FUNCTIONAL AND SECURITY TESTING TECHNIQUES (FSTT 2024)
ASSIGNEMENT

COMPARING DIFFERENT METHODS FOR DEVELOPING PO-BASED SELENIUM WEBDRIVER E2E TEST SUITES

Maurizio Leotta Filippo Ricca

Dibris

Testing of Web Apps

Assure the functional correctness of Web apps is a must!

WHY?

- **Web apps** are **key assets** of our society
  - Business, health care, public administration, …
  - Billions Internet users worldwide
- **Correctness** is **crucial**
  - One bug, 440 millions of dollars lost in less than one hour [Knight Capital Group 2012]

HOW?

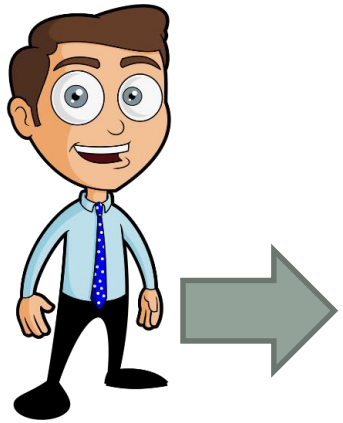
- **End to End Automated testing**
  - **Low level testing** is **too complex** and **not enough** => **E2E testing**!
  - **Manual testing** is **expensive** and **not effective** => **automation required!**

How automated functional E2E Web Tests can be developed?

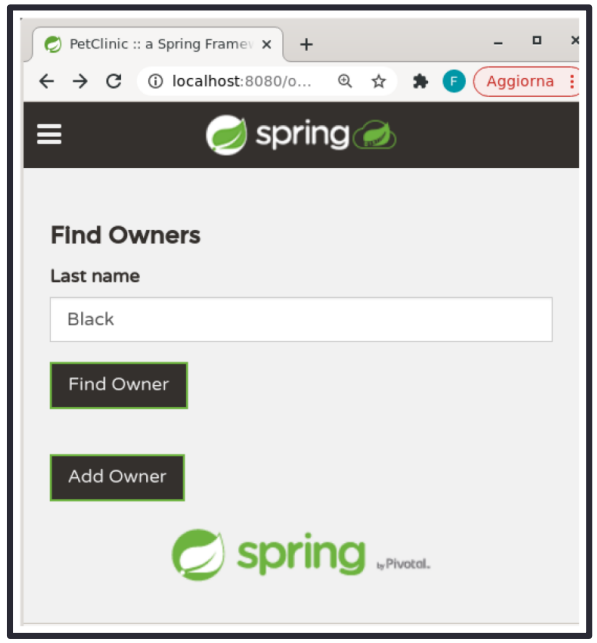
different approaches….

Capture (Record) & Replay web testing

1. **Record** the **actions** performed by the tester on the web application
   - using a specific tool (e.g., Selenium IDE)
2. **Add** one or more **assertions** to the recorded script
3. **Re-execute** them automatically



Tester interacts with the web app



Test sequence automatically recorded

How automated functional E2E Web Tests can be developed?

Capture (Record) & Replay web testing

1. **Record** the **actions** performed by the tester or
   - using a specific tool (e.g., Selenium IDE)
2. **Add** one or more **assertions** to the recorded s
3. **Re-execute** them automatically

Script-based web testing

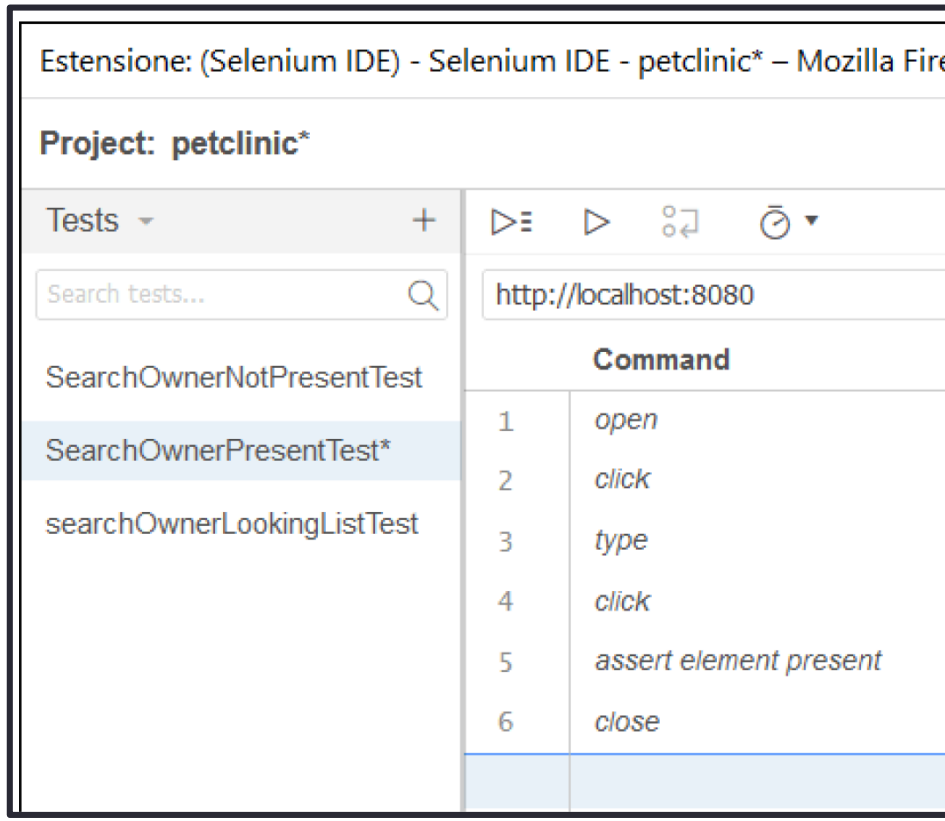
- Test cases are software artefacts created:
  - Using standard programming languages (e.g., Java, C#) and IDE (Eclipse)
  - Resorting to specific testing frameworks (e.g., Selenium WebDriver)

```
public class HomePO {
 WebDriver driver;
 public HomePO(WebDriver driver) {
    this.driver = driver;
 }
 public void gotoFind() {
    driver.findElement(By.css("li:nth-child(2) > a")).click();
 }
}


public class FindPO {
 WebDriver driver;
 public FindPO(WebDriver driver) {
  this.driver = driver;
 }
 public void findOwner(String s) {
  driver.findElement(By.id("lastName")).sendKeys(s);
  driver.findElement(By.css(".btn:nth-child(1)")).click();
 }
 public String returnErrorMsg() {
  return driver.findElement(By.cssSelector("b")).getText();
 }
}



public class OwnerPO {
 WebDriver driver;
 public OwnerPO(WebDriver driver) {
  this.driver = driver;
 }
 public String returnOwnerName() {
  return driver.findElement(By.cssSelector("b")).getText();
 }
}
```

Which E2E approach choose?

Both of them have **strengths** and **weaknesses**

- **Capture Replay web testing**
  - building test cases is a quite simple task
    - even for persons without strong programming experience
  - test cases are developed quickly
  - test cases contain hard-coded values (the inputs)
  - test cases are strongly coupled with web pages
  - each test suite contains a lot of duplicated code
    - common steps among different test cases



Which E2E approach choose?

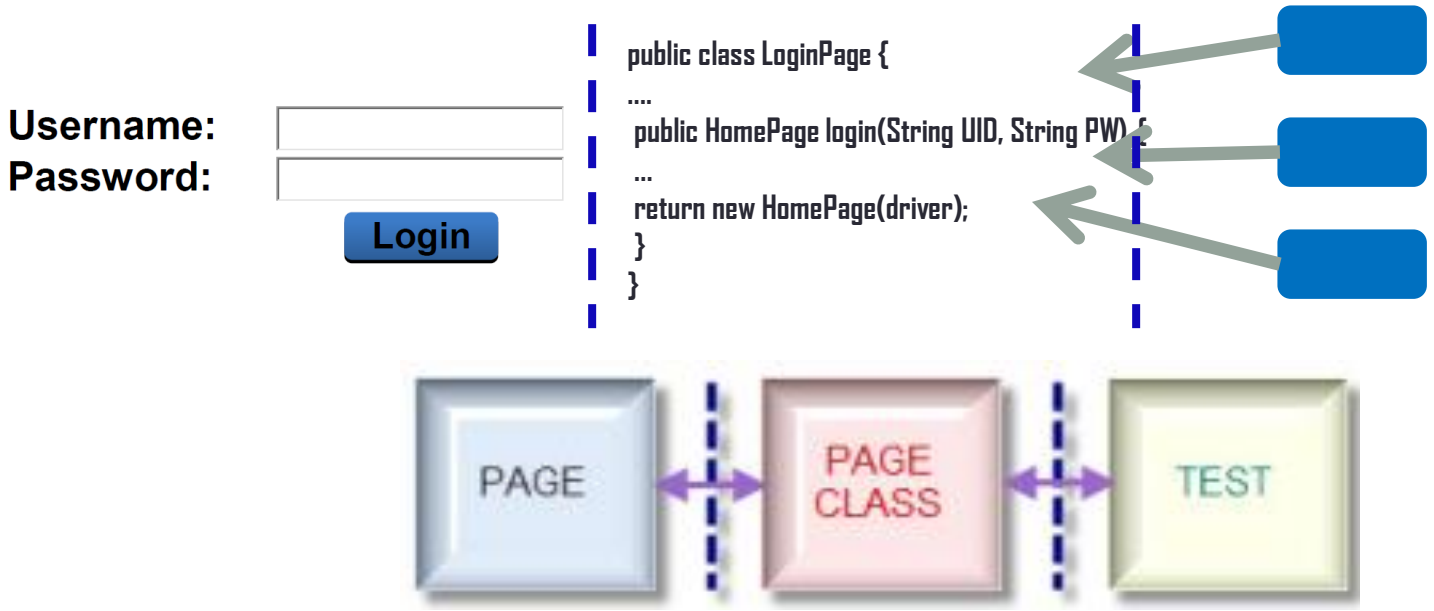
Both of them have **strengths** and **weaknesses**

- **Script-based web testing**
  - test cases are more flexible
    - conditional statements, loops, logging, and exceptions
  - data-driven test cases
    - executed multiple times passing them different arguments
  - test cases seem to be, generally, more maintenable
    - when adopting specific pattern such as the ***PO Pattern***
  - technical skills are required
  - test case development require more time

```
public class HomePO {
 WebDriver driver;
 public HomePO(WebDriver driver) {
    this.driver = driver;
 }
 public void gotoFind() {
    driver.findElement(By.css("li:nth-child(2) > a")).click();
 }
}


public class FindPO {
 WebDriver driver;
 public FindPO(WebDriver driver) {
  this.driver = driver;
 }
 public void findOwner(String s) {
  driver.findElement(By.id("lastName")).sendKeys(s);
  driver.findElement(By.css(".btn:nth-child(1)")).click();
 }
 public String returnErrorMsg() {
  return driver.findElement(By.cssSelector("b")).getText();
 }
}



public class OwnerPO {
 WebDriver driver;
 public OwnerPO(WebDriver driver) {
  this.driver = driver;
 }
 public String returnOwnerName() {
  return driver.findElement(By.cssSelector("b")).getText();
 }
}
```

Page Object Pattern

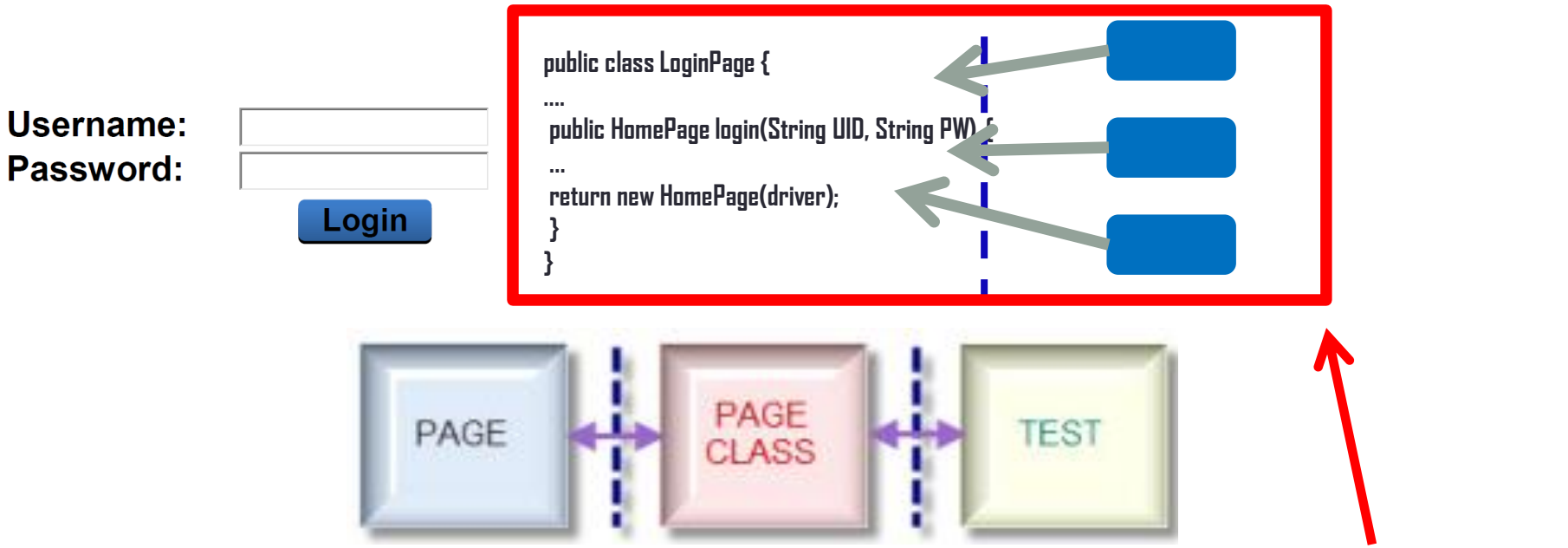
- Allows to insert a **level of abstraction** between the **test cases** and the **web pages**
  - with the aim of **reducing** the **coupling** among them



- First step: **create** a **page class** for **each web page** involved
- Each **method encapsulates** a page's functionality
  - (e.g. Login)

Page Object Pattern

- Allows to insert a **level of abstraction** between the **test cases** and the **web pages**
  - with the aim of **reducing** the **coupling** among them

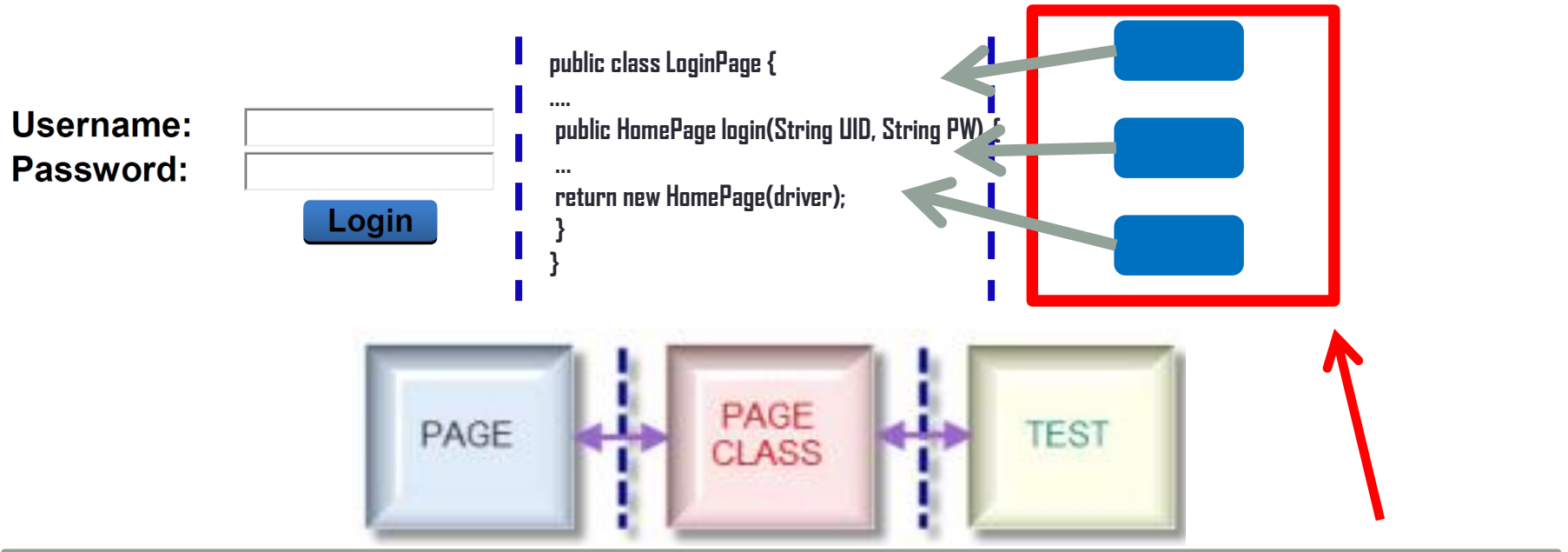


- First step: **cr**
- Each **metho**
  - (e.g. Login)

The same method can be called by several test cases

Page Object Pattern

- Allows to insert a **level of abstraction** between the **test cases** and the **web pages**
  - with the aim of **reducing** the **coupling** among them



- First step: **c**
- Each **meth**
  - (e.g. Login)

In this way the test cases are simplified

No references to the page implementation

Combined Approach and ASSESSOR+

a new approach able to

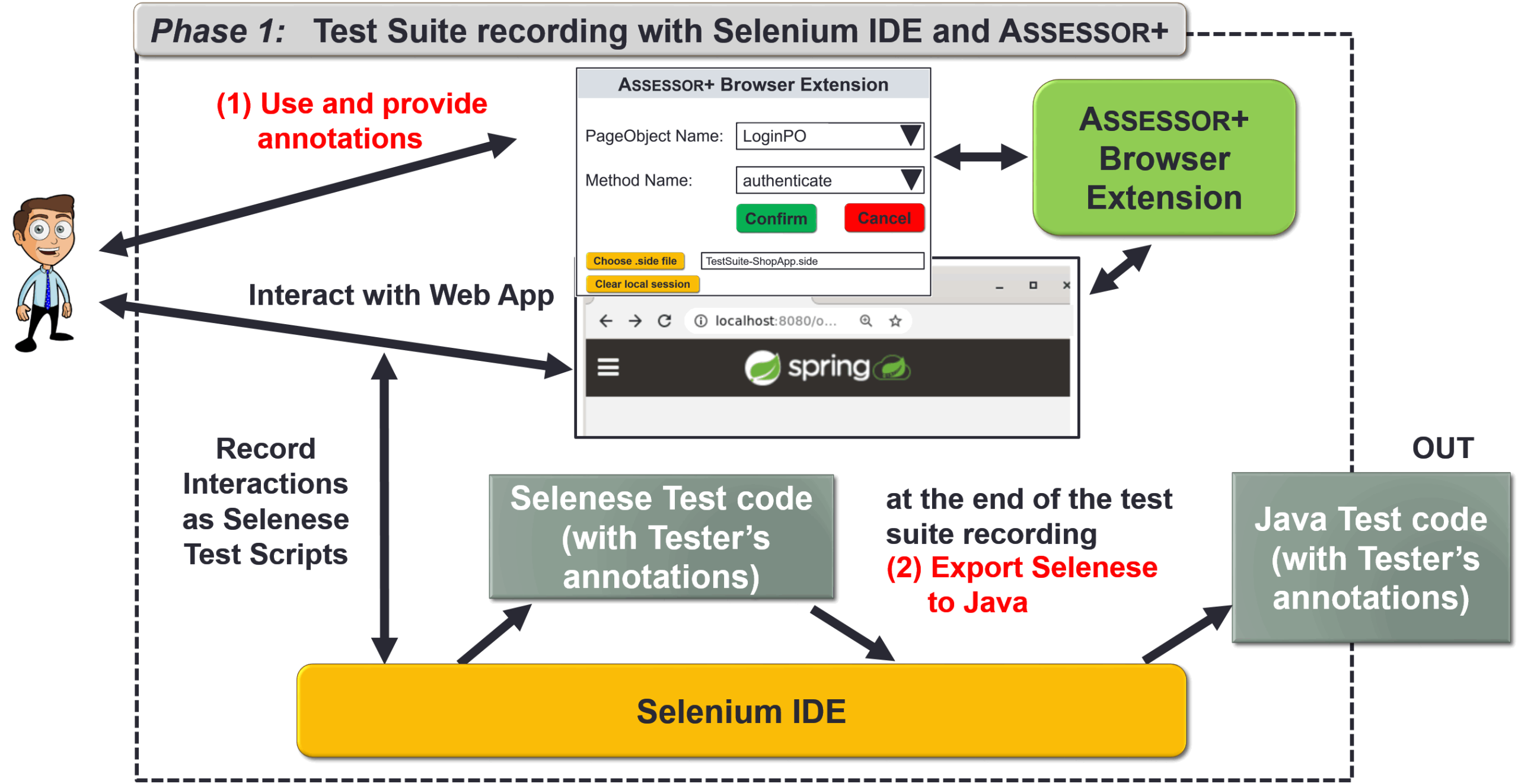
combine the strengths of the **C&R** and **Script-Based** approaches

GOAL: maximize the overall benefits

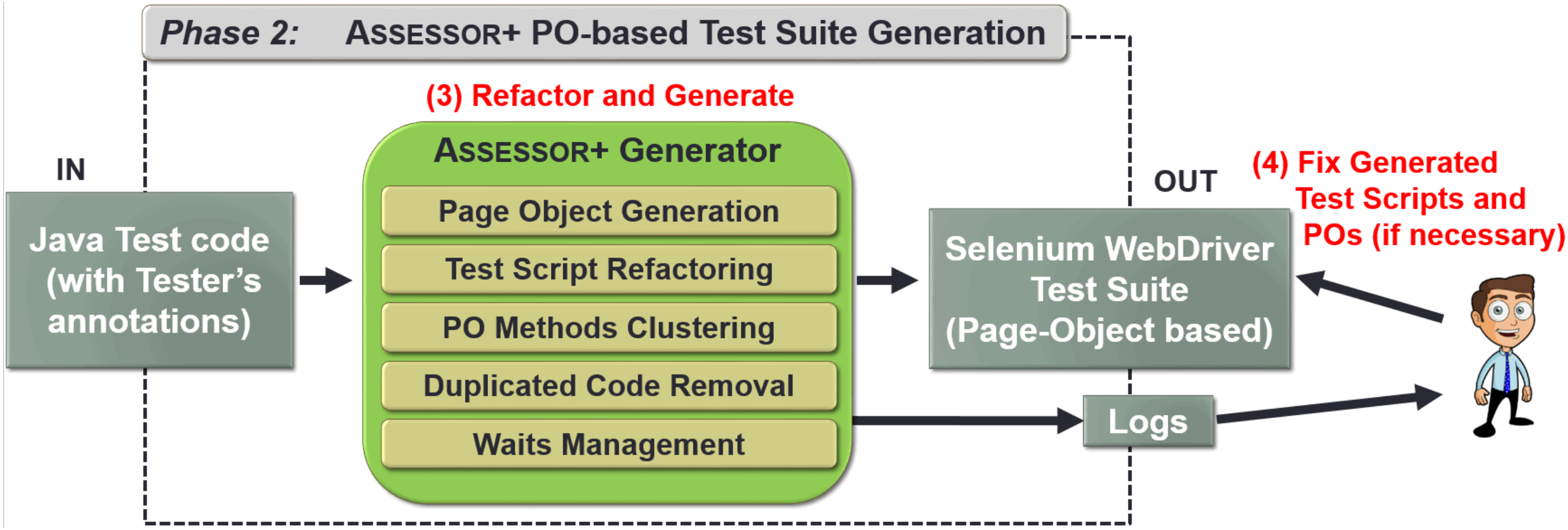
ASSESSOR+

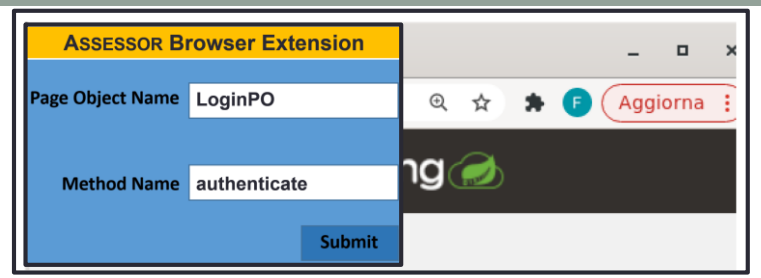
a supporting tool to render the combined approach practical and cost-effective

- Module 1: **ASSESSOR+ Browser Extension** (Selenium IDE plugin)
- Module 2: **ASSESSOR+ Generator** (standalone Java program)


Phase 1: Test Suite recording with Selenium IDE and ASSESSOR+
(1) Use and provide annotations
ASSESSOR+ Browser Extension
PageObject Name:
LoginPO
Method Name:
authenticate
Confirm
Cancel
Choose .side file
TestSuite-ShopApp.side
Clear local session
ASSESSOR+ Browser Extension
Interact with Web App
localhost:8080/o...
spring
Record Interactions as Selenese Test Scripts
Selenese Test code (with Tester's annotations)
at the end of the test suite recording
(2) Export Selenese to Java
OUT
Java Test code (with Tester's annotations)
Selenium IDE

ASSESSOR+


Phase 2: ASSESSOR+ PO-based Test Suite Generation
(3) Refactor and Generate
IN
Java Test code (with Tester's annotations)
ASSESSOR+ Generator
Page Object Generation
Test Script Refactoring
PO Methods Clustering
Duplicated Code Removal
Waits Management
OUT
Selenium WebDriver Test Suite (Page-Object based)
Logs
(4) Fix Generated Test Scripts and POs (if necessary)



(1) Use and provide annotations

Web app under test

Interact

Selenium IDE Plugin + ASSESSOR

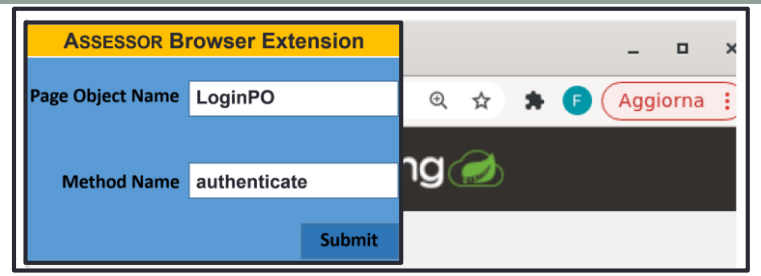
Record test script as Selenese code

userLoginSuccess

Selenium IDE - shopizer – Mozilla Firefox

Command	Target	Value
open	http://localhost:8080/shop/	
echo	{ASSESSOR}:HomePO:goToLogin	
click	id=customerAccount	
click	linkText=Sign in	
echo	{ASSESSOR}:LoginPO:authenticate	
click	id=username	
type	id=username	test@test.it
click	id=password	
type	id=password	test
click	id=loginbutton	
echo	{ASSESSOR}:DashboardPO:checkStatus	
assert element present	linkText=Logout	

Selenese Test code (with Tester's annotations)



(1) Use and provide annotations

Web app under test

Interact

Selenium IDE Plugin + ASSESSOR

Record test script as Selenese code

(2) Export Selenese to Java

userLoginSuccess

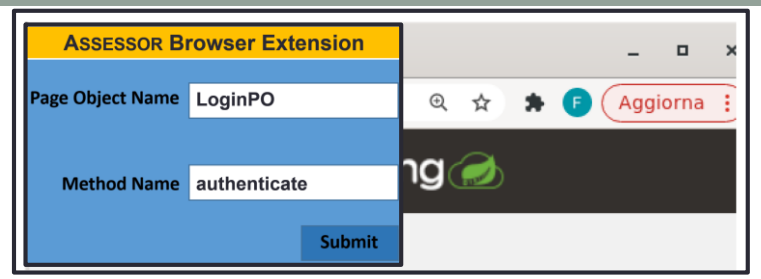
Selenium IDE - shopizer – Mozilla Firefox

Command	Target	Value
open	http://localhost:8080/shop/	
echo	{ASSESSOR}:HomePO:goToLogin	
click	id=customerAccount	
click	linkText=Sign in	
echo	{ASSESSOR}:LoginPO:authenticate	
click	id=username	
type	id=username	test@test.it
click	id=password	
type	id=password	test
click	id=loginbutton	
echo	{ASSESSOR}:DashboardPO:checkStatus	
assert element present	linkText=Logout	

Selenese Test code (with Tester's annotations)

```
@Test
public void userLoginSuccess() {
 driver.get("http://localhost:8080/shop/");
 System.out.println("{ASSESSOR}:HomePO:goToLogin");
 driver.findElement(By.id("customerAccount")).click();
 driver.findElement(By.linkText("Sign in")).click();
 System.out.println("{ASSESSOR}:LoginPO:authenticate");
 driver.findElement(By.id("username")).click();
 driver.findElement(By.id("username")).sendKeys("test@test.it");
 driver.findElement(By.id("password")).click();
 driver.findElement(By.id("password ")).sendKeys("test");
 driver.findElement(By.id("loginButton")).click();
 System.out.println("{ASSESSOR}:DashboardPO:checkStatus");
 {
  List<WebElement> e = driver.findElements(By.linkText("Logout"));
  assert(e.size() > 0);
 }
}
```

Java Test code (with Tester's annotations)



Web app under test

Interact

(1) Use and provide annotations

Selenium IDE Plugin + ASSESSOR

ASSESSOR Generator

Logs

Record test script as Selenese code

(2) Export Selenese to Java

(3) Refactor

userLoginSuccess

Selenium IDE - shopizer – Mozilla Firefox

Command	Target	Value
open	http://localhost:8080/shop/	
echo	{ASSESSOR}:HomePO:goToLogin	
click	id=customerAccount	
click	linkText=Sign in	
echo	{ASSESSOR}:LoginPO:authenticate	
click	id=username	
type	id=username	test@test.it
click	id=password	
type	id=password	test
click	id=loginbutton	
echo	{ASSESSOR}:DashboardPO:checkStatus	
assert element present	linkText=Logout	

Selenese Test code (with Tester's annotations)

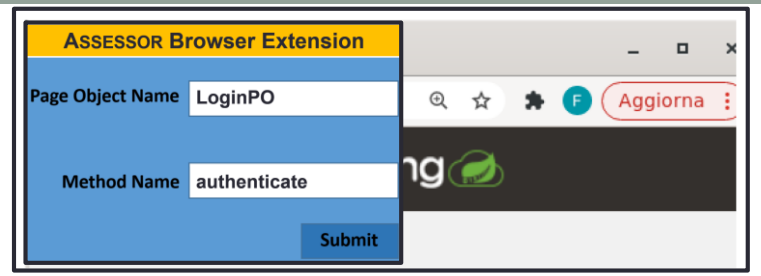
```
@Test
public void userLoginSuccess() {
 driver.get("http://localhost:8080/shop/");
 System.out.println("{ASSESSOR}:HomePO:goToLogin");
 driver.findElement(By.id("customerAccount")).click();
 driver.findElement(By.linkText("Sign in")).click();
 System.out.println("{ASSESSOR}:LoginPO:authenticate");
 driver.findElement(By.id("username")).click();
 driver.findElement(By.id("username")).sendKeys("test@test.it");
 driver.findElement(By.id("password")).click();
 driver.findElement(By.id("password ")).sendKeys("test");
 driver.findElement(By.id("loginButton")).click();
 System.out.println("{ASSESSOR}:DashboardPO:checkStatus");
 {
  List<WebElement> e = driver.findElements(By.linkText("Logout"));
  assert(e.size() > 0);
 }
}
```

Java Test code (with Tester's annotations)

```
@Test
public void userLoginSuccess() {
 driver.get("http://localhost:8080/shop/");
 HomePO homePO = new HomePO(driver, js, vars);
 homePO.goToLogin();
 LoginPO loginPO =  new LoginPO(driver, js, vars);
 loginPO.authenticate("test@test.it", "test");
 DashboardPO dashboardPO =
                          new DashboardPO(driver, js, vars);
 assert (dashboardPO.getListLinkText_Logout().size() > 0);
}
```

```
public class LoginPO {
 ...
 public void authenticate(String key1, String key2) {
  // wait for the first element to interact with
  By elem = By.id("username");
  MyUtils.WaitForElementLoaded(driver, elem);
  // and then continue
  driver.findElement(By.id("username")).click();
  driver.findElement(By.id("username")).sendKeys(key1);
  driver.findElement(By.id("password")).click();
  driver.findElement(By.id("password")).sendKeys(key2);
  driver.findElement(By.id("loginButton")).click();
 }
 ...
}
```

Selenium WebDriver Test Suite (Page-Object based)



Web app under test

Interact

(1) Use and provide annotations

(4) Modify Generated Test Scripts and POs (if necessary)

Selenium IDE Plugin + ASSESSOR

ASSESSOR Generator

Logs

Record test script as Selenese code

(2) Export Selenese to Java

(3) Refactor

userLoginSuccess

Selenium IDE - shopizer – Mozilla Firefox

Command	Target	Value
open	http://localhost:8080/shop/	
echo	{ASSESSOR}:HomePO:goToLogin	
click	id=customerAccount	
click	linkText=Sign in	
echo	{ASSESSOR}:LoginPO:authenticate	
click	id=username	
type	id=username	test@test.it
click	id=password	
type	id=password	test
click	id=loginbutton	
echo	{ASSESSOR}:DashboardPO:checkStatus	
assert element present	linkText=Logout	

Selenese Test code (with Tester's annotations)

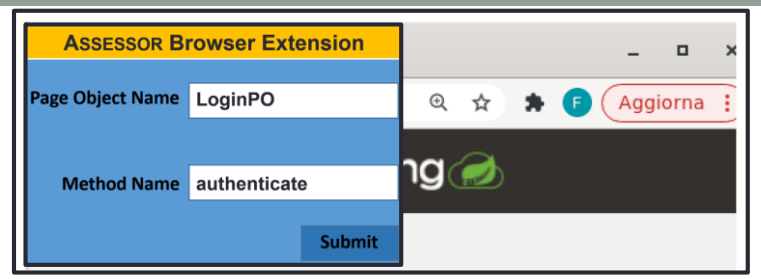
```
@Test
public void userLoginSuccess() {
 driver.get("http://localhost:8080/shop/");
 System.out.println("{ASSESSOR}:HomePO:goToLogin");
 driver.findElement(By.id("customerAccount")).click();
 driver.findElement(By.linkText("Sign in")).click();
 System.out.println("{ASSESSOR}:LoginPO:authenticate");
 driver.findElement(By.id("username")).click();
 driver.findElement(By.id("username")).sendKeys("test@test.it");
 driver.findElement(By.id("password")).click();
 driver.findElement(By.id("password ")).sendKeys("test");
 driver.findElement(By.id("loginButton")).click();
 System.out.println("{ASSESSOR}:DashboardPO:checkStatus");
 {
  List<WebElement> e = driver.findElements(By.linkText("Logout"));
  assert(e.size() > 0);
 }
}
```

Java Test code (with Tester's annotations)

```
@Test
public void userLoginSuccess() {
 driver.get("http://localhost:8080/shop/");
 HomePO homePO = new HomePO(driver, js, vars);
 homePO.goToLogin();
 LoginPO loginPO =  new LoginPO(driver, js, vars);
 loginPO.authenticate("test@test.it", "test");
 DashboardPO dashboardPO =
                          new DashboardPO(driver, js, vars);
 assert (dashboardPO.getListLinkText_Logout().size() > 0);
}
```

```
public class LoginPO {
 ...
 public void authenticate(String key1, String key2) {
  // wait for the first element to interact with
  By elem = By.id("username");
  MyUtils.WaitForElementLoaded(driver, elem);
  // and then continue
  driver.findElement(By.id("username")).click();
  driver.findElement(By.id("username")).sendKeys(key1);
  driver.findElement(By.id("password")).click();
  driver.findElement(By.id("password")).sendKeys(key2);
  driver.findElement(By.id("loginButton")).click();
 }
 ...
}
```

Selenium WebDriver Test Suite (Page-Object based)



Web app under test

Interact

(1) Use and provide annotations

(4) Modify Generated Test Scripts and POs (if necessary)

Selenium IDE Plugin + ASSESSOR

ASSESSOR Generator

Logs

Record test script as Selenese code

(2) Export Selenese to Java

(3) Refactor

userLoginSuccess

Selenium IDE - shopizer – Mozilla Firefox

Command	Target	Value
open	http://localhost:8080/shop/	
echo	{ASSESSOR}:HomePO:goToLogin	
click	id=customerAccount	
click	linkText=Sign in	
echo	{ASSESSOR}:LoginPO:authenticate	
click	id=username	
type	id=username	test@test.it
click	id=password	
type	id=password	test
click	id=loginbutton	
echo	{ASSESSOR}:DashboardPO:checkStatus	
assert element present	linkText=Logout	

```
@Test
public void userLoginSuccess() {
 driver.get("http://localhost:8080/shop/");
 System.out.println("{ASSESSOR}:HomePO:goToLogin");
 driver.findElement(By.id("customerAccount")).click();
 driver.findElement(By.linkText("Sign in")).click();
 System.out.println("{ASSESSOR}:LoginPO:authenticate");
 driver.findElement(By.id("username")).click();
 driver.findElement(By.id("username")).sendKeys("test@test.it");
 driver.findElement(By.id("password")).click();
 driver.findElement(By.id("password ")).sendKeys("test");
 driver.findElement(By.id("loginButton")).click();
 System.out.println("{ASSESSOR}:DashboardPO:checkStatus");
 {
  List<WebElement> e = driver.findElements(By.linkText("Logout"));
  assert(e.size() > 0);
 }
}
```

```
@Test
public void userLoginSuccess() {
 driver.get("http://localhost:8080/shop/");
 HomePO homePO = new HomePO(driver, js, vars);
 homePO.goToLogin();
 LoginPO loginPO =  new LoginPO(driver, js, vars);
 loginPO.authenticate("test@test.it", "test");
 DashboardPO dashboardPO =
                    new DashboardPO(driver, js, vars);
 assert (dashboardPO.getListLinkText_Logout().size() > 0);
}
```

```
public class LoginPO {
...
public void authenticate(String key1, String key2) {
 // wait for the first element to interact with
 By elem = By.id("username");
 MyUtils.WaitForElementLoaded(driver, elem);
 // and then continue
 driver.findElement(By.id("username")).click();
 driver.findElement(By.id("username")).sendKeys(key1);
 driver.findElement(By.id("password")).click();
 driver.findElement(By.id("password")).sendKeys(key2);
 driver.findElement(By.id("loginButton")).click();
}
...
}
```

Selenese Test code (with Tester's annotations)

Java Test code (with Tester's annotations)

Selenium WebDriver Test Suite (Page-Object based)

Assignment

Goal

Investigate the ***impact of adopting ASSESSOR+*** on the development of Selenium WebDriver PO-based Java E2E test suite compared to the classical Manual development

Apps

- **4 Apps** from a **GitHub repository** where you can find:
  - **Docker image** of the each app already available (Bludit, MantisBT, Prestashop, Claroline)
  - **Gherkin descriptions** of the test scripts to implement

https://github.com/SoftEng-UniGE/BEWT-Specifications

Approaches compared (Treatments)

- **MANUAL**
  - full development of a Selenium WebDriver PO-based Java E2E test suite in the IDE
- **ASSESSOR+**
  - ***record*** the test suite in Selenium IDE while annotating the test scripts with ASSESSOR+ Plugin
  - ***export*** the Java code
  - ***generate*** the PO-based test suite with ASSESSOR+ Generator, and
  - ***refine*** the test scripts if needed

Procedure

Students are assigned to two Groups
Each **student develops each of the four test suite with ONLY one approach and following this order:**

Group 1

ASSESSOR+	Manual	ASSESSOR+	Manual
Bludit	MantisBT	Prestashop	Claroline

Group 2

Manual	ASSESSOR+	Manual	ASSESSOR+
Prestashop	Claroline	Bludit	MantisBT

We want **to measure the time required to develop each test script**

So for each test script to implement:
1. read the Gherkin test case specification, and

If you are **developing a test script MANUALLY**:
2. save the **start time**
3. develop the test script and when it works
4. save the **stop time**
5. move to the next test script of the test suite

Procedure

If you are **developing a test suite with ASSESSOR+:**

PHASE 1: (Test Suite recording with Selenium IDE and ASSESSOR+ Plugin)

2. save the **start time (recording)**
3. record the test script with Selenium IDE and ASSESSOR+ Plugin
- (i.e., place the PO annotations during the recording). When the recording of the test script is completed:
4. save the **stop time (recording)**
5. move to the next test script

PHASE 2: Export and Generation

When the entire test suite is recorded, export the test suite in Java (using Selenium IDE), run ASSESSOR+ Generator in order to generate the PO-based test suite, and start to refine the test scripts (if needed), so:

PHASE 3: Test Suite Refinement

6. save the **start time (refinement)**
7. and then refine the first test script and when it works
8. save the **stop time (refinement)**
9. move to the next test script

ASSESSOR+
Tutorial and Demo