

Functional Web Application Testing with Selenium WebDriver

Matteo Biagiola
biagiola@fbk.eu

Courtesy of
Filippo Ricca, DIBRIS,
University of Genova

AGENDA

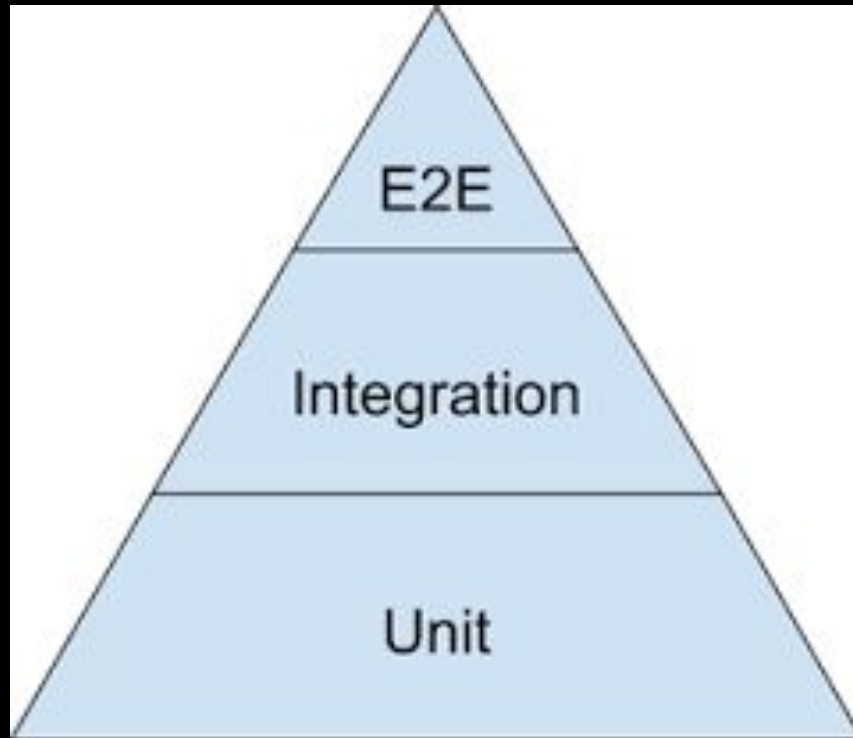
- Basics of functional testing (today)
- Selenium WebDriver (today)
 - Practice
- Page Object Pattern (next lab)



BASICS OF FUNCTIONAL TESTING

Levels of software testing

Test pyramid rule



Quantity

Testing the system
as a whole (GUI)



Individual units are
combined and tested
as a group



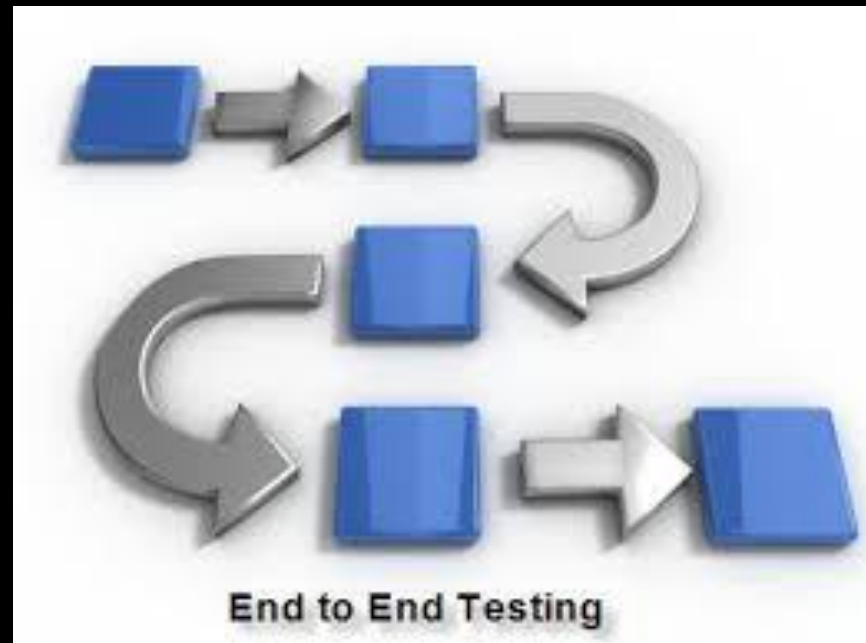
Testing of a single
function/class



End-to-End (E2E) Testing

Testing the system as a whole, all interfaces and backend systems

**Test entire flow
in real conditions**



It is performed from start to finish under **real world scenarios** like communication of the application with hardware, network, database and other applications ...

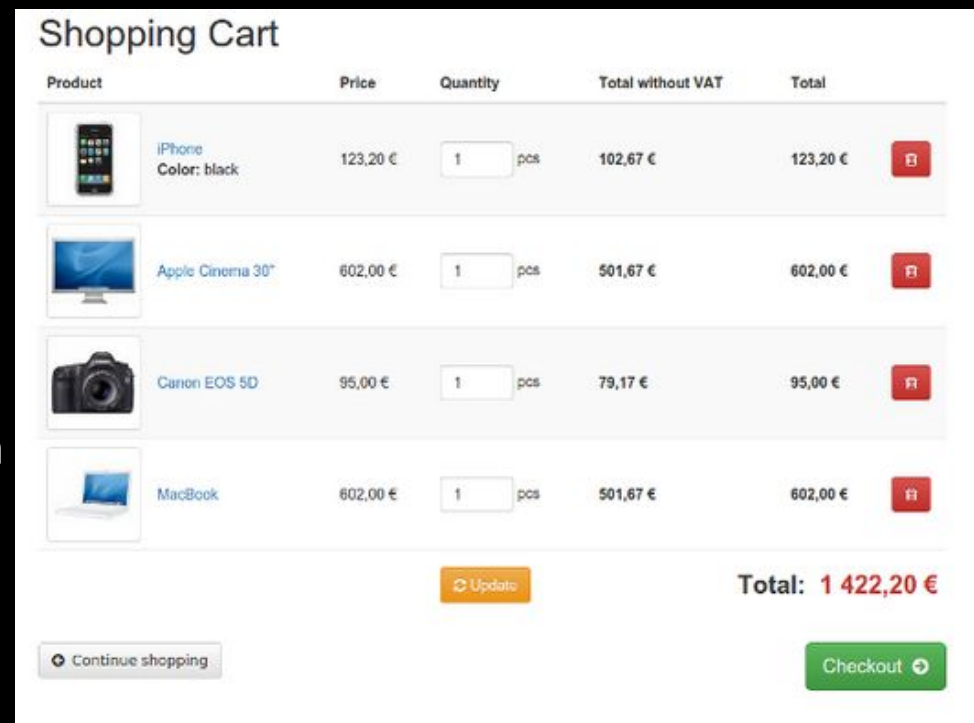
E2E Test Case

- **Triple:**









- Sequence of **user events** from the initial page (state) to the target page (state)
- Sequence of Inputs
 - E.g., to fill in a form
- Expected result (oracle)


- Examples of **user events**:

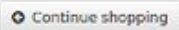

- Click a button
- Select a CheckBox or Radio Button
- Insert text in a text field
 - an input value is necessary



The screenshot shows a 'Shopping Cart' page with a table of items. Each item row includes a product image, name, price, a quantity input field, unit, subtotal, and a total column. At the bottom, there is an 'Update' button, a 'Total' amount, and 'Continue shopping' and 'Checkout' buttons.

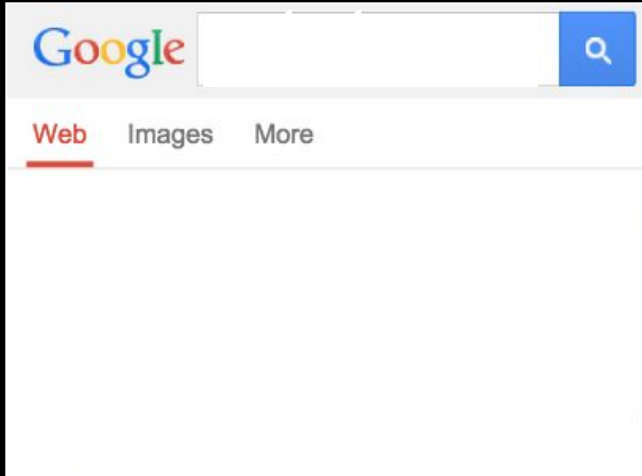
Product	Price	Quantity		Total without VAT	Total	
 iPhone Color: black	123,20 €	<input type="text" value="1"/>	pcs	102,67 €	123,20 €	
 Apple Cinema 30"	602,00 €	<input type="text" value="1"/>	pcs	501,67 €	602,00 €	
 Canon EOS 5D	95,00 €	<input type="text" value="1"/>	pcs	79,17 €	95,00 €	
 MacBook	602,00 €	<input type="text" value="1"/>	pcs	501,67 €	602,00 €	

 **Total: 1 422,20 €**

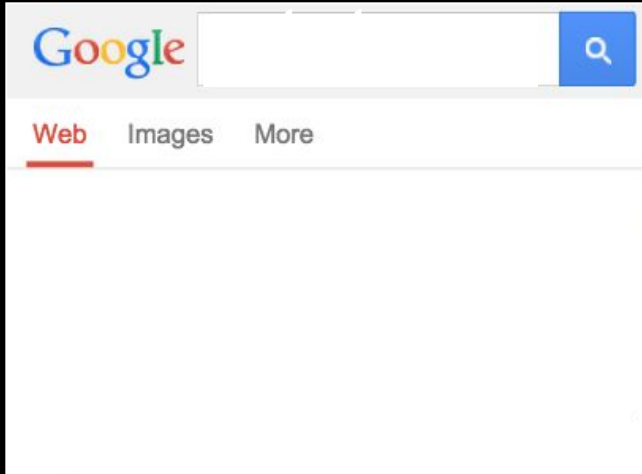
E2E Test case example

Web App under



E2E Test case example

Web App under

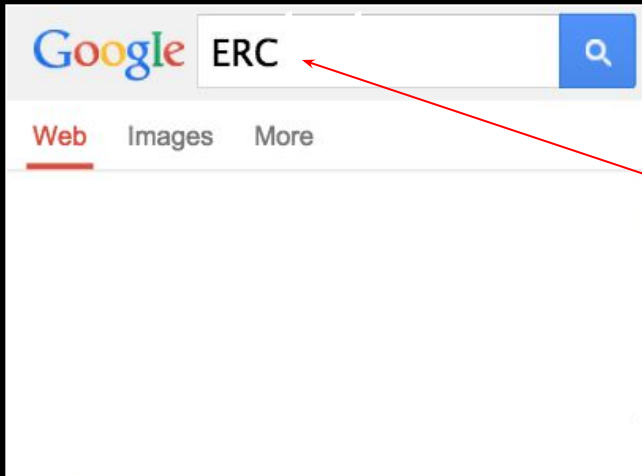


Test case (can be executed

1. Open *www.google.com*
2. Insert "ERC" in input field
3. Click submit button
4. Check first result = "ERC: *European Research*

E2E Test case example

Web App under

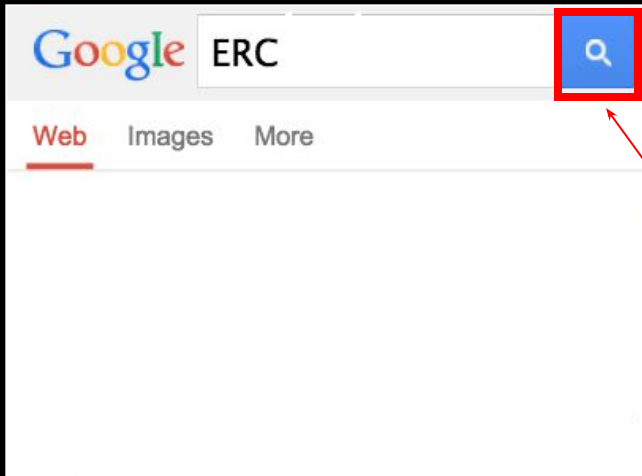


Test case (can be executed manually)

1. Open *www.google.com*
2. Insert "ERC" in input field
3. Click submit button
4. Check first result = "ERC: *European Research*

E2E Test case example

Web App under



Test case (can be executed manually)

1. Open *www.google.com*
2. Insert "ERC" in input field
3. Click submit button
4. Check first result = "ERC: European Research

E2E Test case example

Web App under



Test case (can be executed

manually)

1. Open *www.google.com*
2. Insert "ERC" in input field
3. Click submit button
4. Check first result = "ERC: European Research
"

E2E Test case example

Web App under



Test case (can be executed manually)

1. Open *www.google.com*
2. Insert "ERC" in input field
3. Click submit button
4. Check first result = "ERC: European Research

If so the test case passes!

E2E Login Test Case: Happy case

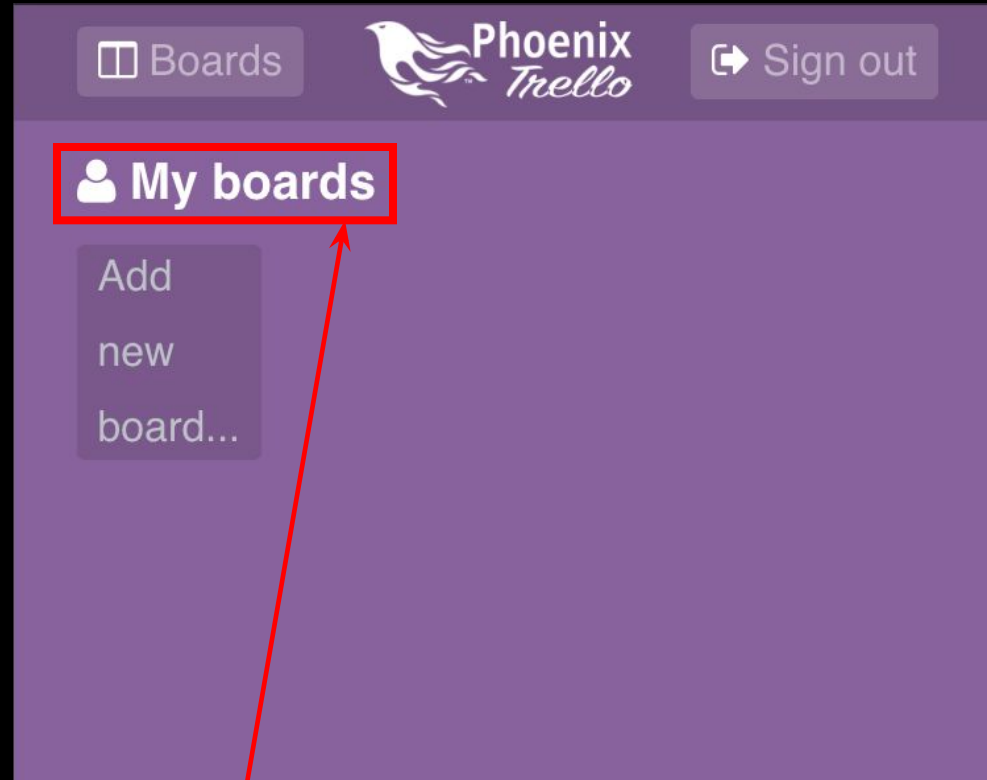


john@phoenix-trello.com

.....


Sign in

Create new account



check result

E2E Login Test Case: Corner case




asd@asd.com

.....

Sign in

Create new account



Invalid email or password

asd@asd.com

.....

Sign in

Create new account

check result

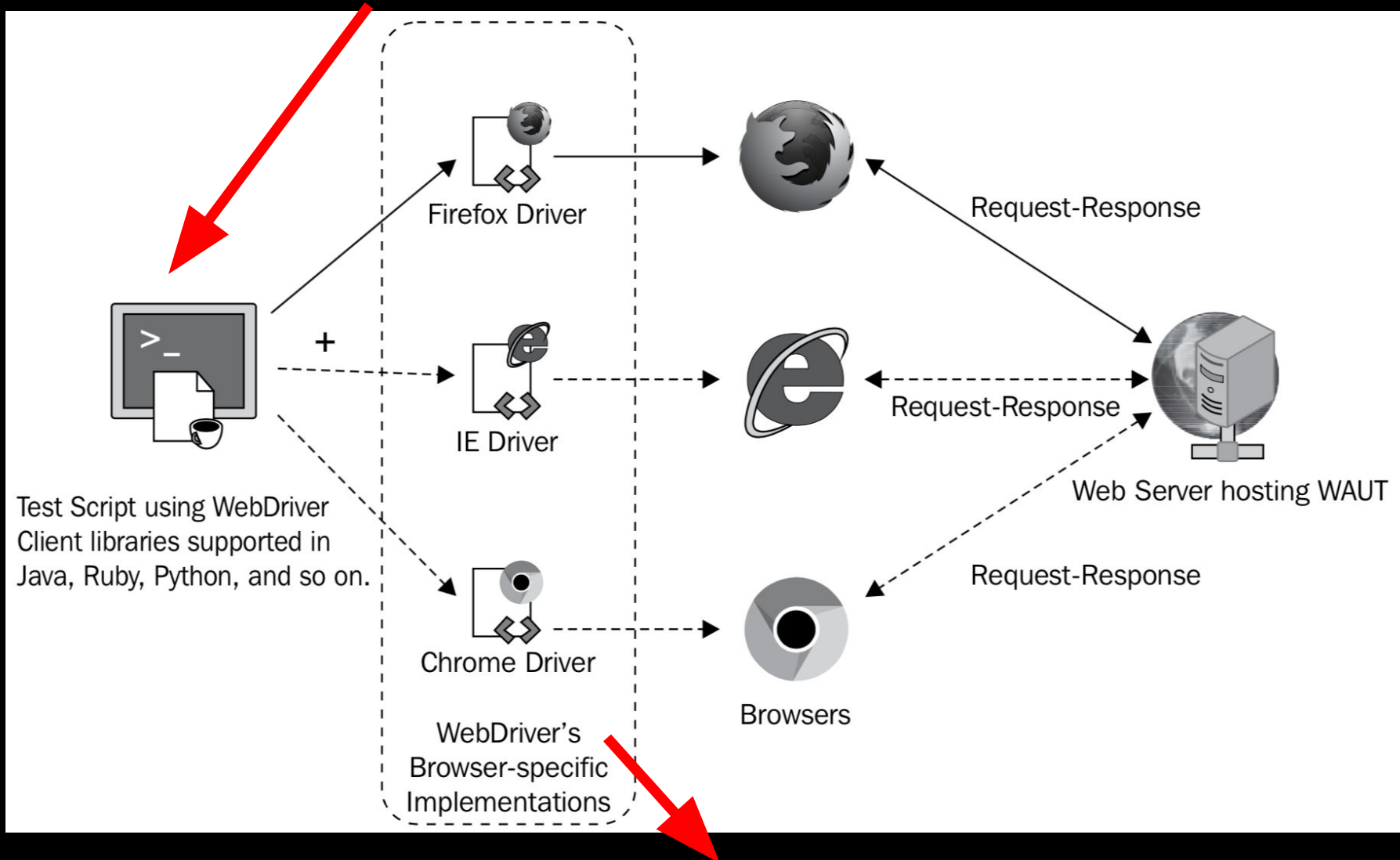
SELENIUM WEBDRIVER

Selenium WebDriver



- Automates the browser
- Software designed to support modern dynamic web pages
- Exploits browser's native support for automation (WebDriver)
- Exposes these features through a uniformed programming interface (API)

Selenium WebDriver



Will work with the browser natively; execute command from outside the browser as the application user would.

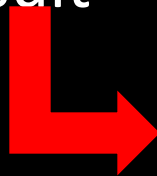
How to use Selenium WebDriver

Write a Test
script that ...

- (1) Goes to a page
- (2) Locates an element
- (3) Does something with that element (e.g.,
click)

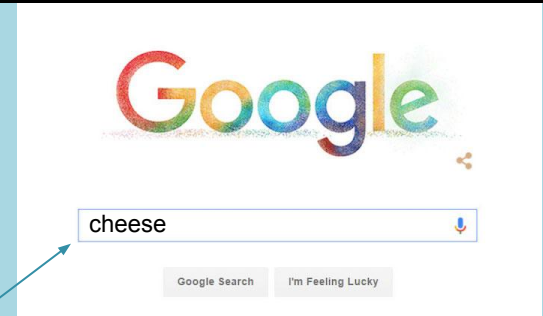
...

- (i) Locates another element
- (i+1) Does something with that element
- (i+2) Asserts the result



A WebDriver Example

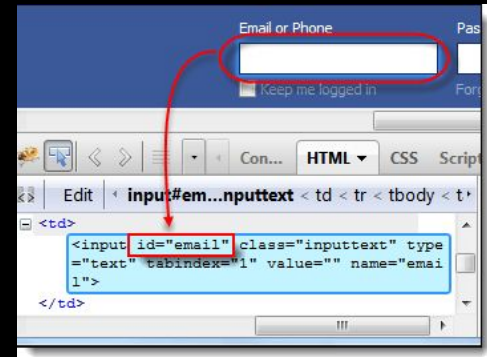
```
...  
libraries  
...  
  
public class GoogleTest {  
  
    public static void main(String[] args) {  
  
        // Specify the path to Firefox driver executable  
        System.setProperty("webdriver.gecko.driver", "/path/to/geckodriver");  
        // Create a new instance of the Firefox driver  
        WebDriver driver = new FirefoxDriver();  
  
        // And now use this to visit Google  
        driver.get("http://www.google.it");  
  
        // Find the text input element by its ID  
        WebElement element = driver.findElement(By.id("gs_hfif0"));  
  
        // Enter something to search for  
        element.sendKeys("cheese");  
  
        // Now submit the form. WebDriver will find the form for us from the element  
        element.submit();  
  
        // Check the title of the page  
        assertEquals("cheese - Search with Google", driver.getTitle());  
  
        //Close the browser  
        driver.quit();  
  
    }  
}
```



How to locate an element

- **By id**

- HTML: `<input id="email" ... />`
- WebDriver: `driver.findElement(By.id("email"));`



- **By name**

- HTML: `<input name="cheese" type="text" />`
- WebDriver: `driver.findElement(By.name("cheese"));`

- **By Xpath**



- HTML

```
<html>
  <input type="text" name="example" />
  <input type="text" name="other" />
</html>
```

- WebDriver: `driver.findElement(By.xpath("/html/input[1]"));`

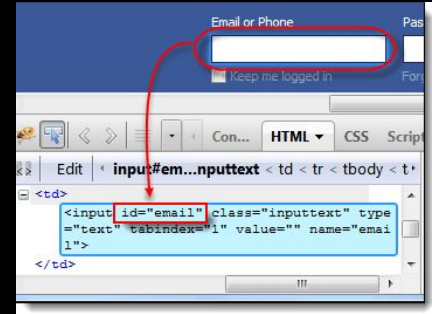
- ...

Selenium Supported Locators

XPath	Name
CSSSelector	LinkText
ClassName	PartialLinkText
ID	TagName

Different types of locators!

XPath locators: an alternative



- **Ids are the best choice**, however...
 - Ids don't always exist
 - adding Ids everywhere is impractical or not viable
 - Their uniqueness is not enforced
 - In some cases, they are 'auto-generated' and so unreliable

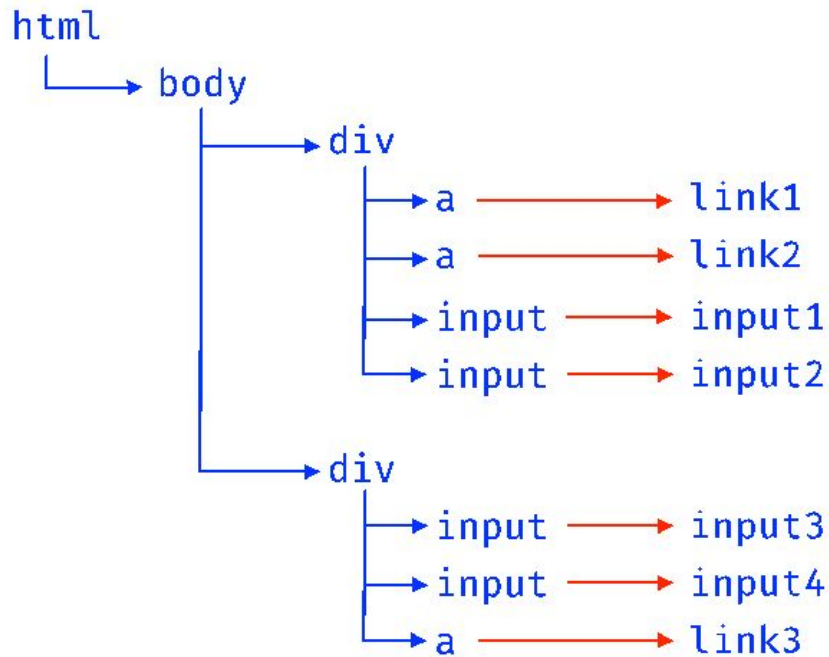
```
driver.findElement(By.id("id_fZI_1")).click();
```

- So **often it is not possible using IDs locators**
 - An alternative is using XPath expressions

Absolute vs. relative XPath

XPath (XML Path Language) is a query language XPath (XML Path Language) is a query language for selecting nodes XPath (XML Path Language) is a query language for selecting nodes from an XML document

DOM tree representation



Absolute XPath. It begins with single slash "/" which means start the search from the **root** node

/html/body/div/input => input{1,2,3,4}

/html/body/div[1]/input[2] => input2

Relative XPath. It begins with double slash "/" which represents **search in the entire web page**

//input => input{1,2,3,4}

//div/input[1] => input{1,3}

//div[2]/input[2] => input4

Still relative: /body//a => link{1,2,3}

Search in the entire subtree under the body node

How to compute XPath



Learn Selenium-WebDriver with Core Java.

A place to learn Selenium-WebDriver and Core Java with real time scenarios.

About Me



Sanjay Kumar

ChroPath Creator.
Product Evangelist.
Living in Bangalore. Pls
follow me on twitter,
because I am more active
there- <https://twitter.com/SanjayKumaarr>

[View my complete profile](#)

Saturday, November 17, 2018

ChroPath Community Design Team

Hi All,
want to contribute to ChroPath, here is the great opportunity to help our QA/Dev community.
Please join the ChroPath Community Design Team by filling below form. Last date to join is 30th Nov. Thank you.

This team will participate in

- 1) Defining new releases,
- 2) Validating beta release and
- 3) Guiding ChroPath to a better product so community benefits.

[Link to join team](#)

Posted by [Sanjay Kumar](#)

Reactions: ☐ like (0) ☐ useful (0) ☐ good (0) ☐ no reaction (0)

No comments: [Links to this post](#)

Total Pageviews

728819

Labels

[.bat command to execute testng.xml thru cmd](#) (1)
['invocationCount](#) (1)
[@DataProvider](#) (1)
[@Parameters](#) (1)
[Actions Class](#) (1)
[Advanced level FAQ on Selenium-WebDriver](#) (10)
[advanced level real time scenario](#) (19)
[Agile Methodolgy](#) (1)
[alert pop-up](#) (1)
[ANT](#) (2)
[Architecture of Selenium](#) (1)
[AutoIt](#) (1)

Wednesday, November 14, 2018

ChroPath 4.0

Inspector Console Debugger {} Style Editor @ Performance Memory Network Storage Accessibility ChroPath

selectors type selector and press enter

inspect 1 matching node found. Find the matching node below :

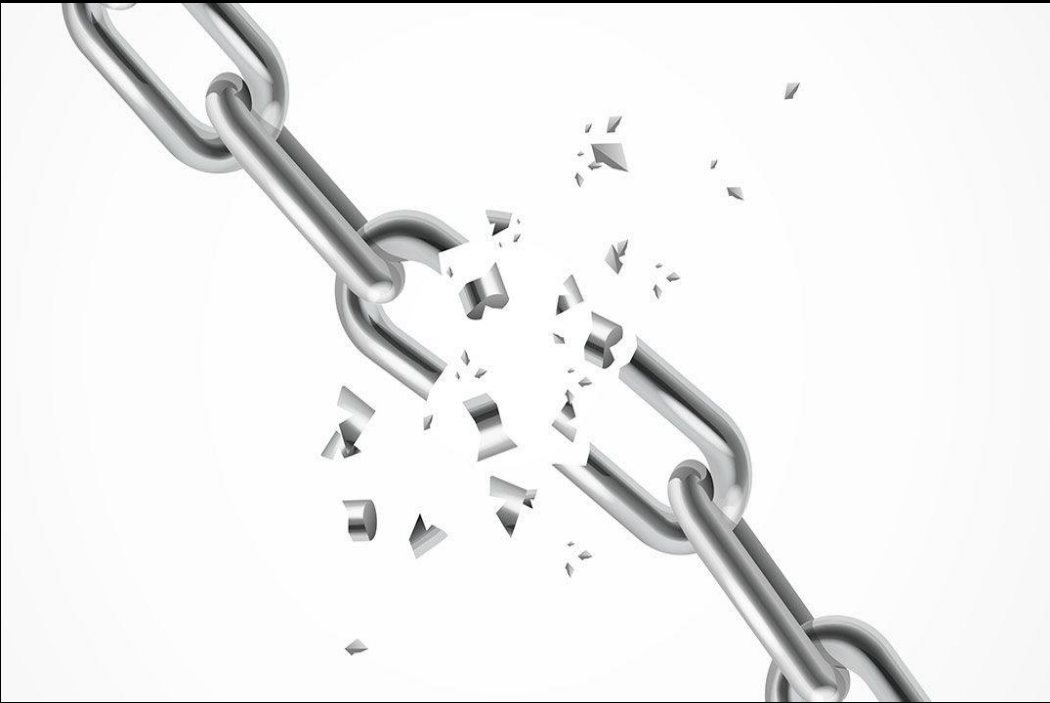
rel XPath: //h1[@class='title']

abs XPath: /html[1]/body[1]/div[4]/div[2]/div[2]/header[1]/div[1]/div[2]/div[1]/div[1]/div[1]/h1[1]

CSS: body.variant-light:nth-child(2) div.content:nth-child(4) div.content-outer div.fauxborder-left.content-fauxborder-left div.content-inner div.header-outer div.fauxborder-left.header-fauxborder-left div.region-inner.header-inner div.header.section div.widget.Header div:nth-child(1) div.titlewrapp...

> <h1 class="title" xpath="1"></h1>

How to cope with dynamism



- One of top reason tests fail
- Due to Javascript DOM manipulation at runtime
- Workaround:
 - Static delays ←
 - Dynamic delays

Questions?



AGENDA

- Basics of functional testing (last time)
- Selenium WebDriver (last time)
- Page Object Pattern (today)



PAGE OBJECT DESIGN PATTERN

Really Simple Web App



<http://www.kimschiller.com/page-object-pattern-tutorial/>

Test Script for Simple Web App

```
public class SimpleTestClass extends BaseTest {

    private static WebDriver driver;

    @Test
    public void testMethod(){
        // Find the Web elements composing the form
        WebElement firstname = driver.findElement(By.id("//input[@id='firstname']"));
        WebElement lastname = driver.findElement(By.id("//input[@id='lastname']"));
        WebElement address = driver.findElement(By.id("//input[@id='address']"));
        WebElement zipcode = driver.findElement(By.id("//input[@id='zipcode']"));
        // Enter firstname and lastname
        firstname.sendKeys("foo");
        lastname.sendKeys("bar");
        // Enter address and zipcode
        address.sendKeys("street foobar, 10");
        zipcode.sendKeys("18017");
        // Now submit the form
        WebElement submitButton = driver.findElement(By.xpath("//input[@id='signup']"));
        submitButton.click();
        // Check the confirmation Header (novel page)
        assertEquals("Thank you!", driver.findElement(By.xpath("html[1]/body[1]/h1[1]")).getText());
    }
}
```

Test Script quality



- Test scripts are difficult to read
 - A lot of implementation details
- Often changes in the Web app breaks multiple tests
 - Fragile test scripts
- Duplication of locators and code across test scripts
 - **no reuse !**



IDEA

Adopting the Page Object Pattern!

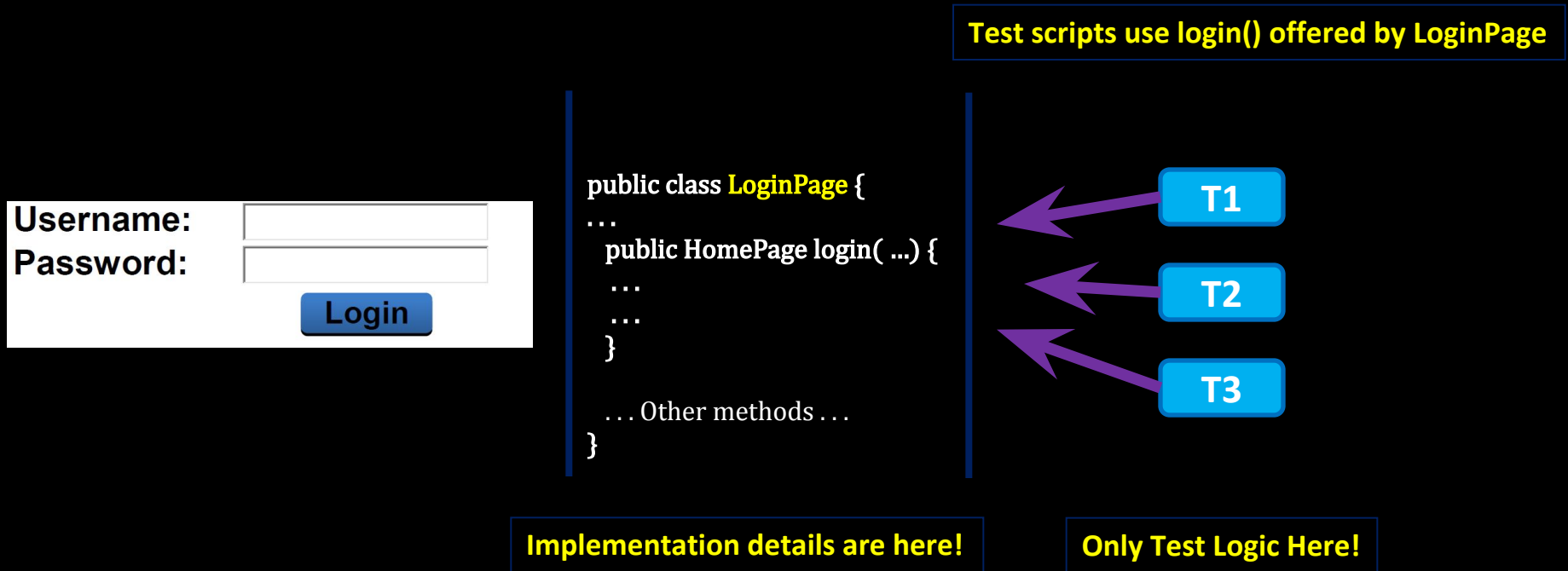
Page Object Pattern

- A level of abstraction between the test scripts and the web pages
 - with the aim of reducing the coupling among them



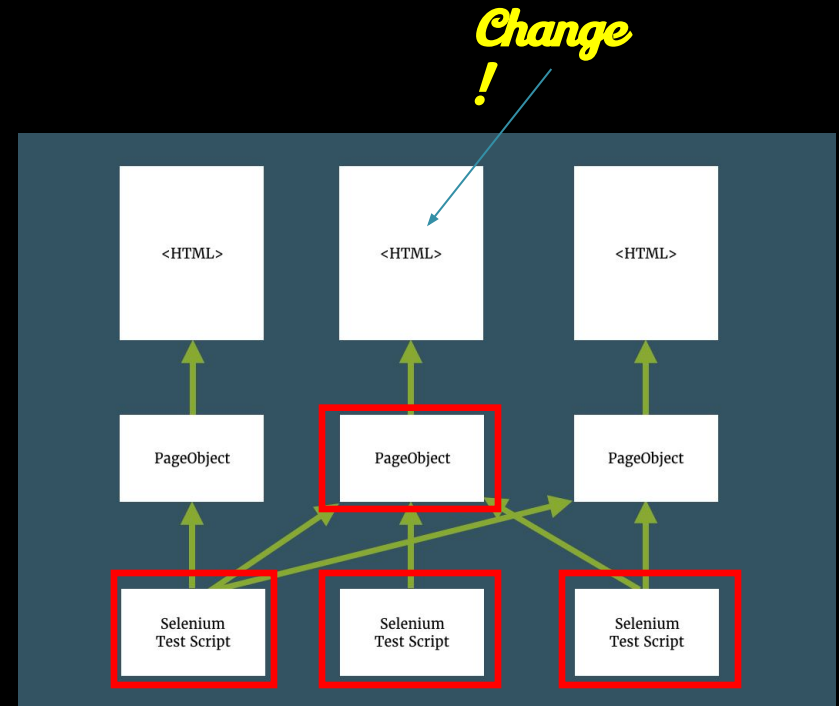
- **Idea:** creating a page class for each web page
- Each method encapsulates a page's functionality
 - e.g., Login

Page Object Pattern: the idea ...



Advantages

- **Test scripts are simpler**
 - Implementation details are in the POs
 - Easier to read (app specific API)
- **Reuse**
 - The same method is called by several Test scripts
 - e.g., login()
- **Maintenance effort reduction**
 - A change in a Web page can affect only one PO
 - not a bunch of Test scripts!



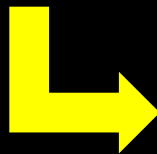
Page Object Creation: SignUpPage

```
public class SignUpPage extends PageObject {  
  
    @FindBy(xpath="//input[@id='firstname']")  
    private WebElement firstName;  
  
    @FindBy(xpath="//input[@id='lastname']")  
    private WebElement lastName;  
  
    @FindBy(xpath="//input[@id='address']")  
    private WebElement address;  
  
    @FindBy(xpath="//input[@id='zipcode']")  
    private WebElement zipCode;  
  
    @FindBy(xpath="//input[@id='signup']")  
    private WebElement submitButton;  
  
    public SignUpPage(WebDriver driver) {  
        super(driver);  
    }  
}
```

@FindBy annotation!

PO methods

```
public void enterName(String firstName, String lastName){  
    this.firstName.clear();  
    this.firstName.sendKeys(firstName);  
    this.lastName.clear();  
    this.lastName.sendKeys(lastName);  
}  
  
public void enterAddress(String address, String zipCode){  
    this.address.clear();  
    this.address.sendKeys(address);  
    this.zipCode.clear();  
    this.zipCode.sendKeys(zipCode);  
}  
  
public ReceiptPage submit(){  
    this.submitButton.click();  
    return new ReceiptPage(driver);  
}  
}
```



Page Object Creation: ReceiptPage

```
public class ReceiptPage extends PageObject {  
  
    @FindBy(xpath="/html[1]/body[1]/h1[1]")  
    private WebElement header;  
  
    public ReceiptPage(WebDriver driver) {  
        super(driver);  
    }  
  
    public String confirmationHeader(){  
        return header.getText();  
    }  
}
```

Getter method

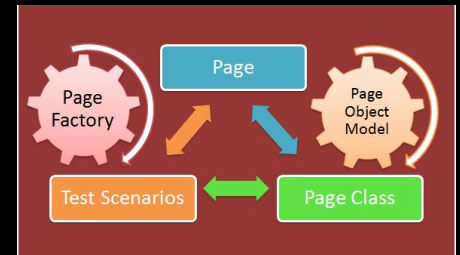
**Assertions are inserted
in the Test Script**



Page Object Creation: PageObject

- Now, this will actually not work, until we **initialize the Web Elements** that we have annotated

```
public class PageObject {  
    protected WebDriver driver;  
  
    public PageObject(WebDriver driver){  
        this.driver = driver;  
        PageFactory.initElements(driver, this);  
    }  
}
```



The PageFactory (WebDriver framework) locates the element on the page using the annotated selector and instantiates the Web elements

(Refactored) Test script

Setup is included

PO methods are called

```
public class SignUpFormTest extends BaseTest {  
  
    @Test  
    public void signUp() {  
        SignUpPage signUpPage = new SignUpPage(driver);  
  
        signUpPage.enterName("foo", "bar");  
        signUpPage.enterAddress("street foobar, 10", "18017");  
        ReceiptPage receiptPage = signUpPage.submit();  
  
        assertEquals("Thank you!", receiptPage.confirmationHeader());  
    }  
}
```

Getter method is used for creating the assertion

Page Objects Best practices



1. A page object should not have any assertion
but only **getter methods**
2. A page object should represent meaningful
elements of a page (components) and not
necessarily a complete page
3. When you navigate you should return a page object
for the next page
4. Factorize common components of a page
Ex. menu present in several pages = PO component
the pages containing the Menu = PO containers

PO Containers and Components (1)

Software Engineering & Programming Languages Research Group

Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS) - University of Genova

[Home](#)[People](#)[Publications](#)[Projects](#)[Research](#)[Meetings](#)[Theses & Stages](#)[Contacts](#)

Links



Dipartimento di Informatica,
Bioingegneria, Robotica
e Ingegneria dei Sistemi



Università degli Studi di Genova

News

Best Student Paper Award (ICWE
2016)

ACM TOSEM paper accepted!

mini Italian Agile Day Genova2014

Welcome to the Software Engineering & Programming Languages Research Group web site

Our overall aim is to develop a number of coherent and integrated innovative techniques for the development of reliable software systems. Modern software systems are complex, distributed, heterogeneous, dynamically evolving, often operate in contexts with unpredictable behaviour, and should be able to adapt to changes in the environment or the requirements. Typically, they are not developed from scratch, but, rather, by composing and modifying existing systems. Hence, we focus on providing maximal flexibility and adaptability, still guaranteeing reliability.

Notably, we investigate new development methodologies (including analysis, design and testing techniques), flexible linguistic constructs for code evolution and reuse (including integration with code mobility and the agent paradigm), and extensions of traditional static typechecking by dynamic/incremental checks and more sophisticated types which express semantic properties.

Our current research interests include:

- Design, foundations and implementation of programming languages
- Object-oriented paradigm
- Advanced modular programming
- Type systems
- Service Oriented Architecture
- Visual notations
- Model Driven Engineering
- Empirical Software Engineering
- Software and Web Testing
- Agent Oriented Software Engineering

**Tirocini
Prove Finali
Tesi**

Leggi QUI!



PO Containers and Components (2)


Home Page

Software Engineering & Programming Languages Research Group


Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS) - University of Genova

- Home
- People
- Publications
- Projects
- Research
- Meetings
- Theses & Stages
- Contacts

Links



Dipartimento di Informatica,
Bioingegneria, Robotica
e Ingegneria dei Sistemi



Università degli Studi di Genova

News

Best Student Paper Award (ICWE 2016)
ACM TOSEM paper accepted!
mini Italian Agile Day Genova2014

Welcome to the Software Engineering & Programming Languages Research Group web site

Our overall aim is to develop a number of coherent and integrated innovative techniques for the development of reliable software systems. Modern software systems are complex, distributed, heterogeneous, dynamically evolving, often operate in contexts with unpredictable behaviour, and should be able to adapt to changes in the environment or the requirements. Typically, they are not developed from scratch, but, rather, by composing and modifying existing systems. Hence, we focus on providing maximal flexibility and adaptability, still guaranteeing reliability.

Notably, we investigate new development methodologies (including analysis, design and testing techniques), flexible linguistic constructs for code evolution and reuse (including integration with code mobility and the agent paradigm), and extensions of traditional static typechecking by dynamic/incremental checks and more sophisticated types which express semantic properties.

Our current research interests include:

- Design, foundations and implementation of programming languages
- Object-oriented paradigm
- Advanced modular programming
- Type systems
- Service Oriented Architecture
- Visual notations
- Model Driven Engineering
- Empirical Software Engineering
- Software and Web Testing
- Agent Oriented Software Engineering



Tirocini
Prove Finali
Tesi

Leggi QUI!

```
public class HomePO extends PageObject {
    @FindBy(how=How.XPATH, xpath="...")
    private WebElement home;
    @FindBy(how=How.XPATH, xpath = "...")
    private WebElement people;

    public HomePO goToHome() {
        home.click();
        return new HomePO(driver);
    }

    public PeopleContainerPO goToPeople() {
        people.click();
        return new PeoplePO(driver);
    }
}
```


PO Containers and Components (3)

People Page

Software Engineering & Programming Languages Research Group
Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS) - University of Genova

Home **People** Publications Projects Research Meetings Theses & Stages Contacts

Links

People

dibris
Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi

unige
UNIVERSITÀ DEGLI STUDI DI GENOVA
Università degli Studi di Genova

News
Best Student Paper Award (ICWE 2016)
ACM TOSEM paper accepted!

	Egidio Astesiano Full Professor Personal Web Page E-mail: astes@disi.unige.it		Gianna Reggio Associate Professor Personal Web Page E-mail: gianna.reggio@unige.it
	Maura Cerioli Associate Professor Personal Web Page E-mail: maura.cerioli@unige.it		Elena Zucca Associate Professor Personal Web Page E-mail: elena.zucca@unige.it
	Filippo Ricca Associate Professor Personal Web Page E-mail: filippo.ricca@unige.it		Davide Ancona Associate Professor Personal Web Page E-mail: davide.ancona@unige.it

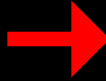
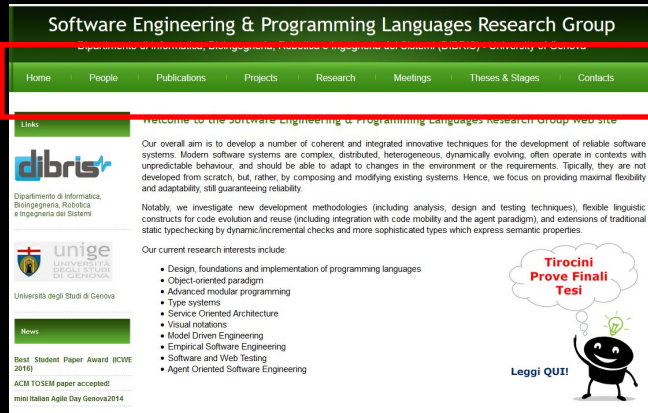
```
public class PeoplePO extends PageObject
{
    @FindBy(how=How.XPATH, xpath="...")
    private WebElement home;
    @FindBy(how=How.XPATH, xpath = "...")
    private WebElement people;

    public HomePO goToHome() {
        home.click();
        return new HomePO(driver);
    }

    public PeopleContainerPO goToPeople() {
        people.click();
        return new PeoplePO(driver);
    }
}
```

Bad choice!! A lot of duplicated code ...

PO Containers and Components (4)



```
public class MenuComponentPO extends PageObject {
    @FindBy(how=How.XPATH, xpath = "... ")
    private WebElement home;
    @FindBy(how=How.XPATH, xpath = "... ")
    private WebElement people;

    public HomeContainerPO goToHome() {
        home.click();
        return new HomeContainerPO(driver);
    }

    public PeopleContainerPO goToPeople() {
        people.click();
        return new PeopleContainerPO(driver);
    }

    ...
}
```

MenuComponentPO

HomeContainerPO

PeopleContainerPO

PubContainerPO

...

Questions?



Useful References

- **Selenium WebDriver Documentation**

- http://www.seleniumhq.org/docs/03_webdriver.jsp
- <https://selenium.dev/selenium/docs/api/java/org/openqa/selenium/support/FindBy.html>

- **Selenium WebDriver Tutorial**

- <http://toolsqa.com/selenium-tutorial/>
- <http://toolsqa.com/selenium-webdriver/configure-eclipse-with-selenium-webdriver/>
- <http://elementalselenium.com/tips/47-waiting>

- **ChromeDriver**

- <https://sites.google.com/a/chromium.org/chromedriver/getting-started>

- **XPath in Selenium: Complete Guide**

- <http://www.guru99.com/xpath-selenium.html>

- **Getting started with Page Object Pattern**

- <https://www.pluralsight.com/guides/software-engineering-best-practices/getting-started-with-page-object-pattern-for-your-selenium-tests>

- **Page Factory**

- <https://github.com/SeleniumHQ/selenium/wiki/PageFactory>