

Test Framework Opzetten

Beginnen bij 0

Linear scripten

- De opdrachten/test cases tot nu toe
- Begin bij stap 1 en eindigen bij de laatste stap
- Wellicht al opgevallen:
 - Veel code duplicatie
 - Niet erg robuust tegen applicatie veranderingen
 - Wel belangrijk om te beheersen, toch?

Een Test Framework

- Wat is jullie beeld daarbij?
- Wie heeft er ervaring mee?
- Wat is het positieve van een framework?

3



Pluspunten van een framework

- Code duplicatie zoveel mogelijk voorkomen
- Centraal code positioneren -> robuust tegen veranderingen
- Overzichtelijker
- In teamverband er aan werken/uitbreiden

4



"Minpunten" van een framework



Waar begin ik???

5



Plan maken, mindset veranderen

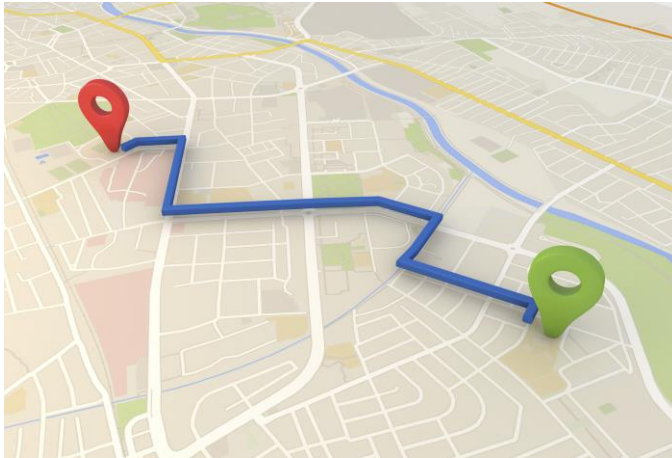


6



Mindset change

- Ipv code linear als 1 grote weg..



7



Mindset change

- .. Blokken maken die aan elkaar worden geketend



8



Mindset change

- Kijkende naar UI web automation...



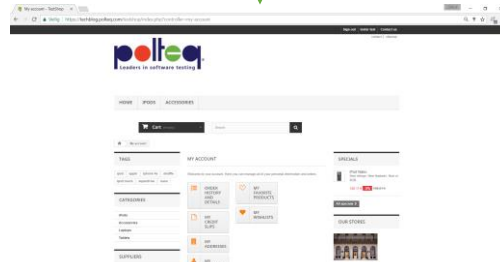
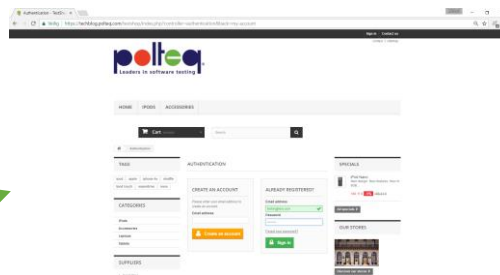
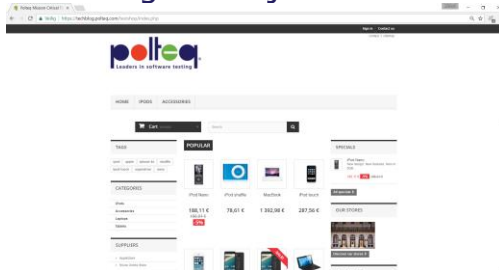
Wat zou 1 schakel kunnen zijn?



9

Mindset change

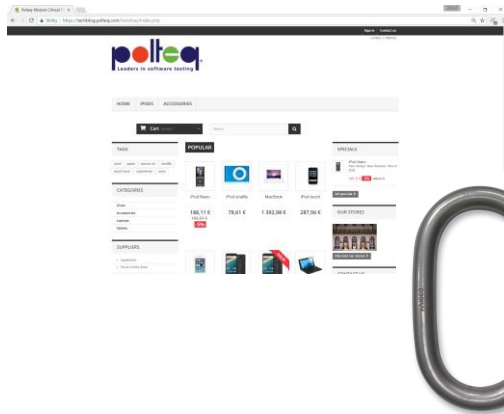
- De Pagina's zijn onze schakels!



10

Mindset change

- Code centraal voor 1 pagina
- Bevat de (nodige) verwijzingen
- Bevat de (nodige) functionaliteit



Wat zijn hier de voordelen van?

11



Hoe maak ik in Java een Schakel?

- Essentieel om de code achter de schakel te snappen
- Schakels/Blokken in Java bouwen door gebruik te van Java Classes

12



Java Class

```
Auto auto = new Auto();
```

Wait, what.....?

13



Java Class

- Een Class kan gezien worden als een blauwdruk
- In deze blauwdruk kunnen we verwijzingen stoppen en functionaliteit
- Maar in de blauwdruk roepen we de functionaliteit niet aan!

14



Blauwdruk persoon



voornaam
achternaam
snelheid
→ wandel(snelheid)

15



Blauwdruk persoon → object(en)



voornaam
achternaam
snelheid
→ wandel(snelheid)



Tinus
Tester 5.5
→ wandel(5.5)



Bianca
Bevinding 5.8
→ wandel(5.8)

16



Java Class -> Object

- Als wij de blauwdruk tot leven roepen dan maken we er een object van
- Wij kunnen meerdere objecten maken van 1 enkele blauwdruk
- Het object kan de functionaliteit in een class aanspreken en uitvoeren

```
Auto auto = new Auto();
```

- Datatype Auto maken we een object genaamd auto van de Class Auto

17



Java Class -> Object

- Abstracte uitleg

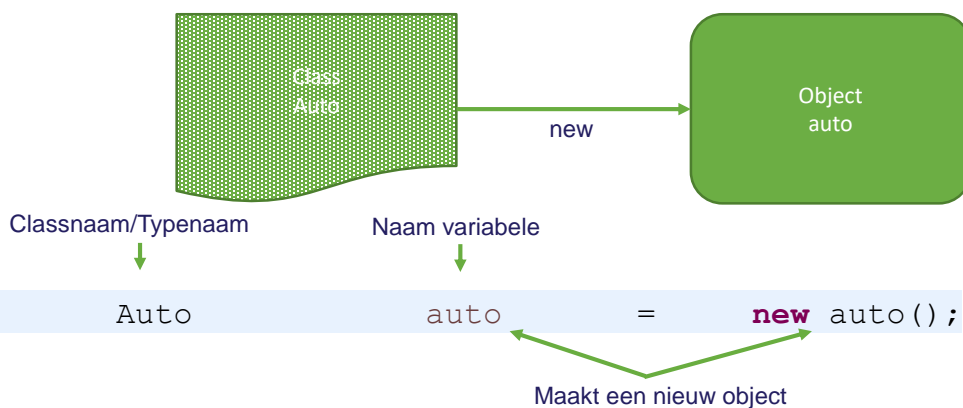
```
Auto auto = new Auto();
```

18



Een object is de concrete uitwerking

- Met een echte uitwerking van de blauwdruk, een echte auto, kun je wel func aanroepen en gebruiken.



19



Een object is de concrete uitwerking

- Met **new** maken we een nieuw object
 - Auto auto = new Auto();
 - Dit noemen we het instantiëren van een object
 - Een object is de instance van de class, dus je kunt zeggen:
 - auto is een object;
 - auto is een instance van de class Auto.
 - auto is een instance van het type Auto.
- Met het object kunnen we een methode aanroepen
 - auto.openDoor(4);

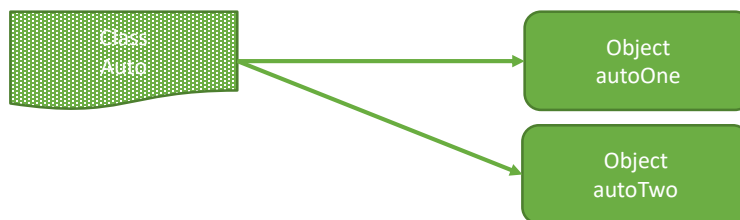
20



Een object is de concrete uitwerking

- We kunnen dus meerdere objecten (instances van een class) maken
- Meerdere auto's

```
- Auto autoOne = new Auto();
- Auto autoTwo = new Auto();
```



21



Samengevat

- Een class is een blauwdruk
 - beschrijft eigenschappen en mogelijkheden;
 - is een (reference) type.
- Een object is een echte uitwerking
 - wordt gemaakt op basis van een class;
 - wordt gemaakt met een constructor;
 - van een class kunnen meerdere objecten worden gemaakt.

22



Maar eerst....

- Maak nieuwe git branch aan en checkout
 - "git branch feature/framework"
 - "git checkout feature/framework"
 - Code away.

23



Java Class - Object

- Laten we samen beginnen met maken de Class Auto

To the code! – Auto.class

24



Opdracht

1. Breid Auto.class uit met de volgende functionaliteit;
 1. Printen merk
 2. Printen deuren
 3. Printen motor type
2. Maak een methode die de torque van de auto berekend en print naar de console. Formule:

$$\text{Torque} = \frac{\{Force \times 5252\}}{\text{engine RPMs}}$$
3. Maak een object van Auto.class in CarGame.class en roep alle functionaliteit aan van de Auto.Class

25



Constructor?

- Maar wat als ik mij auto nu meteen "iets" wil laten doen?
 - Auto heeft een merk en die verandert meestal niet (hopelijk).
 - Auto krijgt in de fabriek een motor en die verandert ook niet (again, hopelijk!)

To the code! – AutoAdvanced.class

26



Opdracht

1. Breid AutoAdvanced.class uit met de volgende functionaliteit;

1. Gebruik de torque formule om een methode te maken

$$\text{Torque} = \frac{\{Force \times 5252\}}{\text{engine RPMs}}$$

2. Deze methode wordt meteen aangeroepen in de constructor

2. Maak een object van AutoAdvanced.class in CarGame.class en roep alle functionaliteit aan van de AutoAdvanced.Class

27



Tijd voor actie -> plan maken

- 1. Welke elementen gaan we gebruiken in onze test
- 2. Locators voor elementen vinden
 - Omzetten naar code
- 3. Functionaliteit van de pagina verwoorden in een methode



28



Testcase 9.01 – Invullen contact formulier

- Opdrachten bijlage -> 9.01
- Correcte Git branch?
- Maak gebruik van het Page Object Model
- Tip: een dropdown invullen werkt iets anders dan "normaal"
- Denk aan validatie!
- WE BEGINNEN SAMEN!! CALM DOWN!!

29



POM – Stap 1 Elementen uitlichten

30



POM – Stap 2 Locators zoeken en verwerken in de code

- Natuurlijk eerst een class aanmaken voor deze pagina!
- Er is een handige short cut: **@FindBy** maar dan hebben we wel een constructor nodig!

31



POM – Stap 2 Locators zoeken en verwerken in de code

```
public class ContactUsPage {

    private WebDriver driver;

    @FindBy(css = "select#id_contact")
    private WebElement subjectHeading;

    @FindBy(css = "input#email")
    private WebElement emailTextField;

    @FindBy(css = "input#id_order")
    private WebElement orderIdTextField;

    @FindBy(css = "textarea#message")
    private WebElement messageTextField;

    @FindBy(css = "button#submitMessage")
    private WebElement sendButton;

    public ContactUsPage(WebDriver driver) {
        this.driver = driver;

        // This call sets the WebElement fields.
        PageFactory.initElements(driver, this);
    }
}
```

32



POM – Stap 3 functionaliteit in een methode stoppen

```
public void fillInContactForm(String subject, String email, String orderID, String message){
    // subjectField.sendKeys(subject);
    emailTextField.sendKeys(email);
    orderIdTextField.sendKeys(orderID);
    messageTextField.sendKeys(message);
    sendButton.click();
}
```

33



POM – Aanroepen in een test

```
10  >> public class FillinContactFormTest extends TestShopScenario {
11
12      @Test
13      > public void fillInForm(){
14          // Open the contact page
15          driver.findElement(By.cssSelector("li#header_link_contact > a"))
16              .click();
17
18          ContactUsPage contactUsPage = new ContactUsPage(driver);
19          contactUsPage.fillInContactForm();
20
21      }
22  }
23
```

String subject, String email, String orderID, String message

- Moeten we dan nu het zelfde doen voor de homepage?

34



JAZEKERS!

Testcase 9.01

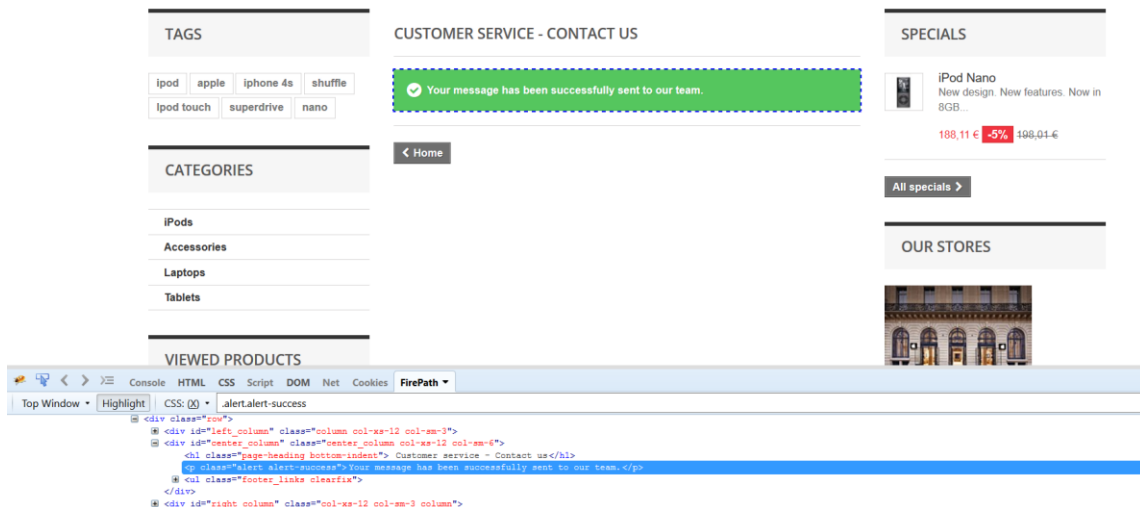
De validatie pagina is open ter interpretatie
Succes!

Testcase 9.01 – Invullen contact formulier

Vragen?

Git add / commit / push

Hoe hebben jullie dit opgelost?



37



Een element benaderen wat er niet meteen is

- WebDriver laten wachten en de DOM te pollen

Implicit Waits

An implicit wait is to tell WebDriver to poll the DOM for a certain amount of time when trying to find an element or elements if they are not immediately available. The default setting is 0. Once set, the implicit wait is set for the life of the WebDriver object instance.

java

```
WebDriver driver = new FirefoxDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
driver.get("http://somedomain/url_that_delays_loading");
WebElement myDynamicElement = driver.findElement(By.id("myDynamicElement"));
```

- WebDriver laten wachten op een specifiek attribuut
 - Explicit wait

http://www.seleniumhq.org/docs/04_webdriver_advanced.jsp#explicit-and-implicit-waits

38



Een element benaderen wat er niet meteen is

- Aangezien het element met de succes message niet meteen verschijnt en..
- We niet de hele test willen laten wachten (implicit) kunnen we een explicit gebruiken voor wanneer het element "er is"

```
public String validationMessage(){
    WebElement alertMessageElement = (new WebDriverWait(driver, 10))
        .until(ExpectedConditions.presenceOfElementLocated(By.cssSelector(".alert.alert-success")));

    return alertMessageElement.getText();
}
```

- Wachten is een kunst!

39



Nu we allemaal het licht van de blokjes hebben gezien

- Voelt dit nog goed?

```
public class TestShopScenario {

    protected WebDriver driver;

    @BeforeMethod
    public void setUp(){
        ChromeDriverManager.getInstance().setup();
        driver = new ChromeDriver();

        // Open the website
        driver.get("https://techblog.polteq.com/testshop/index.php");
    }

    @AfterMethod
    public void tearDown(){
        driver.quit();
    }
}
```

- Wat als we willen switchen naar Firefox?

40



Welkom bij de trend van 2016! (kan ook 2015 zijn)

- Browser Factories!!



41



Browser Factory (Basic)

- Een aparte class die ons de driver terug geeft
- Kunnen kiezen welke driver wij willen (firefox, chrome, ie)
- Maakt gebruik van een switch ipv if/else

To the code! – BrowserFactoryBasic.class

42



Browser Factory (Basic)

```
public class BrowserFactoryBasic {
    public static WebDriver getDriver(String browser){
        WebDriver driver;

        switch(browser.toLowerCase()) {
            case "firefox":
                FirefoxDriverManager.getInstance().setup();
                driver = new FirefoxDriver();
                break;
            case "ie":
                InternetExplorerDriverManager.getInstance().setup();
                driver = new InternetExplorerDriver();
                break;
            case "chrome":
                ChromeDriverManager.getInstance().setup();
                driver = new ChromeDriver();
                break;
            case "edge":
                EdgeDriverManager.getInstance().setup();
                driver = new EdgeDriver();
                break;
            default:
                driver = new ChromeDriver();
        } // end switch
        return driver;
    }
}
```

Internet Explorer doet het niet, zullen we gaan debuggen?



43

Browser Factory (Medior)

- Een aparte class die ons de driver terug geeft
- Kunnen kiezen welke driver wij willen (firefox, chrome, ie)
- Maakt gebruik van een switch ipv if/else
- **Kan de driver extra capabilities meegeven** (aparte methode voor netheid!)

To the code! – BrowserFactoryMedior.class



44

Browser Factory (Medior)

```
public class BrowserFactoryMedior {
    public static WebDriver getDriver(String browser){
        WebDriver driver;

        switch(browser.toLowerCase()) {
            case "firefox":
                return createFireFoxBrowser();
            case "ie":
                return createIEBrowser();
            case "chrome":
                return createChromeBrowser();
            default:
                return createChromeBrowser();
        } // end switch
    }

    private static WebDriver createChromeBrowser() {
        DesiredCapabilities capabilities = DesiredCapabilities.chrome();
        //Chrome options are chrome specific
        ChromeOptions options = new ChromeOptions();
        options.addArguments("--start-maximized");
        options.addArguments("ignore-certificate-errors");
        //Capabilities can used for WebDriver capabilities ie: proxy
        capabilities.setCapability("chrome.switches", "--verbose");
        capabilities.setCapability(ChromeOptions.CAPABILITY, options);
        ChromeDriverManager.getInstance().setup();
        return new ChromeDriver(capabilities);
    }
}
```

Welke regel code is niet meer nodig?

45



Browser Factory (Advanced)

- Een aparte class die ons de driver terug geeft
- Kunnen kiezen welke driver wij willen (firefox, chrome, ie)
- Maakt gebruik van een switch ipv if/else
- Kan de driver extra capabilities meegeven (aparte methode voor netheid!)
- **Browser wordt nu aangemaakt als enum**

46



Enum?

- Enums zijn een lijst van constanten (vandaar all caps)
- Als een lijst met voor gedefinieerden waarden vast staat zijn enums aan te raden
- Echter wel aan te raden met een relatief kleine lijst*
- Gebruik ten opzichte van String (of int): compiler check is beter en je kan geen invaliden constanten meegeven. De waarden zijn pre-defined en zijn al gedocumenteerd (in de code).
- Ideaal dus om Browser errors te voorkomen!

47



Enum? – Samen oefenen

- Oefening met de dagen van de bootcamp
- 5 dagen
- Elke dag krijgt een eigen "actie"
- Tenslotte de class aanroepen en de enum in actie zien!

To the code! – EnumExample.class

48



Opdracht – Browser Factory Advanced maken

1. Check de git branch
2. Maak een nieuwe class genaamd BrowserFactoryAdanced.class
3. Gebruik de medior BrowserFactory als uitgangspunt
4. Maak een enum aan genaamd Browser met de bijbehorende browsers
5. Pas de switch aan zodat de enum wordt gebruikt ipv de String uit BrowserFactoryMedior
6. Check of de nieuwe BrowserFactoryAdvanced werkt met test case 9.01

49



Browser Factory (Advanced)

Uitwerking volgt later want anders kunnen jullie al afkijken 😊

Git add / commit / push

50



The road so far

- Handvaten om ipv linear gemoduleerd te werken
- Java Class gemaakt en aangeroepen
- Java Class met een constructor gemaakt en aangeroepen
- POM opgezet voor meerdere pagina's
- POM pagina's gebruikt om een test te maken
- BrowserFactory gebouwd voor meer gemoduleerd browser control

51



Testcase 9.02 – Inloggen Polteq WebShop

- Opdrachten bijlage -> 9.02
- Correcte Git branch?
- Maak gebruik van het Page Object Model
- Denk aan validatie!
- Kijk goed naar je referentie materiaal

52



Testcase 9.02 – Inloggen Polteq WebShop

Vragen?

Git add / commit / push

53



Testen is niet altijd het groene pad bewandelen, toch?



DEDICATION

No matter how miserable the weather, this guy's gonna make sure
you get where you need to go

Rainy Day test cases!

54



Website heeft ook "error" elementen

CUSTOMER SERVICE - CONTACT US

SEND A MESSAGE

Subject Heading
Customer s...

Message

For any question about a product, an order

Email address
nope ✖

Order reference
jijip

Attach File
No file selected

Send >

CUSTOMER SERVICE - CONTACT US

✖ There is 1 error
1. Invalid email address.

SEND A MESSAGE

Subject Heading
Customer s...

Message

For any question about a product, an order

Email address
nope

Order reference
jijip

Attach File
No file selected

Send >

Hoort dit ook in de POM?

55



UI specific alert kunnen in de POM

- Een UI specific alert maakt onderdeel uit van de DOM
- Kan dus bij andere FindBy's

```
@FindBy(css = "div.alert.alert-danger")
private WebElement errorMessage;
```

- De constructor reserveert geheugen voor het WebElement maar..
- .. Selenium gaat zoeken wanneer het WebElement gebruikt wordt..
- .. We krijgen dus geen error wanneer we een object maken van de Page.

56



Testcase 9.03 – Invullen contact formulier NOK

- Opdrachten bijlage -> 9.03
- Correcte Git branch?
- Maak gebruik van het Page Object Model
- Denk aan validatie!
- Kijk goed naar je referentie materiaal

57



Testcase 9.03 – Invullen contact formulier NOK

Vragen?

Git add / commit / push

58



Testcase 9.04 – Correct email format contact formulier

- Opdrachten bijlage -> 9.04
- Correcte Git branch?
- Maak gebruik van het Page Object Model
- Denk aan validatie!
- Kijk goed naar je referentie materiaal

59



Testcase 9.04 – Correct email format contact formulier

Vragen?

Git add / commit / push

60



"x aantal test cases, allemaal chrome..."



Disclaimer: De Polteq Bootcamp trainers hebben geen aandelen van Google, Chrome of andere Alphabet divisies

61



Semi Hard Coded

- Onze browser keuze is wel ondergebracht in een aparte annotatie..
- ..methode behorende tot de annotatie, roept ook netjes een browser factory aan...
- Maar als ik een enkele test case wil runnen in een andere browser moet ik in de code gaan spitten.

Toch?

62



@Parameter -> Browser makkelijker aanroepen

- Met @Parameter annotatie kunnen wij externe data meegeven aan @Test / @BeforeMethod / AfterMethod enz enz
- De data voor de @Parameter wordt gehaald uit een aparte XML
- Deze XML is te runnen als TestNG class en kan meerdere tests bevatten

63



@Parameter -> Browser makkelijker aanroepen

- Stap1 : voor de duidelijkheid maken we een aparte package aan om daar in te kunnen "experimenteren"
- Stap2 : Nemen een test en testshopsscenario, kopiëren deze in de package en hernoemen ze naar BrowserDrivenTest en TestShopScenarioBrowserDriven
- Stap3 : De test class zelf laten wij met rust want onze driver aanroep zit in de.....
- Stap4 : Voegen de @Parameter annotatie toe met bijbehorende data
- Stap 5 : Bouwen de .xml file

64



@Parameter -> Browser makkelijker aanroepen

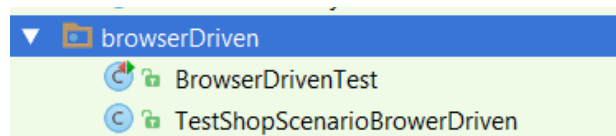
- Stap 6 : Passen de @BeforeMethod methode aan om de parameter te "ontvangen"
- Stap 7 : Testen de .xml file
- Stap 8 : Breiden de .xml file uit met dezelfde test in een andere browser
- Stap 9 : Run de .xml en aanschouw de diversiteit

65



@Parameter -> Browser makkelijker aanroepen

- Stap1 : voor de duidelijkheid maken we een aparte package aan om daar in te kunnen "experimenteren"
- Stap2 : Nemen een test en testshopsscenario, kopiëren deze in de package en hernoemen ze naar BrowserDrivenTest en TestShopScenarioBrowserDriven



66



@Parameter -> Browser makkelijker aanroepen

- Stap3 : De test class zelf laten wij met rust want onze driver aanroep zit in de.....
- Stap4 : Voegen de @Parameter annotatie toe met bijbehorende data

```
@Parameters("browser")
@BeforeMethod
public void setUp() {
    //driver = BrowserFactoryBasic.getDriver("Chrome");
    //driver = BrowserFactoryMedior.getDriver("Chrome");
    driver = BrowserFactoryAdvanced.getDriver(BrowserFactoryAdvanced.Browser.CHROME);

    // Open the website
    driver.get("https://techblog.polteq.com/testshop/index.php");
}
```

67



@Parameter -> Browser makkelijker aanroepen

- Stap 5 : Bouwen de .xml file

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
4
5 <suite name="Suite" parallel="none">
6   <test name="FirefoxTest">
7     <parameter name="browser" value="FIREFOX" />
8     <classes>
9       <class name="browserDriven.BrowserDrivenTest" />
10    </classes>
11  </test>
12 </suite>
```

68



@Parameter -> Browser makkelijker aanroepen

- Stap 6 : Passen de @BeforeMethod methode aan om de parameter te "ontvangen"

```
@Parameters("browser")
@BeforeMethod
public void setUp(BrowserFactoryAdvanced.Browser browser){
    //driver = BrowserFactoryBasic.getDriver("Chrome");
    //driver = BrowserFactoryMedior.getDriver(browser);
    driver = BrowserFactoryAdvanced.getDriver(browser);

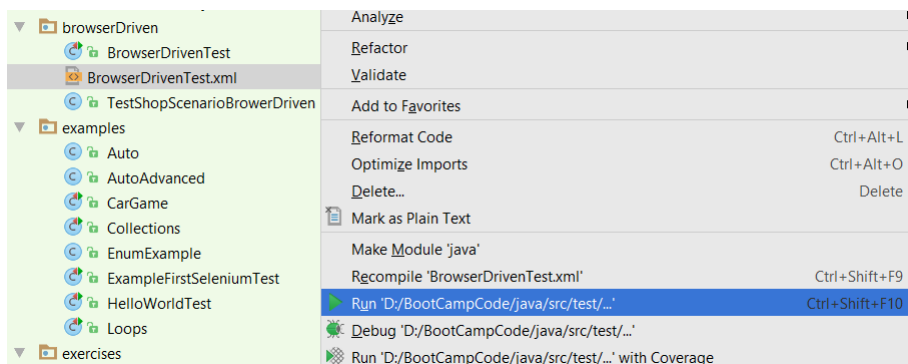
    // Open the website
    driver.get("https://techblog.polteq.com/testshop/index.php");
}
```

69



@Parameter -> Browser makkelijker aanroepen

- Stap 7 : Testen de .xml file

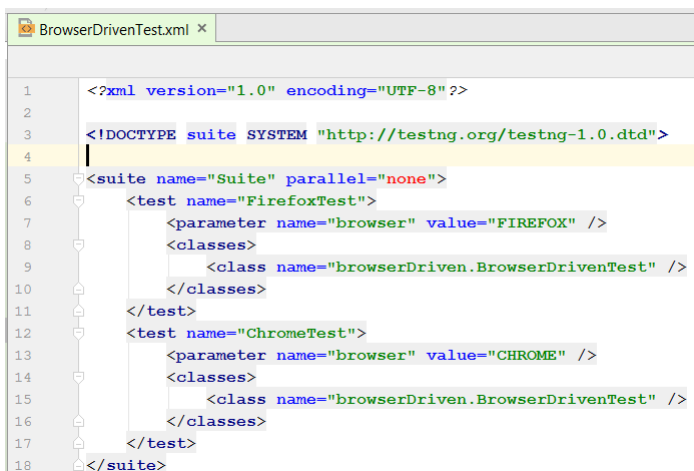


70



@Parameter -> Browser makkelijker aanroepen

- Stap 8 : Breiden de .xml file uit met dezelfde test in een andere browser



```

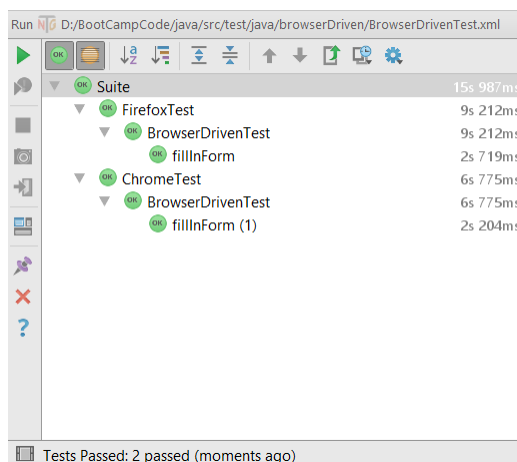
1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
4
5  <suite name="Suite" parallel="none">
6      <test name="FirefoxTest">
7          <parameter name="browser" value="FIREFOX" />
8          <classes>
9              <class name="browserDriven.BrowserDrivenTest" />
10         </classes>
11     </test>
12     <test name="ChromeTest">
13         <parameter name="browser" value="CHROME" />
14         <classes>
15             <class name="browserDriven.BrowserDrivenTest" />
16         </classes>
17     </test>
18 </suite>
  
```

71



@Parameter -> Browser makkelijker aanroepen

- Stap 9 : Run de .xml en aanschouw de diversiteit



Run D:\BootCampCode\java\src\test\java\browserDriven\BrowserDrivenTest.xml

Test Name	Time
Suite	15s 987ms
FirefoxTest	9s 212ms
BrowserDrivenTest	9s 212ms
fillInForm	2s 719ms
ChromeTest	6s 775ms
BrowserDrivenTest	6s 775ms
fillInForm (1)	2s 204ms

Tests Passed: 2 passed (moments ago)

72



Opdracht – Browser Driven Test zelf maken

- Volg de besproken stappen met de TestCase 9.01
- Check de git branch
- Tip: een aparte package voor overzicht? CopyPaste mag altijd

73



Opdracht – Browser Driven Test zelf maken

Vragen?

Git add / commit / push

74



Met parameters kunnen we externe data aanroepen...

Waar zouden wij dit nog meer voor kunnen gebruiken?



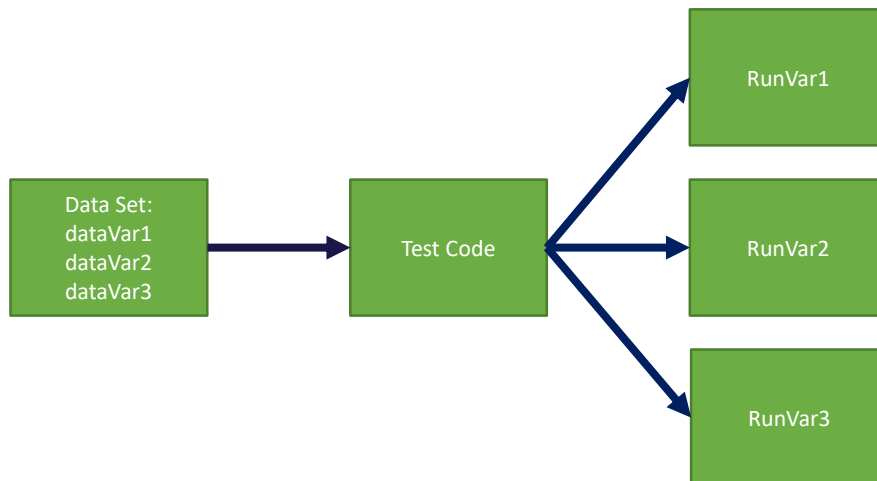
75



Data Driven Testing!

Minimal effort, maximum coverage

Data Driven Testing



77



@Parameter -> Data Driven Testing

- Stap1 : voor de duidelijkheid maken we een aparte package (dataDriven) aan om daar in te kunnen "experimenteren"
- Stap2 : Neem een test, kopieer deze in de package en hernoemen het naar DataDrivenTest
- Stap3 : De TestScenario class zelf laten wij met rust want onze driver aanroep was de vorige oefening
- Stap4 : Voegen de @Parameter annotatie toe met bijbehorende data
- Stap5 : Bouwen de .xml file

78



@Parameter -> Data Driven Testing

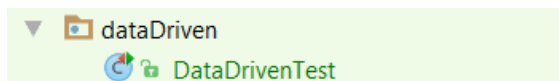
- Stap 6 : Passen de @Test methode aan om de parameter te "ontvangen"
- Stap 7 : Testen de .xml file
- Stap 8 : Breid de .xml file uit met meer data voor dezelfde test
- Stap 9 : Run de .xml en aanschouw de diversiteit

79



@Parameter -> Data Driven Testing

- Stap1 : voor de duidelijkheid maken we een aparte package aan om daar in te kunnen "experimenteren"
- Stap2 : Nemen een test, kopieer deze in de package en hernoemen het naar DataDrivenTest
- Stap3 : De TestScenario class zelf laten wij met rust want onze driver aanroep was de vorige



80



@Parameter -> Data Driven Testing

- Stap4 : Voegen de @Parameter annotatie toe met bijbehorende data

```
@Parameters({"subject", "email", "orderId", "message"})
@Test
public void fillInForm(){
    HomePage homePage = new HomePage(driver);
    homePage.clickContactUS();

    ContactUsPage contactUsPage = new ContactUsPage(driver);
    contactUsPage.fillInContactForm("Customer service", "test@tester.com" , "15", "Duurt lang!");

    Assertions.assertThat(contactUsPage.validationMessage()).as("Validate succesfull contactform submit")
        .isEqualToIgnoringCase("your message has been successfully sent to our team.");
}
```

81



@Parameter -> Data Driven Testing

- Stap5 : Bouwen de .xml file

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Suite" parallel="none">
  <test name="dataVar1Test">
    <parameter name="subject" value="Customer Service" />
    <parameter name="email" value="bootcamper@feelthepain.com" />
    <parameter name="orderId" value="4321234" />
    <parameter name="message" value="Waar zijn m'n eiwitten?!?!?!>
    <classes>
      <class name="dataDriven.DataDrivenTest" />
    </classes>
  </test>
</suite>
```

82



@Parameter -> Data Driven Testing

- Stap 6 : Passen de @Test methode aan om de parameter te "ontvangen"

```
@Parameters({"subject", "email", "orderId", "message"})
@Test
public void fillInForm(String subject, String email, String orderId, String message){
    HomePage homePage = new HomePage(driver);
    homePage.clickContactUS();

    ContactUsPage contactUsPage = new ContactUsPage(driver);
    contactUsPage.fillInContactForm(subject, email, orderId, message);

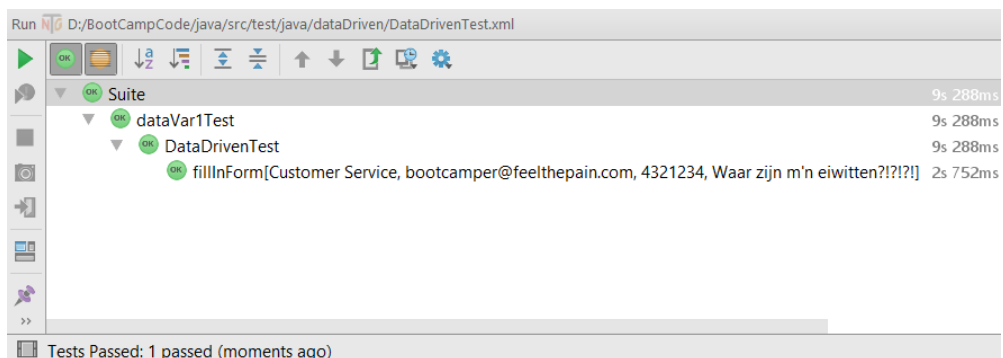
    Assertions.assertThat(contactUsPage.validationMessage()).as("Validate successful contactform submit")
        .isEqualToIgnoringCase("your message has been successfully sent to our team.");
}
```

83



@Parameter -> Data Driven Testing

- Stap 7 : Testen de .xml file



84



@Parameter -> Data Driven Testing

- Stap 8 : Breid de .xml file uit met meer data voor dezelfde test

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
4
5  <suite name="Suite" parallel="none">
6    <test name="dataVar1Test">
7      <parameter name="subject" value="Customer Service" />
8      <parameter name="email" value="bootcamper@feelthepain.com" />
9      <parameter name="orderId" value="4321234" />
10     <parameter name="message" value="Waar zijn m'n eiwitten?!?!?!!" />
11     <classes>
12       <class name="dataDriven.DataDrivenTest" />
13     </classes>
14   </test>
15   <test name="dataVar2Test">
16     <parameter name="subject" value="Customer Service" />
17     <parameter name="email" value="tester@test.com" />
18     <parameter name="orderId" value="09876" />
19     <parameter name="message" value="Mn MacBook is te oud :(" />
20     <classes>
21       <class name="dataDriven.DataDrivenTest" />
22     </classes>
23   </test>
24   <test name="dataVar3Test">
25     <parameter name="subject" value="Customer Service" />
26     <parameter name="email" value="feelfit@nopainnogain.com" />
27     <parameter name="orderId" value="12345" />
28     <parameter name="message" value="Busted my iphone will lifting bro!" />
29     <classes>
30       <class name="dataDriven.DataDrivenTest" />
31     </classes>
32   </test>
33 </suite>

```



85

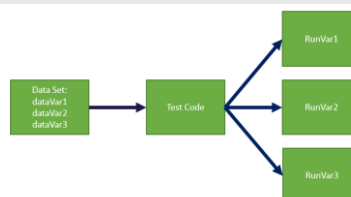
@Parameter -> Data Driven Testing

- Stap 9 : Run de .xml en aanschouw de diversiteit

Run N D D:\BootCampCode\java\src\test\java\dataDriven\DataDrivenTest.xml

Test Case	Duration
Suite	23s 206ms
dataVar1Test	8s 27ms
DataDrivenTest	8s 27ms
fillInForm[Customer Service, bootcamper@feelthepain.com, 4321234, Waar zijn m'n eiwitten?!?!?!]	2s 158ms
dataVar2Test	6s 904ms
DataDrivenTest	6s 904ms
fillInForm[Customer Service, tester@test.com, 09876, Mn MacBook is te oud :()] (1)	2s 201ms
dataVar3Test	8s 275ms
DataDrivenTest	8s 275ms
fillInForm[Customer Service, feelfit@nopainnogain.com, 12345, Busted my iphone will lifting bro!] (2)	2s 80ms

Tests Passed: 3 passed (moments ago)



86

Testcase 9.05 – Vul in een contact formulier (DDT)

- Opdrachten bijlage -> 9.05
- Maak gebruik van het Page Object Model
- Maak gebruik van Data Driven Testing
- Denk aan validatie!
- Kijk goed naar je referentie materiaal

87



Testcase 9.05 – Vul in een contact formulier (DDT)

Vragen?

Git add / commit / push

88



Testcase 9.06 – Log in Polteq WebShop (DDT)

- Opdrachten bijlage -> 9.07
- Maak gebruik van het Page Object Model
- Maak gebruik van Data Driven Testing
- Denk aan validatie!
- Kijk goed naar je referentie materiaal

89



Testcase 9.06 – Log in Polteq WebShop (DDT)

Vragen?

Git add / commit / push

90



Extra Opdracht – DDT met een externe file

- Onze DDT tests werden gedreven door de testNG.xml format
- Het is ook mogelijk om een DDT test op te zetten die gedreven wordt door een externe file
- Zoek uit en bouw een DDT test die wordt gedreven door een .csv of .xls(x)

91



Ons framework gebouwd from scratch

- Pagina's gemoduleerd als classes
- Meerdere TestCases die gebruiken van het bovenstaande POM model
- BrowserFactory gebouwd
- Browser switching made easy
- Framework ondersteund Data Driven Testing

92



Merging time...

- Merge de branch met master (bijv pull request via de site)
- "git checkout master"
- "git pull"



93



Vragen?