to complete the exercise in full.

This workbook is not aiming to be a complete manual.

Recommended learning material

In addition to this workbook it is necessary to use the following material to complete the exercises successfully.

- A sample **Web Shop** application is used for most of the exercises. Please use the link below to start the sample Web Shop application: http://demowebshop.tricentis.com/ In addition to this workbook it is necessary to use the following material to complete the exercises successfully.
- Please use **Internet Explorer** as the default browser for this training.



TEMPLATES

Exercise 1 – Create Login Data TestSheet

Objective

By the end of this exercise, you will be able to create and manipulate a basic TestSheet.

Why is this important?

The TestSheet is where you store the test data needed for automating TestCases, and the Template will use the data from the TestSheet to create TestCases.

Project perspective

You are the tester for the DemoWebShop. According to the requirement data, the login process is important, and you should test it using multiple accounts. In this case, we will create the account details and store them in a TestSheet.

Key elements:

TestSheet Attributes Instances

Instructions

1. Register three separate new user accounts in the **DemoWebShop**. Make sure to **note** the newly created **emails** and **passwords**. They must all be different. The URL for the DemoWebShop is: http://demowebshop.tricentis.com/ ☐

Examples of the emails and passwords:

Email	Password
your.name@example1.test	Tosca123!
your.name@example2.test	Tosca1234!
your.name@example3.test	Tosca12345!

2. Open Tosca Commander and create a new Workspace using the Workspace Template “**BaseSubset_AS2.tsu**". ☐
3. In the TestCaseDesign section, select the folder "**1 Create Log In Data TestSheet**". Create a new TestSheet in this folder and name it "**Log In Data**". ☐
4. Right click on the newly created TestSheet and create two new Attributes named "**Email**" and "**Password**". (You can also use the shortcut Ctrl+N,Ctrl+A to create a new Attribute.) ☐
5. Right click on the Attribute **Email** and create a new **Instance** with the first email value created in step 1. (You can also use the shortcut Ctrl+N,Ctrl+I to create a new Instance.) ☐
6. Repeat the process to add two additional Instances of the **Email** Attribute using the second and third emails as values. ☐
7. Right click on the Attribute **Password** and create a new Instance with the first password value created in step 1. ☐
8. Repeat the process to add two additional Instances of the **Password** Attribute with the second and third password. ☐
9. To add different login credentials, we must add multiple different Instances on the TestSheet itself. To do this, right click on the TestSheet **Log In Data** and create the following Instances: **User1**, **User2** and **User3**. ☐
10. Double click on the TestSheet **Log In Data** in the navigation pane. In the working pane, we now see the users we have just created next to the TestSheet. In the row **Email**, set the values for each of the 3 users by selecting from the drop down as per the table below: ☐

Attribute	User1	User2	User3
Email	<Your email 1>	<Your email 2>	<Your email 3>

11. Set the values of Password for each of the three users by selecting from the drop down as per the table below: ☐

Attribute	User1	User2	User3
Password	<Your password 1>	<Your password 2>	<Your password 3>

Examples of the emails and passwords:

Attribute	User1	User2	User3
Email	your.name@example1.test	your.name@example2.test	your.name@example3.test
Password	Tosca123!	Tosca1234!	Tosca12345!

Exercise 2a – Log In Process

Objective

By the end of this exercise, you will have created a TestCase that automates the process of a simple login.

Why is this important?

This TestCase will be converted into a Template, which will then provide the skeleton for that Template.

Project perspective

The DemoWebShop, like most other online web stores, requires the user to log in before they can purchase goods from the store. You need to create a TestCase that automates the login process in order to avoid doing it manually each time.

Instructions

1. Navigate to the TestCases root folder and add the Test Configuration Parameters “**Email**” and “**Password**”. ☐
2. Enter the first pair of Email and Password created in lesson 1 as values for the Test Configuration Parameters created above. ☐
3. Add the Test Configuration Parameter “**Browser**” with the value "**InternetExplorer**". ☐
4. Within the root folder “**TestCases**”, create a new folder called **Template**. In there, create a subfolder called "**2a Log In Process**". ☐
5. In the subfolder "**2a Log In Process**", create a new TestCase called "**Log In Process**". ☐
6. Add the Reusable TestStepBlocks "**Precondition**" and "**Postcondition**" from the Library to the TestCase "**Log In Process**". ☐

Hints

» To recap the creation of Test Configuration Parameters, please see Lesson 2 of Automation Specialist Level 1.

» TestSteps and Reusable TestStepBlocks can easily be searched and added to a TestCase by using the “Add TestStep” function (CTRL+T).

Exercise 2b - Resolve Reference

Objective

By the end of the exercise, you will be able to use the Resolve Reference command to break the connection between the Reusable TestStepBlock and the TestCase.

Why is this important?

Some elements of the Reusable TestStepBlocks will not be relevant for the TestCases that the Template generates. The Resolve Reference command allows you to use the structure of the ReusableTestStepBlocks and to amend the values without changing the Values in the Library.

Project perspective

If we want to use the Email and Password data from the TestSheet in the TestCase, we must be able to link them to the Email and Password TestStepValues. Right now, the Log in TestStep belongs to the Reusable TestStepBlock Precondition. You need to Resolve the Reference to edit the TestStep.

Key elements:

Reusable TestStepBlock

Instructions

1. Duplicate the TestCase folder "**2a Log In Process**" and rename it to "**2b Resolve References**". ☐
2. Right click on the Reusable TestStepBlock Reference “**Precondition_Reference**” and choose “Resolve Reference”. ☐
3. Create a new folder in the TestCase “**Log In Process**” and name it “**Workflow**”. Place the folder in between **Precondition** and **Postcondition_Reference**. ☐
4. Move the TestStep **Log In** from the **Precondition** folder to the **Workflow** folder. ☐
5. Enter the following **Values** for the TestStepValues as per the table below: ☐

	TestStep	**TestStepValue**	**Value**	**ActionMode**
Precondition	Open the Application	Url	http://demowebshop.tricentis.com	Input
Workflow	Log in	Email	{CP[Email]}	Input
		Password	{CP[Password]}	Input

6. Run the TestCase created in the **ScratchBook**. ☐
7. Set the "**WorkState**" of the TestCase to **Completed**. ☐

Hints

» Once a reference is resolved, it is not possible to revert the action. You can use the undo arrow on the top left of Tosca. However, this option is only available if you have not saved the project after resolving the references.

Exercise 2c - Convert to Template

Objective

By the end of the exercise, you will be able to create a Template from the TestCase Log In Process.

Why is this important?

The Template is one of the main points of focus for TestCaseDesign. Using a TestCase Template will allow you to create several new TestCases at once.

Project perspective

The login process of the DemoWebShop needs to be tested multiple times with many different accounts. It is time-consuming to duplicate one TestCase multiple times and then edit the values manually. It is also hard to maintain the TestCases when something is changed. Instead, you can create a Template from your Log in Process TestCase and link the data from the TestSheet to it.

Instructions

1. Duplicate the folder "**2b Resolve References**" and rename it to "**2c Convert to Template**". ☐
2. Right click on the TestCase "**Log In Process**" and select "**Convert to Template**". ☐

Exercise 3a - Link Template and Values

Objective

By the end of this exercise, you will be able link a Template to a TestSheet by using the values from that TestSheet.

Why is this important?

For a Template to work, it needs to supply the TestCases with the necessary information. One of the sources of information we can use is the TestSheet.

Project perspective

You have just created a Template and a TestSheet that fit the needs of your project. However, they are still two separate objects which cannot communicate with each other yet. For the Template to find the correct data, you will need to link the values to the appropriate TestStepValues.

Key elements:

Attribute

Instance

Instructions

1. Duplicate the TestCase folder "**2c Convert to Template**" and rename the new folder to "**3a Link Template and Values**". ☐
2. In the **TestCaseDesign** section, locate the "**Log In Data**" TestSheet that was created in Exercise 1. Drag and drop the "**Log In Data**" TestSheet onto the "**Log In Process**" Template. ☐
3. In the TestStep Workflow >> "**Log in**" of the Template, delete any data in the "**Email**" and "**Password**" TestStepValues. ☐
4. Drag and drop the corresponding Attributes from the "**Log In Data**" TestSheet to the value column of the "**Log in**" TestStep. ☐

TestStep Folder	**TestStep**	**TestStepValue**	**Value**
Workflow	Log in	Email	{XL[Email]}
		Password	{XL[Password]}

Hints

- Make sure you have linked the correct TestSheet to the right TestCase Template before Linking the Values.
- "Jump to Schema Definition" from the context menu on the linked TestCase will show the TestSheet to which the Template is linked. You can also check the name of the linked TestSheet using the Property tab.
- Attributes can also be linked manually following the syntax: {XL[Attribute.SubAttribute]} or {XL[Attribute.SubAttribute.SubSubAttribute]} but this is **not best practice**.

Exercise 3b - Instantiation

Objective

By the end of this exercise, you will be able to Instantiate a Template and generate TestCases from the test data provided.

Why is this important?

The final stage of Template creation is to automatically generate all the TestCases from the TestSheet. This is achieved through Instantiation.

Project perspective

After linking the TestSheet to the Template, you can then Instantiate the Template, which automatically generates multiple TestCases that will test the login process for the three different user accounts on the TestSheet.

Key elements:

TemplateInstance folder

Instructions

1. Duplicate the folder "**3a Link Template and Values**" and rename it to "**3b Instantiation**". ☐
2. Right click on the TestCase Template . Select the ellipsis icon from the context menu to see extra options and click on "**Create TemplateInstance**". Select "**Yes**" on the prompt asking whether you want to start the instantiation now. ☐
3. Run the "**TemplateInstance of Log In Process**" folder in the **Scratchbook**. ☐

Hints

» You can also find the Instantiate button in the TestCases section of the Ribbon..

Exercise 4a - Adapt the TestSheet

Objective

By the end of this exercise, you will be able to adapt a TestSheet by updating existing Attributes and Instances or creating new ones.

Why is this important?

As the test scope changes, the TestSheet also has to change. It is important that you can change basic test data by yourself, which makes testing possible.

Project perspective

You received a new Login TestSheet from your Test Design Specialist that includes a new Attribute, Validation. For the Valid Values TestSheet Instance, you will need to fill in valid login credentials.

Instructions

1. Navigate to the TestCaseDesign section and find the folder "**4a Set Conditions**". ☐
2. Double click on the TestSheet "**Log In Data - Including Invalid users**". ☐
3. For the "**Value - Email**" Attribute, add an Instance with the first of your previously registered emails. ☐
4. For the "**Value - Password**" Attribute, add an Instance with the password that fits with your above email. ☐
5. Double click on the TestSheet to go back to the TestSheet view. For the TestSheet Instance "**Valid Values**", choose your newly created email and password values from the drop-down menu on the "**Value - Email**" and "**Value - Password**" rows. ☐

Hints

» If your valid password is the same as the existing valid password (Tosca1234!), you don't need to create a new Password Instance.

Exercise 4b - Adapt the Template

Objective

By the end of this exercise, you will be able to change the necessary TestSteps to create a new, functioning Template that fits the new Requirements and new TestSheet.

Why is this important?

The TestCase used to create the Template does not validate the error message from the SUT when we fail to log in. Therefore, the Template must be amended to include this function. Also, the existing Template no longer reflects the data in the new TestSheet.

Project perspective

You already have a Template that tests a successful Login Process. You now want to test when a login fails as well. For this, you need to see an error message and verify that it is the correct one.

Instructions

1. Navigate to the TestCases section. ☐
2. Duplicate the TestCase folder "**3b Instantiation**" and rename the new folder to "**4b Adapt the Template**". ☐
3. Delete the TemplateInstances folder "**TemplateInstance of Log In Process**". ☐
4. Add the Module "**Login Errors**" to the TestStep folder "**Workflow**" and rename it to "**Validate Log in Error Message**". ☐
5. Link the "**Log In Data** - **Including Invalid users**" TestSheet to the new "**Log in Process**" Template. ☐
6. In the TestStep "**Log in**", delete the existing Value of "**Email**" and "**Password**" TestStepValues. ☐
7. Link the TestStepValues "**Email**" and "**Password**" to the Attributes "**Value** - **Email**" and "**Value** - **Password**" in the new TestSheet "**Log In Data** - **Including Invalid users**". ☐
8. Right click on the TestStep folder "**Postcondition _Reference**" and click on "**Resolve Reference**". ☐

Hints

» Before linking a new Attribute to a TestStepValue that is already linked to another Attribute, you need to delete the Value of the TestStepValue first.

Exercise 5a – Set Conditions

Objective

By the end of this exercise, you will be able to set the Conditions needed to correctly steer the TestCases.

Why is this important?

For the Template to work with all sets of data, we need to set Conditions so that TestSteps or TestStepValues run only when they are needed.

Project perspective

You have adapted the Template to include a TestStep that verifies if the Login Process is completed. In case the Log in TestStep fails, executing the Log out TestStep would be impossible, and would result in the TestCase failing. For Tosca to know which TestStep or TestStepValue to steer in certain situations, we set Conditions.

Key elements:


Condition Folder


Condition TestStep

Instructions

1. In the TestCase section, duplicate the folder "**4b Adapt the Template**" and rename it to "**5a Set Conditions**". ☐
2. In the TestCaseDesign section, navigate to the "**Log in Data - Including Invalid users**" TestSheet in folder "**4a Set Conditions**" and find the Attribute "**Validation**". ☐
3. Navigate back to the "**Log In Process**" Template in folder "**5a Set Conditions**". For the Workflow > "**Validate Log in Error Message**" TestStep, set the values of TestStepValues "**Wrong Email Error**" and "**Wrong Password**" to "**Innertext==**". ☐
4. Then, link the "**Validation**" Attribute from the TestSheet to the TestStepValues by dragging and dropping the Attribute onto the TestStepValues. ☐

Validate Log In Error M...			
Wrong Email Error	InnterText == {XL[Validation]}	Verify	String
Wrong Password	InnterText == {XL[Validation]}	Verify	String

5. Drag and drop the Instances of the Attribute "**Validation**" onto the Conditions of the TestStep "**Validate Login Error Message**" and its TestStepValues. See the example below and adjust them where needed: ☐

Validate Log In Error M...				**Validation != "Success"**
Wrong Email Error	InnterText == {XL[Validation]}	Verify	String	Validation == "Please enter a valid email address."
Wrong Password	InnterText == {XL[Validation]}	Verify	String	Validation == "Login was unsuccessful. Please correct the errors and try again."

6. Drag and drop the Instance “**Success**” in the “**Validation**” Attribute onto the Condition of the TestStep “**Log out**”. ☐

TestStep Folder	TestStep	Condition
Postcondition		
	Log out	Validation == "Success"

Hints

- To add Conditions to the TestSteps, make sure that the "Condition" column is visible. By default, this column is not visible. Remember to use the "Column Chooser" to add additional columns.
- To check if an element of the Testcase has a Condition, check if its icon has changed.

Exercise 5b – Instantiation

Objective

By the end of this exercise, you will be able to instantiate the amended Template with data from the new TestSheet to create new TestCase Instances.

Why is this important?

The main purpose of creating a Template is to automatically generate all the required TestCases using the TestSheet. This is done through Instantiation.

Project perspective

After adapting the Template as well as linking new values and setting Conditions, you need your TestCases to reflect those changes. Instantiating will help you accomplish this task.

Instructions

1. Duplicate the folder "**5a Set Conditions**" and rename it "**5b Instantiation**". ☐
2. Instantiate the Template by right clicking on the Template and choose “**Create TemplateInstance**”. You can find this option by clicking the ellipsis in the mini-toolbar. Select “**Yes**” on the prompt asking whether you want to start the instantiation now. ☐
3. Run the TestCases created in the **ScratchBook**. ☐

Hints

- As we are linking a new TestSheet, we will need to instantiate the TestCase again instead of Reinstantiating it.

Exercise 6a – Link TemplateInstances to a RequirementSet

Objective

By the end of this exercise, you will be able to link the TemplateInstances to the RequirementSet.

Why is this important?

By doing so you will know which Requirements of your SUT are covered with your TestCases and therefore how your risks are covered. Linking the TemplateInstance folder means that if any new TemplateInstances are generated in the folder, they will automatically be linked with the Requirements.

Project perspective

Now that you have completed your TemplateInstances, you will need to link them to the provided RequirementSet so that you and your Test Manager can have an overview of the overall project.

Key elements:

TemplateInstances Folder

RequirementSet

Requirement

Instructions

1. Navigate to the "**Requirements**" section. Select the folder "**DemoWebShop**". ☐
2. Expand the "**WebShop Frontend**" RequirementSet and navigate to Customer tasks > **Log in**. ☐
3. Drag and Drop the "**TemplateInstance of Log in Process**" folder onto the "**Log in**" Requirement in order to link them. ☐

Exercise 6b – Link TemplateInstances to an ExecutionList

Objective

By the end of this exercise, you will be able to create an ExecutionList and link the TemplateInstances.

Why is this important?

ExecutionLists are the last stage of the test development process, and the TemplateInstances must be linked to the ExecutionList and Requirements for the testing process to complete. This link also helps us understand how much Requirement coverage has been achieved so far.

Project perspective

You have set up the Template and generated all the necessary TestCases for your testing purposes. However, it is not enough to just run them in the Scratchbook. Since your manager and the developers need to see reports on how many times the system was tested and which Requirements were covered.

Key elements:

ExecutionLists

Instructions

1. Navigate to the Execution section in Tosca and create an ExecutionList folder under the root Execution folder. ☐
2. In the newly created ExecutionList Folder create an ExecutionList named "**6b Log in Process**". ☐
3. Drag and drop "**TemplateInstance of Log in Process**" TemplateInstance Folder to the ExecutionList "**6b Log in Process**" to link them. ☐
4. Run the ExecutionList. ☐

Exercise 7a – Build Template & Link Values

Objective

By the end of this exercise, you will be able to build a Template from a more advanced TestCase and link it to a more complicated TestSheet.

Why is this important?

The Log in Process TestCase we have used until now is just a simple TestCase for you to get used to the concept of TestCaseDesign and Templates. In real-life project scenarios, you are more likely to work with a more complicated TestCase, such as the Discount Code TestCase.

Project perspective

You are to test a new function in the DemoWebShop: the discount codes. Most online web stores offer special deals and promotions for marketing purposes. One of the most common forms of promotion is to offer discount codes. You need to test that the valid discount codes give the appropriate discounts and invalid codes entered do not give any discount. In order to do so, the first steps are to build a Template out of the existing TestCase and then link the TestSheet to that Template.

Instructions

1. Within the Template folder, create a TestCase folder named "**7a Build Template**". Copy the TestCase: "**Discount Code**" from the "**Automation Specialist 1 TestCases**" folder and paste it into the new folder. ☐
2. Convert the TestCase "**Discount Code**" into a Template. ☐
3. Resolve the References for all folders except "**Precondition_Reference**" and "**Order Product_Reference**". ☐
4. Add a new folder after "**Order Product_Reference**", named: **Verify Coupon Code**. ☐
5. Add the Reusable TestStepBlock "**Empty Shopping Cart, Log out and Close Browser**" to the Template. Resolve the Reference and rename it to "**Empty Shopping Cart**". ☐
6. Adapt the folder structure to match the following: ☐
   - Precondition_Reference
   - Order Product_Reference
   - Verify Coupon Code
   - Checkout Process
   - Start Checkout
   - Verification of Prices
   - Confirmation
   - Verification of Success
   - Empty Shopping Cart
   - Postcondition
7. Check that the structure of the TestCase matches the one outlined below. Change the TestSteps in the TestStep folders: "Verify Coupon Code" and "Empty Shopping Cart" as shown. ☐

TestStep Sub-Folder	TestStep
TestStep Folder Precondition_Reference	
	Business Parameters
TestStep Folder **Order Product_Reference**	
	Navigate to Apparel and Shoes
	Navigate to Blue Jeans
	Order the BlueJeans
TestStep Folder **Verify Coupon Code**	
	Navigate to Shopping Cart (moved from TestStep folder "Start Checkout")
	"Shopping Cart Procedures" Rename "Shopping Cart Procedures" to "Enter Code" (moved from TestStep folder "Start Checkout", then delete the "Start Checkout" folder.)
TestStep Folder **Checkout Process**	
Checkout Process	Billing address - Continue
Shipping (sub-folder of Checkout Process)	Shipping address - Continue
Shipping (sub-folder of Checkout Process)	Shipping method - Choose Ground
Checkout Process	Payment method - Choose PM
Checkout Process	Payment Information - Credit card
TestStep Folder **Verification of Prices**	
sub-folder ***Recovery Scenarios***	
	Verification of Prices
TestStep Folder **Confirmation**	
	Confirm the Order
TestStep Folder **Verification of Success**	
	Verify the Order Success
TestStep Folder **Empty Shopping Cart**	
	Navigate to Cart
	WHILE Statement
	Delete the "**Log out**" and "**Close the Application**" TestSteps
TestStep Folder **Postcondition**	
	Log out
	Close the Application

Exercise 7b – Link Values

Objective

By the end of this exercise, you will be able to link the Values from the TestSheet to their respective TestStepValues.

Why is this important?

Without the Values from the TestSheet, the Template contains no test data.

Project perspective

There are different discount codes that give different types of discounts. One code applies only to the goods, another code applies to the shipping methods. Additionally, there is a code that applies a discount to the total price. There are also different methods of shipping, each has its own cost. When we verify the prices, we will need to keep the shipping cost and the discount in mind.

First, we calculate the base price the customer needs to pay. Then we calculate the shipping cost and the discount. Finally, we add the values together for a total price.

Instructions

1. Duplicate the TestCase folder named "**7a Build Template**" and rename it "**7b Link Values**". ☐
2. Link the **TestSheet** "**Calculate Discount**" in folder **7 Calculate Discount** in the TestCaseDesign section to the **TestCase Template** "**Discount Code**". ☐
3. Add the necessary Attributes from the TestSheet "Calculate Discount" to the TestCase. TestSheet Attributes are required in the following TestStepValues: ☐

TestStep	TestStepValue	Path in TestSheet	Value
Business Parameters:	Email	Calculate Discount >> Precondition >> User >> Value - Email	XL Link to "**Value** - **Email**" from the '**Calculate Discount**' TestSheet
	Password	Calculate Discount >> Precondition >> User >> Value - Password	XL Link to "**Value** - **Password**" from the TestSheet
Enter Code	Discount Code	Calculate Discount >> Workflow >> Value - Discount Code	Delete the existing value. XL Link to "**Value** - **Discount Code**" from the TestSheet
	Coupon Applied	Calculate Discount >> Verifications >> Value - Message	Verify that the InnerText of the message has the appropriate value as in the TestSheet (adjust ActionMode to **WaitOn** to wait for the text to appear)

4.. Right click on the TestStep "**Verification of Prices**" and select "**Allow Reorder**". With the "Allow Reorder" function, TestStepValues can be rearranged. ☐

5. For the table Cart total, select the 2nd column. Create the TestStepValues by selecting the appropriate option from the grey <Cell>. Then, reorder them to form the following structure:
   - Sub-Total:
   - Sub-Total:
   - Shipping:*
   - Shipping:*
   - Shipping:*
   - Discount:
   - Discount:
   - Discount:
   - Discount:
   - Total:
   - Total:

   ☐
6. Fill in the following data for the TestStepValues: ☐

TestStepValue	Path in TestSheet	Value
Sub-Total		Use {Math} function to calculate verify the base price of the order without applying shipping cost or discount by multiplying the buffered price of a BlueJeans to the quantity **{MATH[{B[PriceBlueJeans]}*{B[QuantityBlueJeans]}]}**
Sub-Total		Buffer the Sub-Total
Shipping:*		Verify 10.00
Shipping:*	Calculate Discount >> Verifications >> Value - Discount %	Use {Math} function to Verify the percentage discount from the TestSheet for the cost of the Shipping (10.00) **{MATH[(1-{XL[Verifications.Value - Discount %]})*10]}**
Shipping:*		Buffer the shipping costs
Discount:	Calculate Discount >> Verifications >> Value - Discount %	Use {MATH} function to Verify, the sum of the buffered SubTotal and the shipping cost, multiply by the discount percentage from the TestSheet, the percentage discount for the total cost of the order including shipping costs: **[({B[SubTotal]}+{B[Shipping]})*{XL[Verifications.Value - Discount %]}]}**
Discount:	Calculate Discount >> Verifications >> Value - Discount Flat Rate	Verify, from the TestSheet, the flat rate discount: **-{XL[Verifications.Value - Discount Flat Rate]}**
Discount:	Calculate Discount >> Verifications >> Value - Discount %	Verify the percentage discount on the sub-total from the TestSheet **-{MATH[{XL[Verifications.Value - Discount %]}*{B[SubTotal]}]}**
Discount:		Buffer the discount
Total:		Verify the total price (taking into account shipping costs and the discount) **{MATH[{B[SubTotal]}+{B[Shipping]}+{B[Discount]}]}**
Total:		Verify the total (taking into account only shipping costs) **{MATH[{B[SubTotal]}+{B[Shipping]}]}**

Hints

- Previous Values will need to be cleared before you drag and drop new values, as the XL will be added at the end of the current value.
- Be careful with the {MATH[]} function, as some of the TestStepValues (for the Discount values) have a negative sign, as the value of the TestStep should be negative.
- Verifications have Attributes in the TestSheet.
- Depending on the SUT, the DataType for {MATH} may need to be changed to “Numeric”.
- To change Tosca numeric formats, such as the decimal character separator, go to Settings >> TBox >> Number formats.

Exercise 7c – Set Conditions & Instantiation

Objective

By the end of this exercise, you will have added the necessary Conditions to the Template for it to successfully instantiate.

Why is this important?

For a more complicated testing project, more Conditions are required on different levels of the Template in order to fulfill the needs of the project.

Project perspective

After linking the Values, we can see that there are many paths available for testing the discount codes.This depends on what type of shipping an order has, or what discount code was applied. The Template will need Conditions to generate the TestCases that test the situations that occur as a result of choosing certain paths.

If the TestSheet shows that there should be a discount applied on the shipping cost, the Shipping TestStepValue that calculates the discounted shipping cost should be used. Otherwise, the Shipping TestStepValue with the value 10.00 is used. Likewise, each Discount TestStepValue is only used when a certain type of discount code is entered. The total cost is also calculated differently depending on whether there is a discount applied or not.

If the Instance Character is invalid, certain TestSteps should be skipped completely.

Finally, you need to complete the process by instantiating the Template and linking the TemplateInstances to the ExecutionList.

Instructions

1. Duplicate the TestCase folder named “**7b Link Values**” and rename it “**7c Set Conditions**”. ☐

2. To ensure that certain folders will be used only for valid test cases (i.e. those that are not meant to result in an error), add the CharacterInstance Condition **'Instance.Character' != "Invalid**" at the folder level for the following TestStep folders:
   - “Checkout Process”
   - “Verification of prices”
   - “Confirmation”
   - “Verification of Success”

   Note: This must be done manually and cannot be dragged and dropped like the other Conditions we have used previously. ☐

3. Add the CharacterInstance Condition **'Instance.Character' == "Invalid**" for the “**Empty Shopping Cart**” TestSteps Folder to make sure the cart is emptied after the wrong code is entered. ☐

4. Add the necessary **Conditions** to the Template in TestStep folder “**Verification of Prices”>>TestStep “Verification of Prices” >>Cart total** according to the table below: ☐

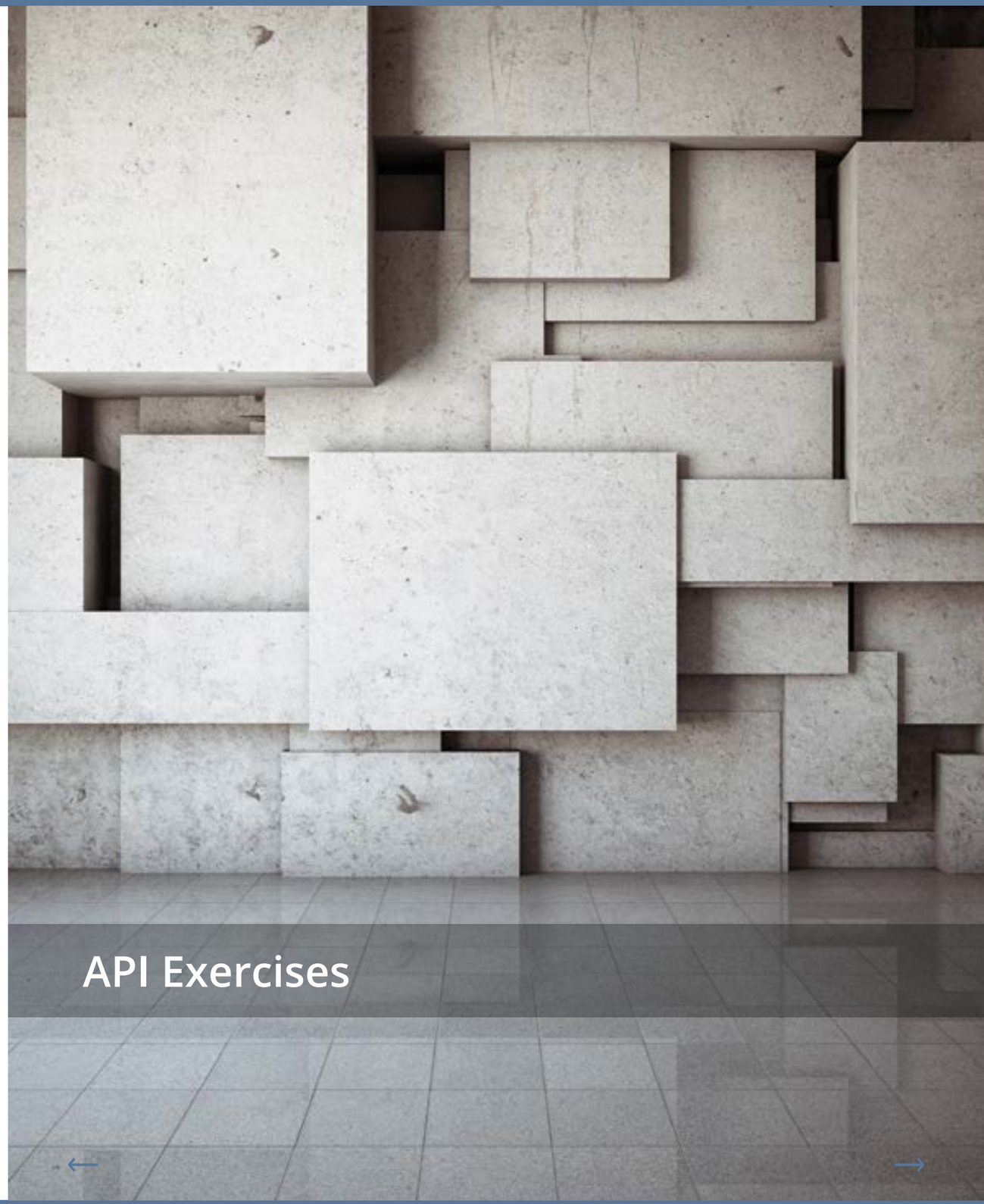
TestStepValue	Value	Condition
Shipping:*	10.00	‘Workflow.Discount Assignment' != "Discount Assignment Shipping”
Shipping:*	{MATH[(1-{XL[Verifications.Value - Discount %]})*10]}	‘Workflow.Discount Assignment’ == “Discount Assignment Shipping”
Discount:	-{MATH[({B[SubTotal]}+{B[Shipping]})*{XL[Verifications.Value - Discount %]}]}	'Workflow.Discount Type' == "Discount Type Percentage" AND 'Workflow. Discount Assignment' == "Discount Assignment Total"
Discount:	-{XL[Verifications.Value - Discount Flat Rate]}	'Workflow.Discount Type' == "Discount Type Flat""
Discount:	-{MATH[{XL[Verifications.Value - Discount %]}*{B[SubTotal]}]}	'Workflow.Discount Assignment' == “Discount Assignment Sub-Total”

TestStepValue	Value	Condition
Discount:	Discount Buffer	'Workflow.Discount Assignment' != "Discount Assignment Shipping"
Total:	{MATH[{B[SubTotal]}+{B[Shipping]}+{B[Discount]}]}	Workflow.Discount Assignment' != "Discount Assignment Shipping"
Total:	{MATH[{B[SubTotal]}+{B[Shipping]}]}	Workflow.Discount Assignment' == "Discount Assignment Shipping"

5. Use the **Check Template** command to check for errors. ☐
6. **Instantiate** the Template to create the TestCases from the TestSheet. ☐
7. Link the TemplateInstance Folder "**TemplateInstance of Discount Code**" to the Requirement DemoWebShop >> WebShop Frontend >> Shopping cart >> "**Discounts**". ☐
8. Create an ExecutionList in the "**ExecutionLists**" folder and name it "**7 Discount Code**". Link the "**TemplateInstance of Discount Code**" folder to the ExecutionList. ☐
9. Run the ExecutionList. ☐

Hints

- Drag and Drop the necessary Instances onto the Condition fields before adjusting the Conditions.
- Single quotation marks (') are must be used when writing the name of the Attribute in the Condition column when there are spaces in the Attribute name. If there are no spaces in the Attribute name, they may be omitted.



API

Exercise 8a – API TestCases

Objective

By the end of this exercise, you will be able to create an automated TestCase from an existing API Scan export.

Why is this important?

Being able to use Tosca to automate API tests will help you test functions not visible in the user interface as well as reduce testing time by testing earlier and faster.

Project perspective

Your task now is to test the products in the DemoWebShop, which were changed recently. The SKU of some products are now different. You want to verify if the product BlueJeans still has the same SKU or ID. Since you cannot see the SKU of a product on the UI of the DemoWebShop, you need to use an API TestCase. The web developers have provided you with the API Scan data as a subset.

Instructions

1. In the TestCase section, create a new folder under the root folder and name it "API". ☐
2. In the "**API**" folder, create a new folder and name it "**8a API TestCase**". ☐
3. Create a TestCase in this folder and name it "**Verify Product by SKU**". ☐
4. Add the Module "**GetProductBySku**" and "**GetProductBySku Response**" to the TestCase. ☐
5. Add the following Values as the example shows below: ☐

Name	Value	ActionMode	DataType
Verify Product by SKU			
GetProductBySku			String
sku	BlueJeans	Insert	String
usernameOrEmail	{CP[Email]}	Insert	String
userPassword	{CP[Password]}	Insert	String
GetProductBySku Response			
Id	36	Verify	String
Name	Blue Jeans	Verify	String

7. Set the TestCase **Workstate** to "**Completed**". ☐
8. Run the TestCase in the **Scratchbook**. ☐

Hints

» Make sure to use the correct ActionMode to verify the Response values.

Exercise 8b – Convert to Template, Link Values and Instantiate

Objective

By the end of this exercise, you will be able to create API TestCases, using Values from a TestSheet.

Why is this important?

An API TestCase works just like a normal TestCase in Tosca, which means you can convert it to a Template, link a TestSheet and Instantiate. In the case of UI testing, this means easier execution and faster maintenance.

Project perspective

After creating the API TestCase to verify a product by its SKU, you will also need to link it to a TestSheet to be able to check the SKU of, not only Blue Jeans, but other products as well.

Instructions

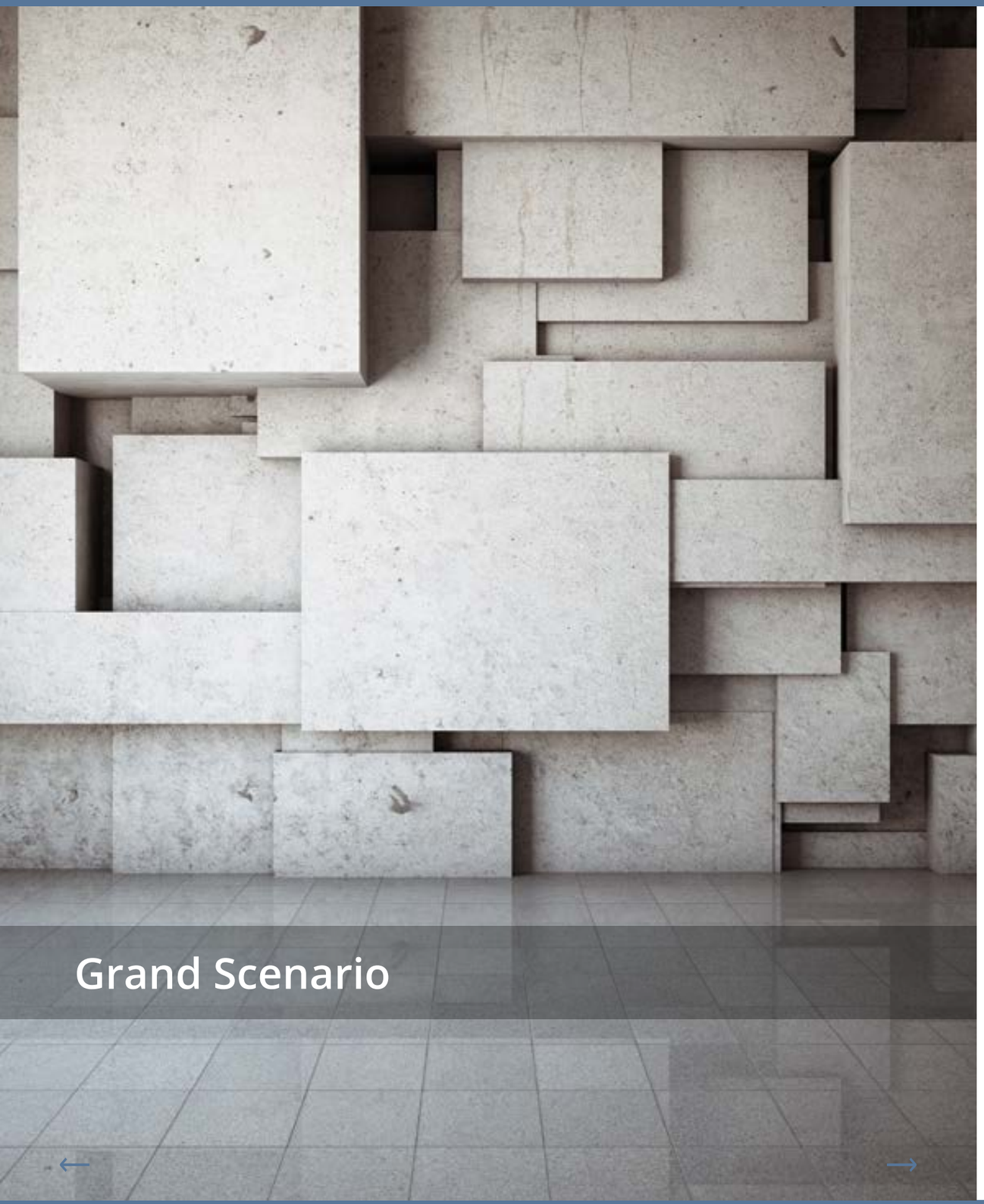
1. Duplicate the folder "**8a API TestCase**" and rename it "**8b Convert to Template**, **Link Values and Instantiate**". ☐
2. Convert the "**Verify Product by SKU**" TestCase to a Template. ☐
3. Link the TestSheet "**Verify Product by SKU**" in the folder "**8 API**" in the TestCaseDesign section to the Template. ☐
4. Link the Attributes from the TestSheet "**Verify Product by SKU**" to the TestStepValues, as the example shows below. ☐

Name	Value	ActionMode	DataType
Verify Product by SKU			
GetProductBySku			
sku	{XL[sku]}	Insert	String
usernameOrEmail	{XL[Email]}	Insert	String
userPassword	{XL[Password]}	Insert	String
GetProductBySku Response			
Id	{XL[Id]}	Verify	String
Name	{XL[Name]}	Verify	String

5. If there is any extra Value showing up next to the newly linked XL paths, delete the extra Values. ☐
6. Instantiate the **Template**. ☐
7. Run the TemplateInstances in the **Scratchbook**. ☐

Hints

» Before you drag and drop Values from the TestSheet to the Template, make sure to delete the existing Values in the TestStepValues.



Grand Scenario: Order Your Own Computer

Objective:

To build your own data-driven test to automate the ordering process of a customized computer from the DemoWebShop.

Why this is Important?

This Grand Scenario will give you the opportunity to practice the concepts learnt during the training and to create a more complex Template. This environment and set of instructions mimic more closely the project environment and tasks distribution.

Business context:

Products have many different features, and it is important that customers are able to choose according to their preference. An example can be found in clothing, where you can choose the desired size, color, fabric etc. The same is true for technology items, like computers.
In our DemoWebShop, we will need to test if products can be configured according to their available features. Second, we want to test if they are priced correctly according to those selected features.
In this case, we will select different features of a customized computer, each with a defined price, order it and then verify that the price is calculated correctly. For this, we will need to build a Template using the data given in the Product Configuration TestSheet.

Available starter packages:

We will use the DemoWebShop to order a desktop computer.

There are 2 types of desktop computers available:

- **Cheap computer**: this is the standard choice. Base price is USD 800.00.
- **Expensive computer**: Base price is USD 1,800.00.

Each computer can be customized with various features and configurations.

Available configurations:

Processor

There are three types of processors available:

- **Medium speed** – this is the standard choice. Additional charge is USD 15.00.
- **Fast speed** – additional charge is USD 100.00.
- **Slow speed** – no additional charge.

RAM

There are three types of RAM sizes available:

- **2 GB** – no additional charge.
- **4 GB** – this is the standard choice. Additional charge is USD 20.00.
- **8 GB** – additional charge is USD 60.00.

Hard Disk Drive

There are two types of Hard Disk Drive sizes available:

- **320 GB** – this is the standard choice. No additional charge.
- **400 GB** – additional charge is USD 100.00.

Software

The standard choice of software is Office Suite. Additional software is also available:

- **An Image Viewer** – additional charge is USD 5.00.
- **An Office Suite** – additional charge is USD 100.00.
- **Another Office Suite** – additional charge is USD 40.00.

Multiple software can be combined, but only one Office Suite can be bought with the same system.

The orders will have to be payed via COD (Cash on Delivery).

Instructions

The Modules, Reusable TestStepBlocks and TestSheet are available in the base subset.

TestCase Overview:

The test automation should go through the usual process:

- Open browser and navigate to the DemoWebShop
- Log in
- Navigate to Computers, select Computer Category, select Desktop
- In the Desktop section, select either “Build your own cheap computer” or “Build your own expensive computer”
- Depending on the TestSheet, customize the computer
- Checkout procedure (add to cart, select payment method, pay)
- Log out and close the browser

During the “**Checkout Process**”, we will have also to verify:
1) if the Sub Total price matches the Product price on the TestSheet and
2) if the Total price calculated by Tosca matches the Total price provided by the SUT.

Finally, the automation process should confirm the order, verify it, log out of the account and close the browser.

In the case when both Office Suites are selected, the TestCase will skip the Check out process and log out immediately, .

The Modules:

All Modules can be found under: Web Shop>> Products >> **Products Category Computers** Module Folder.

Hints

» You are free to choose the delivery method by yourself. Keep in mind that the appropriate additional fee will need to be added(if there are any) to the Total price calculation.

» The structure of this TestCase is somewhat similar to the Discount Code TestCase. Remember to include the surcharge from the payment method (COD) when calculating the total price.

» **Precondition Attributes**: The 2 Attributes you need to pay attention to for this Grand Scenario are:
- Precondition >> Customer >> Type of User >> Email
- Precondition >> Customer >> Type of User >> Password

» These 2 Attributes use data from the Test Configuration Parameters **Email** and **Password**.

» **Verification**: the Attribute Verification >> **Product Price** is the Sub-Total price of the product.