Tricentis

Automation Specialist Level 1

STUDENT EXERCISE WORKBOOK

SURVEY

Tricentis Automation Specialist | Level 1

Student Exercise Workbook

- Version 2019_3
- 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 1 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 1**.

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.

GETTING STARTED

GETTING STARTED

Introduction - System Under Test (SUT)

Objective

By the end of this exercise, you will be able to identify the main steps of the order process within the sample Web Shop application.

Why is this important?

The scenario for this training is that we are responsible for testing the Web Shop, our SUT. To understand how to test the SUT we must have a good understanding of how it works.

Instructions

1. Open the Tricentis Demo Web Shop within Internet Explorer: **http://demowebshop.tricentis.com/** ☐

2. Register in the Web Shop using the following information: ☐

   User data:

• Gender:	< your gender >
• First name:	< your first name >
• Last name:	< your last name >
• E-mail address:	Your email, replace the .com, .at, etc. with .test
• Password:	Tosca1234!
• Confirm password:	Tosca1234!

Example Email address:	Firstname.lastname@yourdomain.test

3. **User address:** ☐

• First name:	< your first name >
• Last name:	< your last name >
• E-mail address:	Email address used above
• Country:	Austria
• City:	Vienna
• Address 1:	Vienna street
• Zip:	1234
• Phone:	00 11 22 33 44 55

4. Navigate through the Web Shop by purchasing some items. Please use the following credit card details. ☐

   Credit card details:

• Select Credit Card:	Visa
• Cardholder name:	Barbara Gordon
• Card number:	4485564059489345
• Expiration date Month:	04
• Expiration date Year:	2022
• Card code:	123

Hints

» Making a rough flow chart of the steps taken through the SUT will help you to understand the structure of the TestCases to come.

Introduction - Tricentis Tosca

Objective

By the end of this exercise, you will be able to create a single User workspace and import a Subset into a workspace.

Why is this important?

Each project will have its own workspace. Subsets allow you to import and export parts of a project, for example the Tricentis Standard Modules.

Instructions

1. Download the Base Subset: **AutomationSpecialistI_BaseSubset.tce** from the Learning Management System. ☐
2. Create a new **single user workspace** using the Base Subset as your template. Name this workspace "**AS1 Training**". ☐

TESTCASE ONE

Exercise 1a - Create a Module Identify by Property

Objective

By the end of this exercise, you will be able to use XScan to create a Module and use the technical properties of the Module to uniquely identify it.

Why is this important?

Modules store the technical information and therefore are the foundation of all test automation in Tosca.

Key elements:

Module Folder Modules XScan Controls

Instructions

1.	Navigate to the **Modules** section.	☐
2.	Navigate to the Module Folder: **Webshop>>Customer>>1a Log in Page**	☐
3.	Open the **Web Shop** in **Internet Explorer** to the "**User Log in**" page. *Remember: Tricentis Tosca can be used with a variety of Browsers, however the training is designed for use with Internet Explorer. You may experience issues using other browsers.	☐
4.	Use **XScan** to scan the login page. Select: • "**Email**" text box • "**Password**" text box • "**Remember Me?**" checkbox • "**Forgot password?**" link • "**Log in**" button	☐
5.	Rename the **controls** to match their purpose, e.g. "**True**" to "**Remember Me?**"	☐
6.	Within XScan, rename the Module to "**Log in Page**".	☐
7.	Save the Module.	☐

Exercise 1b - Identify by Anchor and Create ControlGroup

Objective

By the end of this exercise, you will be able to use the "Identify by Anchor" method to uniquely identify a Module control and create a ControlGroup within a Module.

Why is this important?

Some controls need additional steps to uniquely identify them. Without uniquely identifiable elements, we cannot steer Tosca. ControlGroups make TestCases easier to create and maintain.

Key elements:

Identify by Anchor

ControlGroup

Instructions

1.	Navigate to the **Modules** section.	☐
2.	Navigate to Module folder: **WebShop>>Products>>1b Product Choice Tabs**.	☐
3.	Run XScan, Select: • "**BOOKS**" link • "**COMPUTERS**" link • "**ELECTRONICS**" link • "**APPAREL & SHOES**" link • "**DIGITAL DOWNLOADS**" link • "**JEWLERY**" link • "**GIFT CARDS**" link Make all controls uniquely identifiable using **identify by anchor**. Use the "**UL**" container that contains the links, and identify using the properties "**Tag**" & "**ClassName**". *Remember: use the filter to see more detail of the tree on the screen. *Remember: the anchor control must be uniquely identified for the objects underneath to be uniquely identified.	☐
4.	Within XScan, click on the **Module root** and change the property called "**Title**" using the wildcard to "**Demo Web Shop***" to ensure that the Module will function on all pages.	☐
5.	Rename the Module itself to "**Product Choice Tabs**".	☐
6.	Save and close XScan.	☐
7.	Within this Module create a **ControlGroup** of the 7 controls just scanned. Name this Control group "**Product Categories**".	☐

Hints

» Tricentis Tosca can be used with a variety of Browsers, however the training is designed for use with Internet Explorer. You may experience some issues using other browsers.

» Use "Ctrl+click" to select multiple ModuleAttributes.

» The * symbol acts as a wildcard. In this case, Tosca will perform the action on any window beginning with "Demo Web Shop" (e.g. Demo Web Shop. Books)

Exercise 2 - TestCase Structure and Predefined Test Configuration Parameter

Objective

By the end of this exercise, you will be able to create a logical TestCase structure within Tosca and set a predefined Test configuration parameter (TCP).

Why is this important?

A logical folder structure within your TestCase makes your work easier to create and maintain. Test configuration parameters are used to apply specific values to various TestCases; defined for various object types. A common TCP is to set the browser that Tosca is using to test.

Key elements:

TestCase

TestCaseFolder

Test configuration parameter

Instructions

1.	Navigate to the TestCases section.	☐
2.	Create a TestCaseFolder named "**TestCase 1**". Within this new Folder, create a Folder named "**2 TestCase Structure and TCP**".	☐
3.	Within the **2 TestCase Structure and TCP** subfolder, create a TestCase named "**Shipping Costs**".	☐
4.	Create the following **TestStepFolders** within the TestCase you just created: • Precondition • Order Product • Start Checkout • Checkout Process • Verification of Prices • Confirmation • Verification of Success • Postcondition	☐
5.	On the TestCase level create a Test configuration parameter named "**Browser**" with the value "**InternetExplorer**".	☐
6.	On the "**Details**" tab on the TestCase level, change the TestCase WorkState from "**PLANNED**" to "**IN_WORK**".	☐

Hints

» The folder structure follows the path taken through the application to order a product

» Use the shortcut "Ctrl+N, Ctrl+F" to create a new folder

» Use "Ctrl+," to create a folder on the same level

» Folder structures might change during the process of TestCase creation. Folders might need to be added or deleted – this is normal

Exercise 3a - Open and Close the Application

Objective

By the end of this exercise, you will be able to use Tricentis Standard Modules to open and close an HTML application.

Why is this important?

Opening the SUT is the recommended first step in the automation process. Closing it is the recommended final step.

Key elements:

TestStep

Instructions

1.	Duplicate TestCaseFolder "**2 TestCase Structure and TCP**" and name it "**3a Open and Close the Application**".	☐
2.	View both the **Module** section and the **TestCases** section together on your screen. Navigate to the **Modules** section.	☐
3.	Navigate to: ***Tricentis Standard Modules>>TBox XEngines>>Html>>OpenUrl***.	☐
4.	Drag the Module "**OpenUrl**" and drop it into the TestStepFolder "**Precondition**". Rename this TestStep "**Open Web Shop**".	☐
5.	Navigate to Tricentis ***Standard Modules>>TBox Automation Tools>>Basic Windows Operations>>TBox Window Operation***. Drag & drop this Module into the TestStepFolder "**Postcondition**" and rename the TestStep "**Close Web Shop**".	☐
6.	Navigate to the "**Open Web Shop**" TestStep and in the value column next to "**Url**" enter the following link: **http://demowebshop.tricentis.com**	☐
7.	Navigate to "**Close Web Shop**" TestStep and in the value column next to "**Caption**" enter the text: "**Demo***" and set the ActionMode to **Select**.	☐
8.	In the Value column next to "**Operation**", choose "**Close**" from the drop-down menu.	☐
9.	Ensure that you have closed all DemoWebShop browser windows then right click on the TestCase and select "**Run in ScratchBook**", then save your work.	☐

Hints

» Use the window docking function to customize your workspace.

← →

Exercise 3b - Create TestSteps

Objective

By the end of this exercise, you will be able to fill the TestCase with the necessary TestSteps to automate the Web Shop.

Why is this important?

Modules create the TestSteps for an automated TestCase.

Instructions

1.	Duplicate TestCaseFolder "**3a Open and Close the Application**" and name it "**3b Create TestSteps**".	☐
2.	Using the table below, add Modules from the Modules Section into the correct TestStepFolders. You can do this using drag and drop. You can also use the "**Add TestStep**" function to find the Modules: Click on the TestStepFolder where you wish to add the Module. Use the "**Ctrl+T**" hotkeys and start typing the Module name. A list of Modules which match the search criteria will appear. Select the Module you wish to use by clicking on it and rename it according to the table.	☐

Path to Module	Module Name	TestStepFolder to add Module	Rename the TestStep
Created in Ex. 3a	OpenUrl	Precondition	Open Web Shop
Webshop >> Navigation elements on all pages	Top Menu	Precondition	Navigate to Log in Page
Webshop >> Customer >> 1a Log in Page	Log in Page	Precondition	Log in
WebShop >> Products >> 1b Product Choice Tabs	Product Choice tabs	Order Product	Navigate to Apparel and Shoes
Webshop >> Products	Apparel & Shoes Product Selection	Order Product	Navigate to Blue Jeans
Webshop >> Products >> Products Category Apparel and Shoes	Blue Jeans	Order Product	Order Blue Jeans
Webshop >> Navigation elements on all pages	Top Menu	Start Checkout	Navigate to Shopping Cart
Webshop >> Check out process	Shopping Cart	Start Checkout	Shopping Cart Procedures
Webshop >> Check out process	Billing Address	Checkout Process	Billing Address Continue
Webshop >> Check out process	Shipping Address	Checkout Process	Shipping Address Continue
Webshop >> Check out process	Shipping Method	Checkout Process	Shipping Method Ground
Webshop >> Check out process	Payment Method	Checkout Process	Payment Method Credit Card
Webshop >> Check out process	Payment Information Credit Card	Checkout Process	Payment Information Credit Card
Webshop >> Check out process	Confirm Order	Verification of Prices	Verification of Prices
Webshop >> Check out process	Confirm Order	Confirmation	Confirm the Order
Webshop >> Order confirmation and details	Order Successful	Verification of Success	Verify the Order Success
Webshop >> Order confirmation and details	Order Successful	Postcondition	Continue
Webshop >> Navigation elements on all pages	Top Menu	Postcondition	Log Out
Created in Ex 3a	TBox Window Operation	Postcondition	Close Web Shop

← →

Exercise 4a - TestStepValues

Objective

By the end of this exercise, you will be able to enter test data into the TestSteps to steer the SUT.

Why is this important?

At this point the information for the TestCase must be entered: what to click, enter etc. into the SUT. Entering this information allows Tosca to steer and automate the test process.

Instructions

1. Duplicate TestCaseFolder "**3b Create TestSteps**" and rename it "**4a TestStepValues**". ☐
2. Enter the following Values for the TestStepValues as per the table below: ☐

TestStep Name	TestStepValue	Value
TestStepFolder ***Precondition***		
Open Web Shop	Url	http://demowebshop.tricentis.com
Navigate to Log in Page	Log in	X
Log in	Email	Registered Email Address
	Password	Registered Password DataType: Password
	Log in	X
TestStepFolder ***Order Product***		
Navigate to Apparel and Shoes	Product Categories	Apparel & Shoes
Navigate to Blue Jeans	Blue Jeans	X
Order Blue Jeans	Quantity	25
	Add to cart	X
TestStepFolder ***Start Checkout***		
Navigate to Shopping Cart	Shopping cart	X
Shopping Cart Procedures	Terms of Service	True
	Checkout	X
TestStepFolder ***Checkout Process***		
Billing Address Continue	Continue	X
Shipping Address Continue	Continue	X
Shipping Method Ground	Shipping Methods	Ground
	Continue	X
Payment Method Credit Card	Payment Methods	Credit card
	Continue	X
Payment Information Credit Card	Select credit card	Visa
	Cardholder name	Barbara Gordon
	Card number	4485564059489345
	Expiration date | Month	04
	Expiration date | Year	2022
	Card code	123
	Continue	X
TestStepFolder ***Verification of Prices***		
Verification of Prices	**LATER EXERCISE**	
TestStepFolder ***Confirmation***		
Confirm the Order	Confirm	X
TestStepFolder ***Verification of Success***		
Verify the Order Success	**LATER EXERCISE**	
TestStepFolder ***Postcondition***		
Continue	Continue	X
Log Out	Log Out	X
Close Web Shop	Caption	Demo*
	Operation	Close

Hints

- F9 expands and contracts the detail that can be seen, as it hides all values with no actions specified.
- Each TestStep steers actions within the Web Shop. Some TestSteps require multiple actions, e.g. enter Log in details and click enter.
- Some TestStepValues have Values that can be entered via a drop-down menu.

Exercise 4b - Random Values

Objective

By the end of this exercise, you will be able to generate random text and numbers with Tosca to enter into the SUT.

Why is this important?

TestCases will be run repeatedly over long periods of time. Entering random texts means that there is no need to create text for text fields that do not need to be verified, saving time.

Instructions

1.	Duplicate TestCaseFolder "**4a TestStepValues**" and name it "**4b Random Values**".	☐
2.	Within the TestStepFolder "**Checkout Process**" navigate to TestStep "**Payment Information Credit Card**" enter the correct syntax to: • Generate a random 10 character length text in "**Cardholder name**" • Generate a random 3 digit number between 100 and 999 in the "**Card code**" field	☐

Hints

» Right click in the Value field containing the syntax and select Translate value to test if your equation generates the expected result.

Exercise 5 - Dynamic dates

Objective

By the end of this exercise, you will be able to set Tosca to automatically enter dynamic dates in the "Payment info" Credit card TestStep.

Why is this important?

TestCases will be run repeatedly over long periods of time. Setting dynamic dates means there is less maintenance due to checking/amending dates.

Instructions

1.	Duplicate TestCaseFolder "**4b Random Values**" and name it "**5 Dynamic Dates**".	☐
2.	Within the TestStep "**Payment information Credit Card**" enter the correct syntax to: • Generate a dynamic date in the **Expiration date | Month** field. The date should be **plus 4 months**, displayed as only a **2-digit month** • Generate a dynamic date in the **Expiration date | Year** field. The date should be **plus 3 years**, displayed as only a **4-digit year**	☐

Hints

» Pay close attention to closing any bracket that you open.

Exercise 6a - Buffer

Objective

By the end of this exercise, you will be able to set a Buffer to save information within Tosca.

Why is this important?

The Buffer allows you to temporarily store information in Tosca, which can used later on in the TestCase to verify values, identify Modules etc.

Instructions

1.	Duplicate TestCaseFolder “**5 Dynamic Values**” and name it “**6a Buffer**”.	☐
2.	Navigate to the TestStep “**Order Blue Jeans**“ in the “**Order Product**” Folder.	☐
3.	Buffer the control “**Price**” and name the Buffer value ”**PriceBlueJeans**”.	☐

Hints

» The color of the ActionMode field changes depending on the ActionMode selected, giving a visual confirmation that the correct ActionMode has been selected.

» Clear naming for a Buffer is important. When you use the Buffer value, you must duplicate the name exactly.

Exercise 6b - Verify

Objective

By the end of this exercise, you will be able to verify values generated in the SUT.

Why is this important?

In automated testing, it is important to check that the correct values are generated. The main goal in this TestCase is to verify that the shipping costs according to your order are calculated accurately.

Instructions

1.	Duplicate TestCaseFolder “**6a Buffer**” and name it “**6b Verify**“.	☐
2.	Expand the TestStep “**Verification of Prices**” within the Folder “**Verification of prices**”.	☐
3.	Verify that the shipping cost (when “**Ground**” is selected as a delivery method) is **10.00**.	☐
4.	Run the TestCase in the **ScratchBook**.	☐

Hints

» The shipping costs are in column 2 and the shipping costs cell will be in the drop down list called “Shipping:*”

Exercise 6c - Verify Calculated Value

Objective

By the end of this exercise, you will be able to verify a value in the SUT using a Buffer and verify the appearance of an object in response to an action in the SUT.

Why is this important?

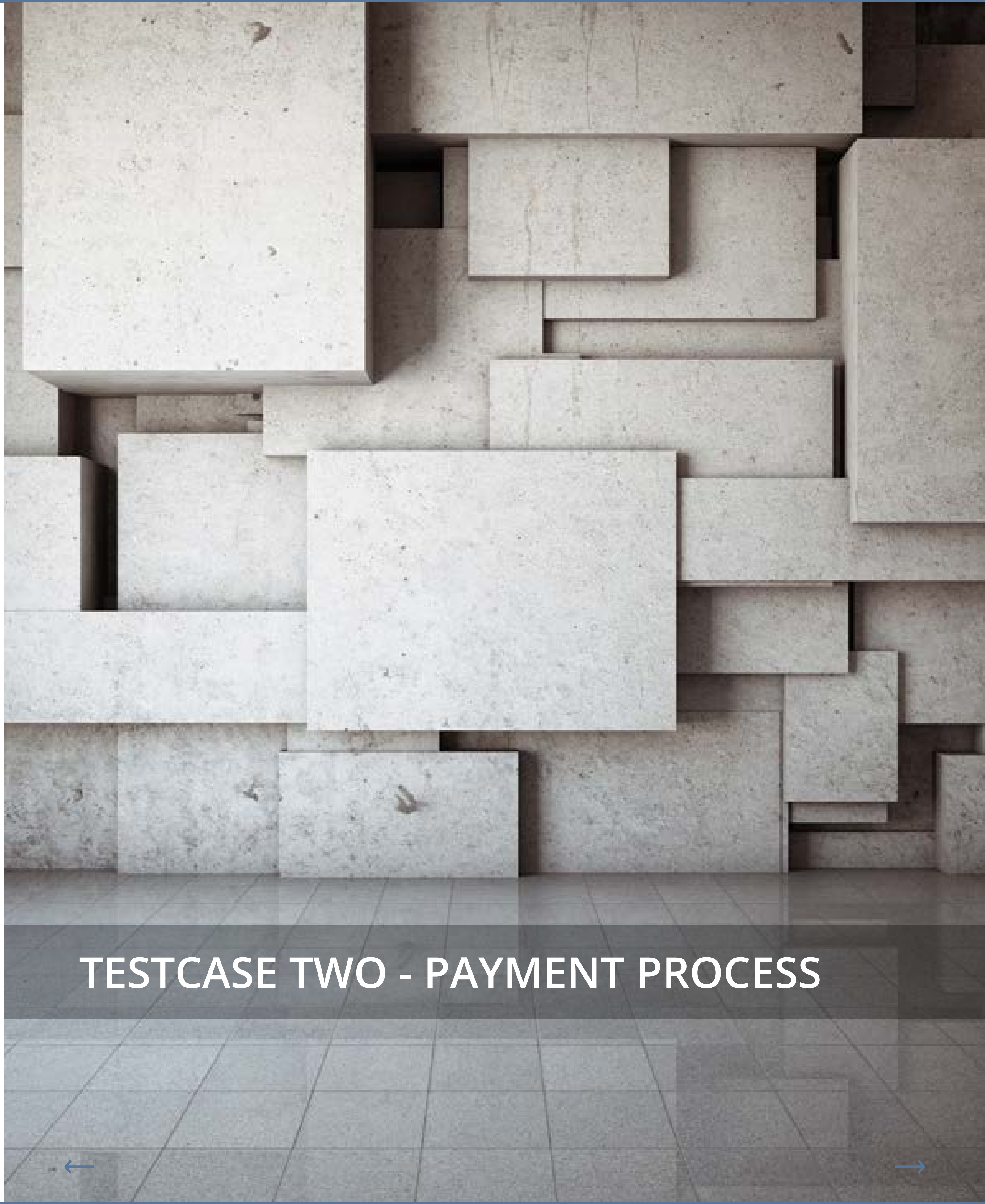
In order to verify the Sub-Total, the previously saved Buffer must be used along with calculations using an on screen message. These steps allow the TestCase to meet all the verification requirements.

Instructions

1.	Duplicate TestCaseFolder "**6b Verify**" and name it "**6c Verify Calculated Value**".	☐
2.	Expand the TestStep "**Verification of Prices**" in the Folder "**Verification of Prices**".	☐
3.	Verify the order "**Sub-Total**" using: • The Buffer "**PriceBlueJeans**" • The MATH function to multiply the Buffer by the quantity of blue jeans ordered (in this case **25**)	☐
4.	Buffer the Sub-Total using the Buffer name "**SubTotal**".	☐
5.	Verify the order "**Total**" using the MATH function. This requires adding the "**SubTotal**" Buffer to the cost of the shipping (**10.00**)	☐
6.	Expand the TestStep "**Verify the Order Success**" in the "**Verification of Success**" Folder. Verify that the "**Message Order successful**" message is visible.	☐
7.	Run the TestCase in the **ScratchBook**.	☐
8.	Set the TestCase WorkState to "**COMPLETED**".	☐

Hints

» The shipping costs are in column 2 and the shipping costs cell will be in the drop down list called "Shipping:*"
» Depending on the SUT, you may need to change the DataType to numeric.
» Make sure to check and adjust your Tosca number formats in Settings>>TBox>>Number formats if the verification fails.
» The Buffer name must exactly match the name used when it was created.



TESTCASE TWO - PAYMENT PROCESS

TESTCASE TWO

Exercise 7a – Create a Library

Objective

By the end of this exercise, you will be able to create a Library of Reusable TestStepBlocks.

Why is this important?

A Library greatly reduces the amount of work needed to create the multiple TestCases that are needed to fully automate TestCases in the SUT.

Key elements:

Library | Reusable TestStepBlock | TestCase Reference

Instructions

1.	On the TestCases **Root Folder**, create a **TestStepLibrary**.	☐
2.	Drag and drop all the Folders from the TestCase in Ex. 6c into the **Library Folder**, except: "**Verification of Prices**".	☐

Hints

» Library folders are sorted alphabetically.

← →

Exercise 7b - Create Reusable TestStepBlocks

Objective

By the end of this exercise, you will be able to use the Reusable TestStepBlocks to create References in a TestCase.

Why is this important?

This exercise will demonstrate how it is much more efficient to use a Library to create automated TestCases.

Instructions

1.	On the TestCases **Root Folder**, create a new TestCaseFolder named "**TestCase 2**".	☐
2.	Within your new Folder, create a subfolder named "**7b Create Reusable TestStepBlocks**". Create a new TestCase named "**Payment Process**". Add the **Test configuration parameter** named "**Browser**" with the value "**InternetExplorer**".	☐
3.	Drag the following Folders from the **Library**, and drop them into the TestCase: • Precondition • Order Product • Start Checkout • Confirmation • Verification of Success • Postcondition	☐
4.	Create two new **TestStepFolders** and name them: • Checkout Process • Verification of Prices Reorder the Folders to follow the Web Shop process flow (see TestCase 1)	☐
5.	Navigate to (**Webshop>>Check out process**) the **Module Folder** named "**7b Payment Information Check Money Order**"	☐
6.	Within this Folder, create a new **Module** of the Web Shop "**Payment Information**" page that appears when you select "**Check Money Order**" and click "**Continue**" Scan this page and add the controls: • "**Message**" table • "**Back**" button • "**Continue**" button Make sure the controls are uniquely identified. Name the Module "**Payment Information Check Money Order**" and save.	☐
7.	Add the following Modules into the "**Checkout Process**" TestStepFolder and rename them accordingly: • Billing Address • Shipping Address • Shipping Method • Payment Method • Payment Information Check Money Order	☐
8.	Add the "**Confirm Order**" Module into the "**Verification of Prices**" TestStepFolder and rename the TestStep "**Verification of Prices**".	☐
9.	Enter the following Values for the new TestStepValues as per the following table:	☐

← →

TestStep	TestStepValue	Value	ActionMode
TestStepFolder ***Checkout Process***			
Billing Address	Continue	X	Input
Shipping Address	Continue	X	Input
Shipping Method	Shipping Methods	Ground	Input
	Continue	X	Input
Payment Method	Payment Methods	Check / Money Order	Input
	Continue	X	Input
Payment Information Check Money Order	Continue	X	Input
TestStepFolder ***Verification of Prices***			
Verification of Prices	Cart Total		Select
	Column: Rename to #2		Select
	Cell: Choose Sub-Total	{MATH[{B[PriceBlueJeans]}*25]}	Verify DataType: Numeric
	Cell: Choose Sub-Total	SubTotal	Buffer
	Cell: Shipping:*	10.00	Verify DataType: Numeric
	Cell: Payment method additional fee:	5.00	Verify DataType: Numeric
	Cell: Total	Use the SubTotal Buffer to verify the total cost adding any shipping / additional fees.	Verify DataType: Numeric

10. Run the TestCase in the **ScratchBook**. ☐
11. Set the TestCase WorkState to "**COMPLETED**". ☐

TESTCASE THREE - DISCOUNT CODE

TESTCASE THREE

Exercise 8a - Rescan, Value Range and Module Merge

Objective

By the end of this exercise, you will be able to use XScan to rescan a Module and then to merge two Modules to avoid duplication.

Why is this important?

Modules can be amended; they do not have to be recreated every time you wish to change them. Modules can be reused any number of times and are not specific to the TestStep. Modules are technical representations of the SUT in Tosca, so we should not have multiple Modules for the same controls. Module Merge helps solve this duplication problem.

Instructions

1.	In the **Web Shop**, add items and navigate to the "**Shopping cart**" page; apply a discount code named: • **AutomationDiscount2** Leave the page open as we will be scanning it momentarily.	☐
2.	Navigate to **Webshop>>Check out process>>8a Rescan Shopping Cart** in the Modules section.	☐
3.	Right click and select Rescan the "**Rescan Cart**" Module which is in the **8a Rescan Shopping Cart** Folder.	☐
4.	Ensure the following **controls** are selected, in addition to those which are already in the Module:	☐

Element	Type	Rename to
Update shopping cart	Button	
discountcouponcode	Textbox	Enter discount code
Apply coupon	Button	
Coupon applied message	DIV	Coupon applied message
giftcardcouponcode	Textbox	Gift card coupon code
Shopping cart cost total	Table	Shopping cart costs

5.	Add a value range of possible codes to the discount code in the Module you just rescanned. The range should include the following values: • **AutomationDiscount2** • **PercentageTotal** • **PercentageShipping** • **FlatTotal** Make sure to separate the values by semicolons.	☐
6.	Expand all the Folders in the **Check out process** Module Folder. You will notice that there are currently two **Shopping Cart** Modules (one called "**Shopping Cart**" and one called "**Rescan Cart**" in the **8a Rescan Cart** Module Folder). Use "**Ctrl+Click**" to select both Modules simultaneously.	☐
7.	In the top ribbon, on the "**Modules**" tab, select **Merge Modules**.	☐
8.	The **Target Module** is the one we will keep, and we need to select the **Attributes** we want to bring from the Source Module into the Target Module. Ensure that the **Rescan Cart** Module (the one we just scanned) is listed as the **Target Module**. If it is not, choose the option to **Switch Modules**.	☐
9.	Because of their identical properties, Tosca should automatically link the Attributes which exist in both Modules.	☐
10.	Select the **Target Module** and **Source Module**, then select the Module Merge option **Enable**. When you are ready, select "**Merge**". Tosca will let you know how many Attributes have been updated, which usages have been relinked to the new Module, and deletes the **Source Module**. Select "**Close**". Rename the Module from "**Rescan Cart**" back to "**Shopping Cart**" and save your work.	☐

Hints

» To identify tables, use Highlight on Screen to locate the correct one.

» AutomationDiscount2 will apply a discount of 20% to the total value of the order.

Exercise 8b - Create the TestCase

Objective

By the end of this exercise, you will be able to use the Reusable TestStepBlocks to create References in a TestCase.

Why is this important?

The creation of logical and easy to follow structures within automated TestCases makes maintenance and problem solving easier.

Instructions

1.	On the TestCase Root Folder, create a new TestCaseFolder named "**TestCase 3**". Create a subfolder named "**8b Create the TestCase**".	☐
2.	Within the subfolder, create a new TestCase, name it "**Discount Code**". Add the Test configuration parameter: "**Browser**" with the value "**InternetExplorer**".	☐
3.	Add the following **Reusable TestStepBlocks** from the Library into the new TestCase: • Precondition • Order Product • Checkout Process • Confirmation • Verification of Success • Postcondition	☐
4.	Create two new **TestStepFolders** named: • Start Checkout • Verification of Prices Reorder the TestStepFolders to match the process flow of the **Web Shop**	☐
5.	Add the following Modules into the "**Start Checkout**" Folder: • "**Top Menu**" – rename the TestStep "**Navigate to Shopping Cart**" • "**Shopping Cart**" – rename the TestStep "**Shopping Cart Procedures**"	☐
6.	Add the following Module into the "**Verification of Prices**" Folder: • "**Confirm Order**" – rename the TestStep "**Verification of Prices**"	☐
7.	Enter the following **Values** for the **TestStepValues** as per the table below:	☐

TestStep	TestStepValue	Value	ActionMode
*TestStepFolder **Start Checkout***			
Navigate to Shopping Cart	Shopping cart	X	Input
Shopping Cart Procedures	Enter discount code	AutomationDiscount2	Input
	Apply coupon	X	Input
	Terms of Service	True	Input
	Checkout	X	Input
*TestStepFolder **Verification of Prices***			
Verification of Prices	Cart Total		Select
	Column: Rename to #2		Select
	Cell: Choose Sub-Total	{MATH[{B[PriceBlueJeans]}*25]}	Verify
	Cell: Choose Sub-Total	SubTotal	Buffer
	Cell: Choose Shipping:*	10.00	Verify
	Cell: Choose Discount:	-{MATH[({B[SubTotal]}+10.00)*.2]}	Verify
	Cell: Choose Discount:	Discount	Buffer
	Cell: Total	Use the SubTotal Buffer to verify the total cost adding any shipping / additional fees.	Verify

8.	Run the TestCase in the **ScratchBook**	☐

Hints

» Multiply a whole number by 0.20 to calculate a 20% discount.

Exercise 8c - ActionMode WaitOn

Objective

By the end of this exercise, you will be able to create a dynamic wait on a TestStepValue using the ActionMode WaitOn.

Why is this important?

WaitOn instructs Tosca to wait for an element or action to occur before moving on to the next TestStep. This benefits the automation as it can stop possible TestCase failures in situations where the SUT does not react immediately e.g. waiting for a credit card to authorize.

Instructions

1.	Duplicate TestCaseFolder "**8b Create the TestCase**" and name it "**8c ActionMode WaitOn**".	☐
2.	Navigate to the TestStep "**Shopping Cart Procedures**" in the folder "**Start Checkout**".	☐
3.	Set the ActionMode **WaitOn** for the TestStepValue "**Coupon applied message**", instructing Tosca to wait for the message to appear before executing the next TestStep. *Remember: use the blue downward arrow to select: **Visible == True**	☐

← →

Exercise 9 - Self-Defined Test Configuration Parameter

Objective

By the end of this exercise, you will be able to set and use self-defined Test configuration parameters.

Why is this important?

Test configuration parameters allow values to be set centrally which can then be used within the TestCases. If the value needs to be changed, rather than changing each occurrence on every TestStep, the value can be amended just once in the central location.

Instructions

1.	Duplicate TestCaseFolder "**8c ActionMode WaitOn**" and rename it "**9 Self Defined TCP**".	☐
2.	Add two additional Test configuration parameters to the TestCase: • **DiscountName** – set Value to "**AutomationDiscount2**" • **DiscountPercentage** – set Value to "**0.20**"	☐
3.	In the "**Shopping Cart Procedures**" TestStep use the "**DiscountName**" TCP in the TestStepValue for "**Enter Discount Code**".	☐
4.	In the "**Verification of Prices**" TestStep, amend the **Discount** calculation to use the "**DiscountPercentage**" TCP instead of the numeric value 0.20 to calculate the discount total.	☐
5	Set the Workstate to "**COMPLETED**".	☐

Hints

» {CP[]} is the syntax required to use a TCP.

← →

Exercise 10 - Business Parameters

Objective

By the end of this exercise, you will be able to create Business Parameters in the Library for use in TestCases.

Why is this important?

Business Parameters enhance Reusable TestStepBlocks, to allow the creation of Reusable TestStepBlocks from TestSteps even when the TestStep contains values that need to be changed in different TestCases.

Key elements:

Business Parameter Container

Business Parameter

Instructions

1.	Duplicate TestCaseFolder "**9 Self Defined TCP**" and rename it "**10 Business Parameters**".	☐
2.	In the **Library**, navigate to the Folder "**Precondition**" and add three **Business Parameters**: • Url • Email • Password	☐
3.	Drag each Business Parameter and drop it into the relevant TestStepValue field to create a **{PL}** link.	☐
4.	In the **TestCase**, expand the Reusable TestStepBlock "**Precondition_Reference**" and enter the correct values for the Business Parameters:	☐

Url	http://demowebshop.tricentis.com
Email	Registered Email address
Password	Registered Password

5.	Set the WorkState to "**COMPLETED**"	☐

Hints

» Remember, all previous TestCases will need to have the relevant Business Parameters entered in order to be run successfully.

TESTCASE FOUR

Exercise 11a - Dynamic Comparison

Objective

By the end of this exercise, you will be able to set a Dynamic Comparison to exclude and buffer part of a string.

Why is this important?

{XB} allows you to verify a dynamic string by excluding the dynamic part with the additional option to Buffer the excluded value. This helps verify strings with dynamic elements as well as use the buffered elements.

Instructions

1.	Create a new TestCaseFolder, named "**TestCase 4**" • Create a subfolder named "**11a Dynamic Comparison**" • Add a TestCase named "**Reorder**" • Add the Test configuration parameter named "**Browser**" with the value "**InternetExplorer**"	☐
2.	Create the TestCase using **Reusable TestStepBlocks** or creating new Folders where necessary: • Precondition • Order Product • Start Checkout • Checkout Process • Verification of Prices (create new folder) • Confirmation • Verification of Success • Buffer Order Number (create new folder) • Previous Orders (create new folder) • Reorder (create new folder) • Start Checkout • Checkout Process • Verification of Prices (create new folder) • Confirmation • Verification of Success • Postcondition	☐
3.	To both occurrences of the "**Verification of Prices**" Folder: • Add the Module "**Confirm Order**" and rename the TestStep "**Verify Prices**"	☐
4.	To the "**Buffer Order Number**" Folder: • Add the Module "**Order Successful**"	☐
5.	To the "**Previous Orders**" Folder: • Add the Module "**Top Menu**" and rename the TestStep "**Navigate to my Account**" • Add the Module "**My Account Menu**" and rename the TestStep "**Navigate to Orders**"	☐
6.	To the "**Reorder**" Folder: • Add the Module "**Order Details**"	☐
7.	In the "**Buffer Order Number**" TestStepFolder, change the TestStepValue "**Order number**" to Verify and Buffer the Order Number from the property "**InnerText**", Syntax for XBuffer is: **{XB[]}**	☐

Exercise 11b - Parent Control and Dynamic ID

Objective

By the end of this exercise, you will be able to use XScan to create a Module with controls that have been identified by a Parent, as well as use XScan to create a Module with a Dynamic ID.

Why is this important?

If the SUT has dynamic information, additional methods are required to fully automate.

Instructions

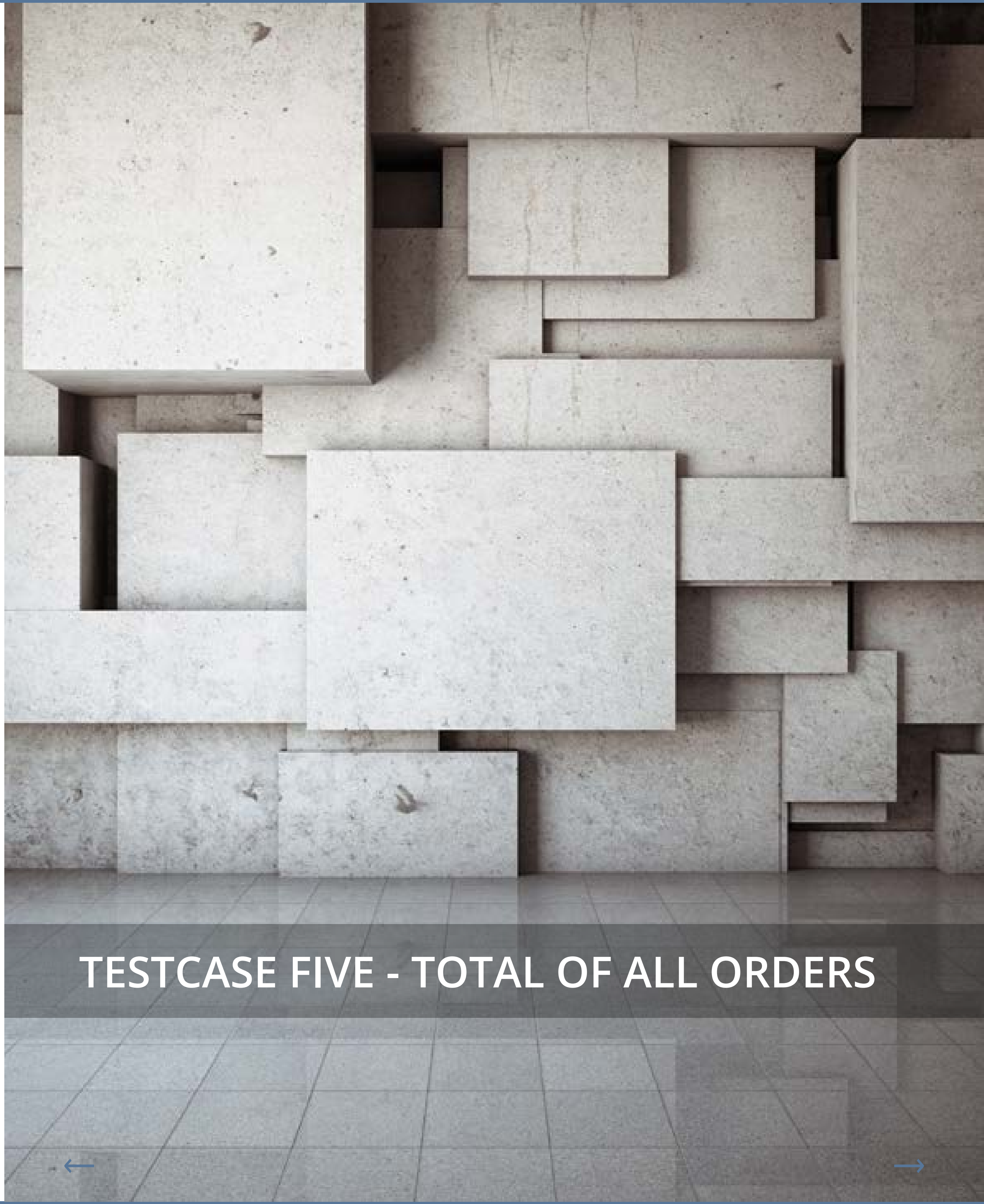
1.	Duplicate TestCaseFolder "**11a Dynamic Comparison**" and rename it "**11b Parent Control and Dynamic ID**".	☐
2.	Navigate to the Folder "**11b Orders Page**" in the Modules section (in ***Webshop>>Customer>>My account***).	☐
3.	Open the **Web Shop** application, log in, then navigate to ***My Account>>Orders***.	☐
4.	Use **XScan** to scan the **Orders** page. Make the following changes to the **Module**: • Identify the **first DIV** which contains the **order information** and the "**Details**" button • Select this DIV, the "**Details**" button and the **LI** that contains the **order total** • Rename the DIV "**OrderInfo**" and the LI "**OrderTotal**" • Uniquely identify "**OrderInfo**" by selecting the **ClassName** and the **OuterText** • Identify the "**OrderTotal**" (LI) by its **InnerText** • Rename the Module "**Order Overview**" • Save the Module *Remember - The DIV must be uniquely identifiable	☐
5	Change the property "**OuterText**" for the Attribute "**OrderInfo**" in the Module we first created to reflect the buffered order number from earlier in the TestCase. Ensure that all the other information in the outer text has been deleted and replaced with a wildcard (*****)	☐
6.	To the "**Previous Orders**" TestStepFolder: Add the "**Order Overview**" Module rename it "**Navigate to Last Order**"	☐
7.	Enter the following Values for the TestStepValues as per the table below:	☐

TestStep	TestStepValue	Value	ActionMode
*TestStepFolder **Previous Orders***			
Navigate to my Account	My account	X	Input
Navigate to Orders	Orders	X	Input
Navigate to Last Order	OrderInfo		Select
	Details	X	Input
*TestStepFolder **Reorder***			
Order Details	Re-order	X	Input
*TestStepFolder **Verification of Prices***			
Verify Prices	Cart total		Select
	Column (rename #2)		Select
	Cell (Sub-Total: from drop down)	{MATH[{B[PriceBlueJeans]}*25]}	Verify
	Cell (Sub-Total: from drop down)	SubTotal	Buffer
	Cell (Shipping:* from drop down)	10.00	Verify
	Cell (Total: from drop down)	{MATH[{B[SubTotal]}+10]}	Verify

8. Run the TestCase in the **ScratchBook**. ☐
9. Set the WorkState to "**COMPLETED**". ☐

Hints

» Remember to add the correct Values in the Business Parameters.



TESTCASE FIVE - TOTAL OF ALL ORDERS

TESTCASE FIVE

Exercise 12a - ExplicitName

Objective

By the end of this exercise, you will be able to rename the Attributes of a Module with the use of an ExplicitName Configuration Parameter and use the property ResultCount within a TestStep.

Why is this important?

Renaming TestStepValues with an ExplicitName configuration parameter allows Tosca to steer using the index, and take into account any dynamic changes.

Key elements:

Configuration Parameter

Instructions

1.	Navigate to the Module Folder "**11b Orders Page**" under **Customer>>My Account**.	☐
2.	Navigate to the properties of the "**Order Total**" Module Attribute of the "**Order Overview**" Module. Change the **InnerText** to identify any text containing "**Order Total:**" within the string.	☐
3.	Add the Configuration Parameter "**ExplicitName**" with the Value "**True**" for the control "**OrderInfo**".	☐
4.	Navigate to the TestCases section, create a Folder named "**TestCase 5**" and a subfolder named "**12a ExplicitName and ResultCount**" • Add a TestCase named "**Total of All Orders**" • Add the Test configuration parameter named "**Browser**" with the value "**InternetExplorer**" • Add the Reusable TestStepBlocks "**Precondition**" and "**Postcondition**" Resolve the reference to the Reusable TestStepBlock "**Postcondition**". Delete the TestStep "**Continue**".	☐
5	Create a TestStepFolder named "**Orders Page**" between the "**Precondition**" and "**Postcondition**" Folders. Create two TestSteps in the "**Orders Page**" Folder, using the Modules: • **Top Menu** – rename to "**Navigate to My Account**" • **My Account Menu** – rename to "**Navigate to My Orders**"	☐
6.	Enter the Business Parameters.	☐
7.	In the "**Navigate to My Account**" and "**Navigate to My Orders**" TestSteps, enter the correct Values.	☐
8.	Run the "**Precondition**" and "**Orders Page**" TestStepFolders in the **ScratchBook** to open the **Web Shop** to the correct page.	☐

Exercise 12b - TBox Set Buffer and ResultCount

Objective

By the end of this exercise, you will be able to use Tbox Set Buffer Module within a TestStep to set the Buffer "SUM" to the value "0".

Why is this important?

When using the "SUM" Buffer, ensuring that it is at "0" prior to the TestCase running will prevent errors caused by existing Buffer values from previous TestCases still being present.
ResultCount calculates how many controls with the same properties are on the page.

Instructions

1.	Duplicate TestCaseFolder "**12a ExplicitName and ResultCount**" and name it "**12b TBox Set Buffer**".	☐
2.	Create a TestStepFolder named "**Set Buffer**" after the "**Orders Page**" folder. Add the "**TBox Set Buffer**" Module (**Tricentis Standard Modules>>Tbox Automation Tools>>Buffer Operations>>TBox Set Buffer**). Rename to "**Set SUM to Zero**".	☐
3.	Change the Buffer names to: • "**SUM**" - set the Value to **0** • "**OrderNumber**" - leave the Value blank	☐
4.	Add the Module "**Order overview**" – rename "**Count Number of Orders**".	☐
5	In the "**Count Number of Orders**" TestStep, enter the correct syntax to count the number of orders (number of **OrderInfo** containers) in the Buffer "**NumberofOrders**".	☐
6.	Run the TestStepFolder "**Set Buffer**" in the **ScratchBook**. Check the buffer that the correct number of orders has been counted.	☐

Hints

» The ResultCount property indicates how many of a certain control exist.

Exercise 12c - Repetition on Folder Level

Objective

By the end of this exercise, you will be able to instruct Tosca to repeat a TestStep a certain predefined number of times.

Why is this important?

Certain processes may need actions to be repeated a certain number of times. The number of repetitions can be set within this property.

Instructions

1.	Duplicate the TestCaseFolder "**12b TBox Set Buffer**" and name it "**12c Repetition on Folder Level**".	☐
2.	Create a new TestStepFolder named "**CountUp**", between the "**Set Buffer**" and "**Postcondition**" TestStepFolders.	☐
3.	Set the "**Repetition**" Property Value to the Buffer: "**NumberofOrders**".	☐

Hints

» Repetition can be found in the Properties tab of the TestStepFolder.

Exercise 12d - ExplicitName on TestStepValue

Objective

By the end of this exercise, you will be able to change the name of a TestStepValue to steer elements by index and dynamically.

Why is this important?

When elements do not have a unique identifier. ExplicitName allows Tosca to steer using the index, and take into account any dynamic changes.

Instructions

1.	Duplicate the TestCaseFolder "**12c Repetition on Folder Leve**l" and name it "**12d ExplicitName on TestStepValue**".	☐
2.	Add the Module "**Order Overview**" into the "**CountUp**" TestStepFolder. Rename it "**Buffer Order Total**".	☐
3.	In the TestStep "**Buffer Order Total**", locate the DIV "**Order Info**". Rename it from "**Order Info**" to **#{Repetition}**. This will ensure that whichever repetition we are on, the corresponding DIV will be selected.	☐
4.	In the "**CountUp**" folder, navigate to the "**Order Total**" control.	☐
5	Use **{XB}** to verify and Buffer the price of the "**OrderTotal**".	☐
6.	Add a **TestStep** to "**CountUp**" folder by adding the "**TBox Set Buffer**" Module (**Tricentis Standard Modules>>TBox Automation Tools>>Buffer Operations>>TBox Set Buffer**). Rename the Module to "**Sum Up**". Enter syntax for the Buffer "**SUM**" Value which will add the buffered "**OrderTotal**" amount to the existing Buffer "**SUM**".	☐
7.	Log out of the Web Shop and close the Browser. Run the TestCase in the **ScratchBook**.	☐
8.	Set the WorkState to "**COMPLETED**".	☐

← → ← →

Exercise 13 - Link TestCases to Requirements

Objective

By the end of this exercise, you will be able to link TestCases to Requirements.

Why is this important?

Linking the TestCase to the Requirements supplies an accurate picture of the state of the entire project.

Key elements:

Requirement Folder	Requirement Set	Requirement	Linked TestCase

Instructions

1.	Navigate to the "**Requirements**" section, expand all of the Requirements in the Folder named "**Webshop Requirements**" (**Requirements>>Webshop Requirements**).	☐
2.	Drag and drop the TestCases with the TestCase WorkState "**COMPLETED**" to the following Requirements: • Exercise 6c "**Shipping Costs**" to the Requirement "**Calculate shipping costs**" • Exercise 7b "**Payment Process**" to the Requirement "**Payment methods**" • Exercise 10 "**Discount Code**" to the Requirement "**Discounts**" • Exercise 11b "**Reorder**" to the Requirement "**Reorder**" • Exercise 12d "**Total of All Orders**" to the Requirement "**Check orders**"	☐

Exercise 14 - Create and Execute an ExecutionList

Objective

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

Why is this important?

TestCases are actually run, or executed, in the Execution section of Tosca, via ExecutionLists.

Key elements:

Execution Folder	ExecutionList Folder	ExecutionList	Execution	Execution Passed	Execution Failed

Instructions

1.	Navigate to the ExecutionLists Folder in the **Execution** section. Create an ExecutionList Folder named "**14 Create and Execute an ExecutionList**" within the Folder named "**Automated**".	☐
2.	Create a new **ExecutionList** named "**Web Shop**".	☐
3.	Confirm that all the **TestCases** are complete and that all the **Business Parameters** are entered.	☐
4.	Drag and drop the TestCases with the TestCase WorkState "**COMPLETED**" into the ExecutionList "**Web Shop**".	☐
5	Drag and drop the **ExecutionList** onto the Requirement Set "**Webshop Frontend**".	☐
6.	Run the **ExecutionList**.	☐

Hints

» Business Parameters will have been applied to all TestCases that used the Reusable TestStepBlock "Precondition".
» If you have many orders in your overview, the final TestCase may take a couple of minutes to run.

Exercise 15 - WHILE Statement

Objective

By the end of this exercise, you will be able to implement a WHILE Statement within a TestStep.

Why is this important?

A WHILE Statement allows the TestCase to execute an action repeatedly until a condition is no longer met.

Key elements:

◇ Condition	Loop	WHILE Statement

Instructions

1.	In the Library, create a **Reusable TestStepBlock** named "**Empty Shopping Cart**". Add the following Modules: • "**Top Menu**", rename it "**Navigate to Cart**" • "**Top Menu**" again, rename it "**Log Out**" • "**TBoxWindow Operation**": **Tricentis Standard Modules>>TBox Automation Tools>>Basic Windows Operations>>TBox Window Operation**, rename it "**Close Web Shop**"	☐
2.	Within the TestStepFolder "**Empty Shopping Cart**" enter the Values in the TestSteps that will perform the actions, as per the table below:	☐

TestStep	TestStepValue	Value	ActionMode
Navigate to Cart	Shopping cart	X	Input
Log Out	Log out	X	Input
Close Web shop	Caption	Demo*	Input
	Operation	Close	Input

3.	Within the ReusableTestStep Block "**Empty Shopping Cart**" create a **WHILE** Statement between the TestSteps "**Navigate to Cart**" and "**Log Out**".	☐
4.	Within the **WHILE** Statement, add the Module "**Shopping Cart**" both in the Condition and in the Loop. Rename the TestStep in the Condition "**Verify Table Exists**", and the TestStep in the Loop, "**Empty Cart**". Add the Values to the TestSteps to complete the following actions:	☐

Condition: TestStep	TestStepValue	Value	ActionMode
Verify Table Exists	Shopping cart products table	Exists == True	Verify

TestStep	TestStepValue	Value	ActionMode
Empty Cart	Shopping cart products table		Select
	Row: $1		Select
	Cell: Remove		Select
	Remove checkbox	True	Input
	Update Shopping cart	X	Input

5.	Manually add multiple items to the Web Shop then run the **WHILE** Statement in the **ScratchBook**.	☐

Hints

» Use the Control Flow Diagram to get a visual representation of the WHILE Statement.

» A Condition verifies that something meets the set state (exists, visible, has a value of etc.)

» A Loop instructs Tosca to steer a process, which then continues until the condition is no longer met or the maximum number of repetitions is reached.

← → ← →

Exercise 16a - Recovery Scenarios

Objective

By the end of this exercise, you will be able to create and use Recovery Scenarios at the TestCase and the TestStepValue level.

Why is this important?

Recovery Scenarios are used to enable Tosca to react to certain common errors, e.g. an unexpected pop-up that appears randomly, on any page in your application, at any time during test execution.

Key elements:

Recovery Scenario Collection

Recovery Scenario

Instructions

1.	Navigate to **TestCase 1>>6c Verify Calculated Value>>Shipping Costs**. Right click on the TestCase and add a **Recovery Scenario Collection**. In the collection add a **Recovery Scenario**. Drag and drop the new Reusable TestStepBlock "**Empty Shopping Cart**" into the Recovery Scenario.	☐
2.	Set the **RetryLevel** of the Recovery Scenario to "**TestCase**".	☐
3.	Ensure that your Tosca recovery settings are correct: • **Settings>>TBox>>Recovery**. Select "**Recover**" for "**On dialog failure**", "**On exception failure**" and "**On verification failure**"	☐
4.	To see recovery work, manually open the Web Shop and log in. Add items to the shopping cart, log out and close the window. Run TestCase 1 "**Shipping Costs**" in the ExecutionList once again by right clicking on the ExecutionEntry and selecting "**Run**". The verification will fail, however, the recovery scenarios will kick in and the TestCase will be re-run successfully.	☐
5	In the Library Folder, add a Recovery Scenario to the Reusable TestStepblock "**Precondition**": • Add the Module "**Top Menu**" and rename it "**Log Out**", set the Value & ActionMode to perform a log out Change the RetryLevel of the Recovery Scenario to "**TestStepValue**". Note: We will not be able to see this Recovery Scenario kick in on Internet Explorer, because closing the browser window logs you out. However, this would be useful if using this TestCase on multiple browsers.	☐

Hints

» Recovery does not work in the Scratchbook.

← →

Exercise 16b - CleanUp Scenarios

Objective

By the end of this exercise, you will be able to create and use CleanUp Scenarios.

Why is this important?

CleanUp Scenarios kick in when a TestCase and all Recovery Scenarios fail. It will reset your system under test to a clean state and then move on to execution of the next TestCase.

Key elements:

CleanUp Scenario

Instructions

1.	Create a **Recovery Scenario Collection** in the root TestCaseFolder. Within this Folder, add a **CleanUp Scenario**.	☐
2.	In the CleanUp Scenario: • Add the Reusable TestStepBlock "**Empty Shopping Cart**". • After this, add the Module **Tricentis Standard Modules >> TBox Automation Tools >> Process Operations >> TBox Start Program**, rename it "**Close all Internet Explorer Instances**".	☐
3.	Within this new TestStep, enter the correct Values into "**Path**" and "**Arguments**" as per the table below:	☐

TestStep	TestStepValue	Value	ActionMode
Close all Internet Explorer instances	Path	taskkill	Input
	Arguments		Select
	Argument	/f	Input
	Argument	/im	Input
	Argument	iexplore.exe	Input

4.	Navigate to **TestCase 2>>7b Reusable TestStepBlocks>>Payment Process**, and make sure that the Business Parameter "**Password**" from within the "**Precondition**" TestStep is empty.	☐
5.	In the ExecutionList, rerun the 2 ExecutionEntries "**Payment Process**" and "**Discount Code**".	☐
6.	Check the executed CleanUp in the **ExecutionList** on **LogLevel**.	☐
7.	Once finished, re-enter the Business Parameter "**Password**" in the TestCase 2 "**Precondition**" TestStep.	☐

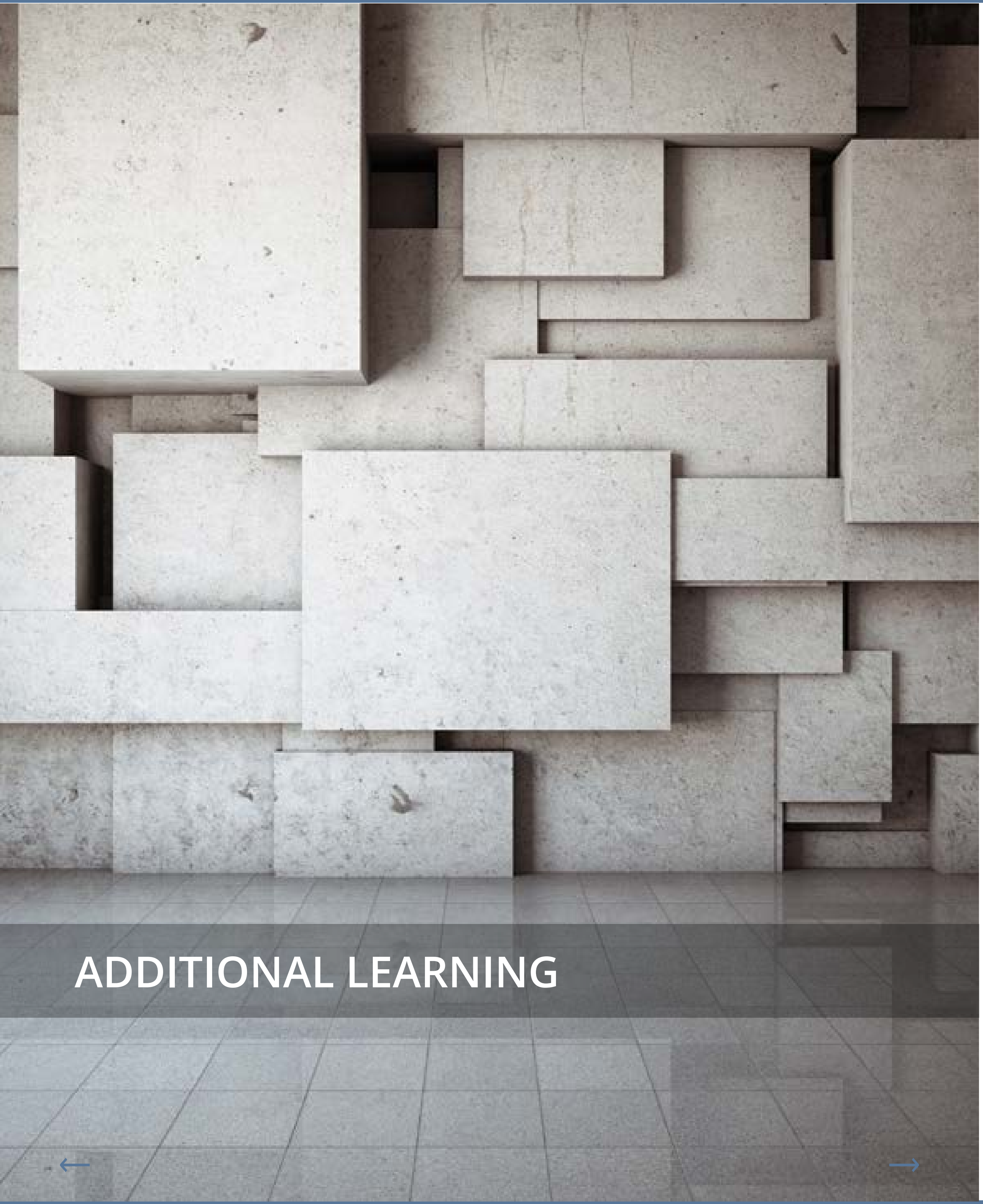
Hints

» Remember: CleanUp kicks in if no Recovery Scenario works.

» .The Argument "/f" specifies that process(es) will be forcefully terminated.

» The Argument "/im" specifies the image name of the process to be terminated.

← →



ADDITIONAL LEARNING

Exercise 17a - Constraint

Objective

By the end of this exercise, you will be able to use the ActionMode Constraint within a TestStep.

Why is this important?

ActionMode Constraint limits the search in a table to identify a specific item, regardless of where that item sits in the table.

Instructions

1.	Create a TestCaseFolder named "**Additional Exercises**". Create a subfolder named "**17a Constraint**".	☐
2.	Create a new TestCase named "**Constraint**", add the Test configuration parameter "**Browser**" with the value "**InternetExplorer**".	☐
3.	In the Web Shop, navigate to the "**Digital downloads**" section and manually add the two Music downloads named "**Music 2**" (one costs **10.00** the other is **3.00**) to the Shopping cart. Navigate to the "**Books**" section and manually add the book named "**Computing and Internet**" to the Shopping cart.	☐
4.	Into the TestCase, add the Module "**Shopping Cart**".	☐
5	Use the ActionMode **Constraint** to delete the "**Music 2**" product which costs **10.00** from the Shopping cart, leaving the other two products untouched regardless of their order.	☐

Exercise 17b - FireEvent

Objective

By the end of this exercise, you will be able to use the Steering Parameter FireEvent within a TestStep.

Why is this important?

The Steering Parameter FireEvent notifies the SUT that a change has been made to the properties.
In the following example, just entering the answer will not trigger the next step; simply entering the text does not indicate that a change has been actioned, additionally pressing enter or select is not an option. The FireEvent Steering Parameter resolves this.

Instructions

1.	Open the obstacle: http://obstaclecourse.tricentis.com/Obstacles/78264	☐
2.	Scan the obstacle course page with **XScan**, select the two labels containing the numbers and the textfield "**Result**". Rename them in a way that is logical. Save the new Module in the Module Folder "**17b FireEvent**".	☐
3.	On the Module Attribute for the Result textbox, change the value of the Steering Parameter called "**FireEvent**" from "**change**" to "**input**".	☐
4.	Create a TestCase Folder named "**17b FireEvent**" in the Additional Exercises Folder. Create a TestCase named "**FireEvent**" add the Test configuration parameter "**Browser**" with the value "**InternetExplorer**", then add the new Module.	☐
5	Use the ActionMode **Buffer** and the **MATH** function to add the two values together to enter the value into the "**Result**" textbox.	☐
6.	Run in the **ScratchBook**, When you have properly used the **FireEvent** Parameter, the obstacle course will alert you that you were successful. If you do not receive this alert, Tosca may still say the TestCase has passed but you have not completed the task.	☐

Hints

» Both the Parameter name and the value are case sensitive.