

| Podstawy Selenium

Agenda

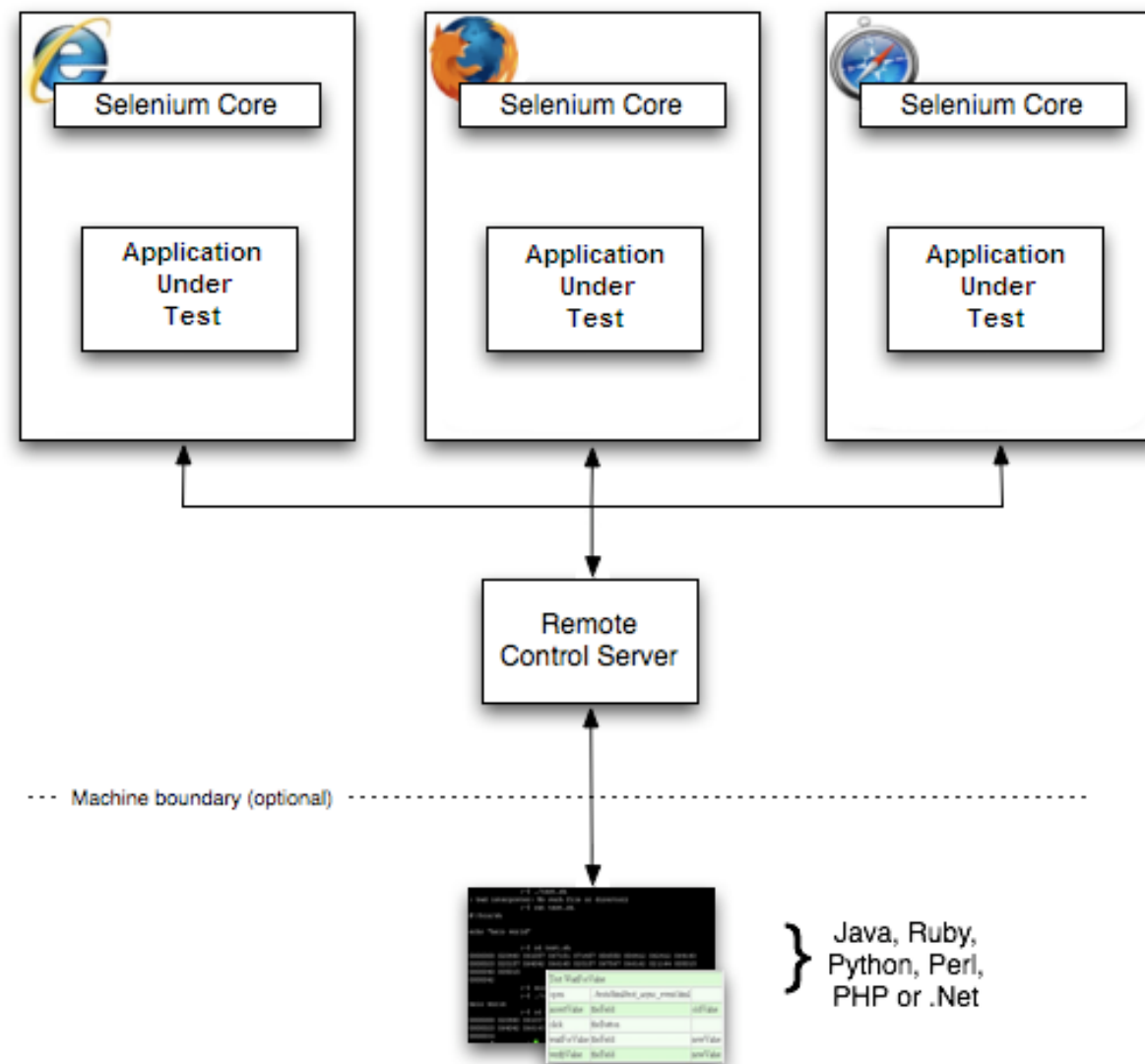
- Czym jest Selenium
- Jak działa Selenium
- Lokatory
- Xpath - podstawy
- Niektóre funkcje Selenium
- Waits
- Page Object / Page Factory
- Data Driven Testing
- Pytania rekrutacyjne z Selenium



Selenium

- Darmowe narzędzie do automatyzowania akcji po stronie przeglądarki internetowej.
- Języki (Java, C#, Python, Ruby, Perl, PHP)
- Framework działający z innymi frameworkami
- Przyjazne API

Windows, Linux, or Mac (as appropriate)...



Selenium WebDriver

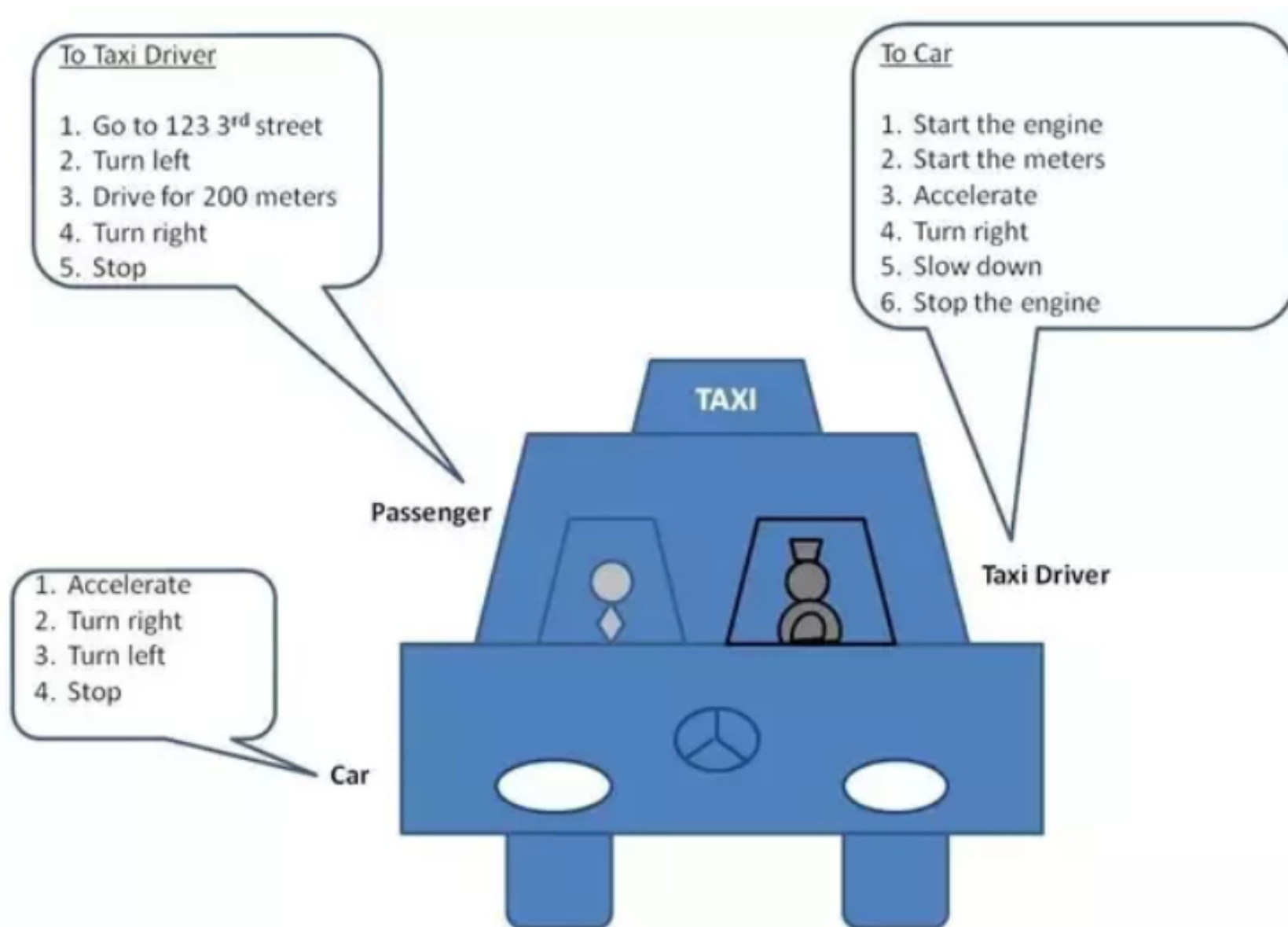


Selenium IDE

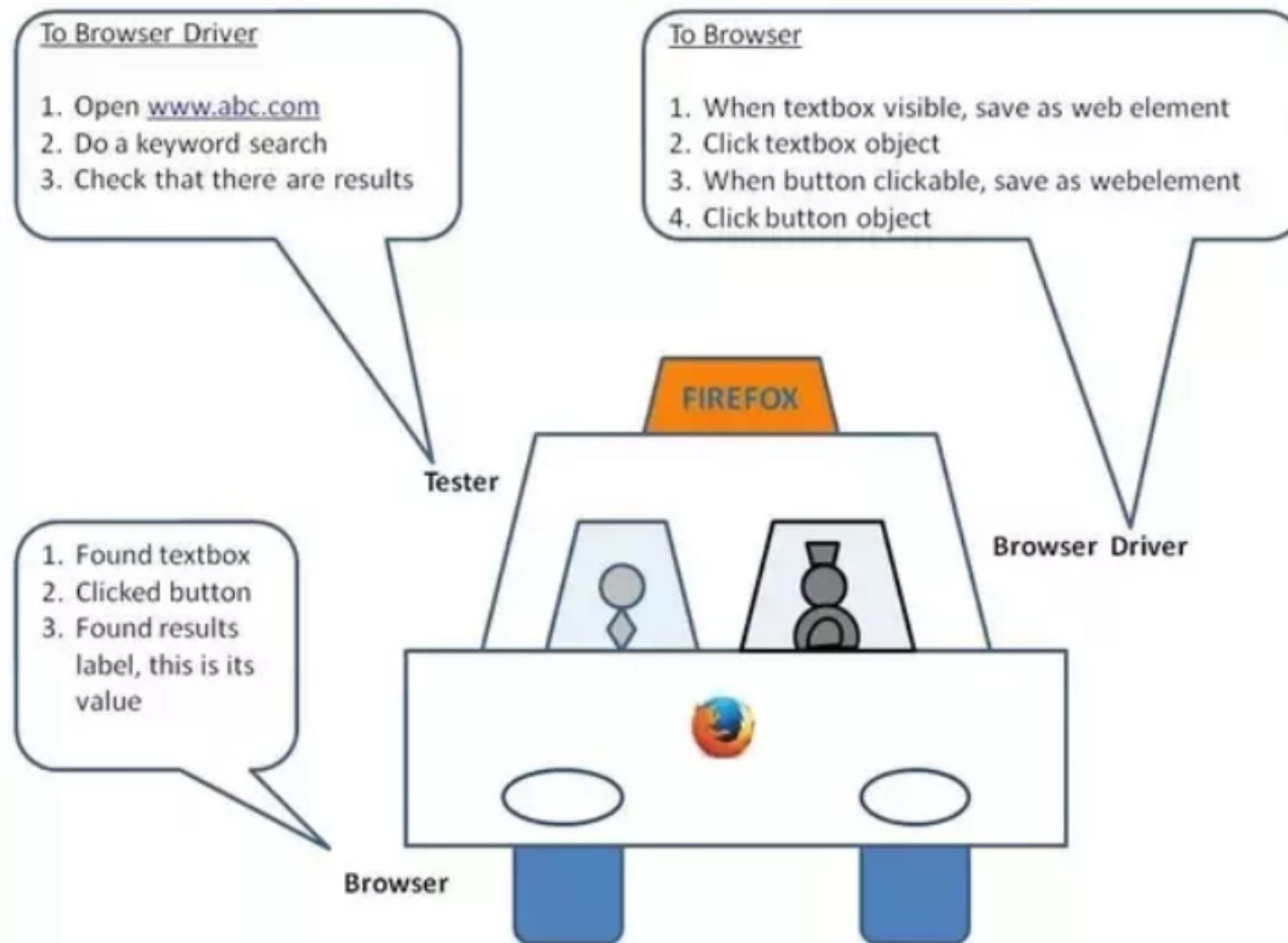


Jak działa Selenium

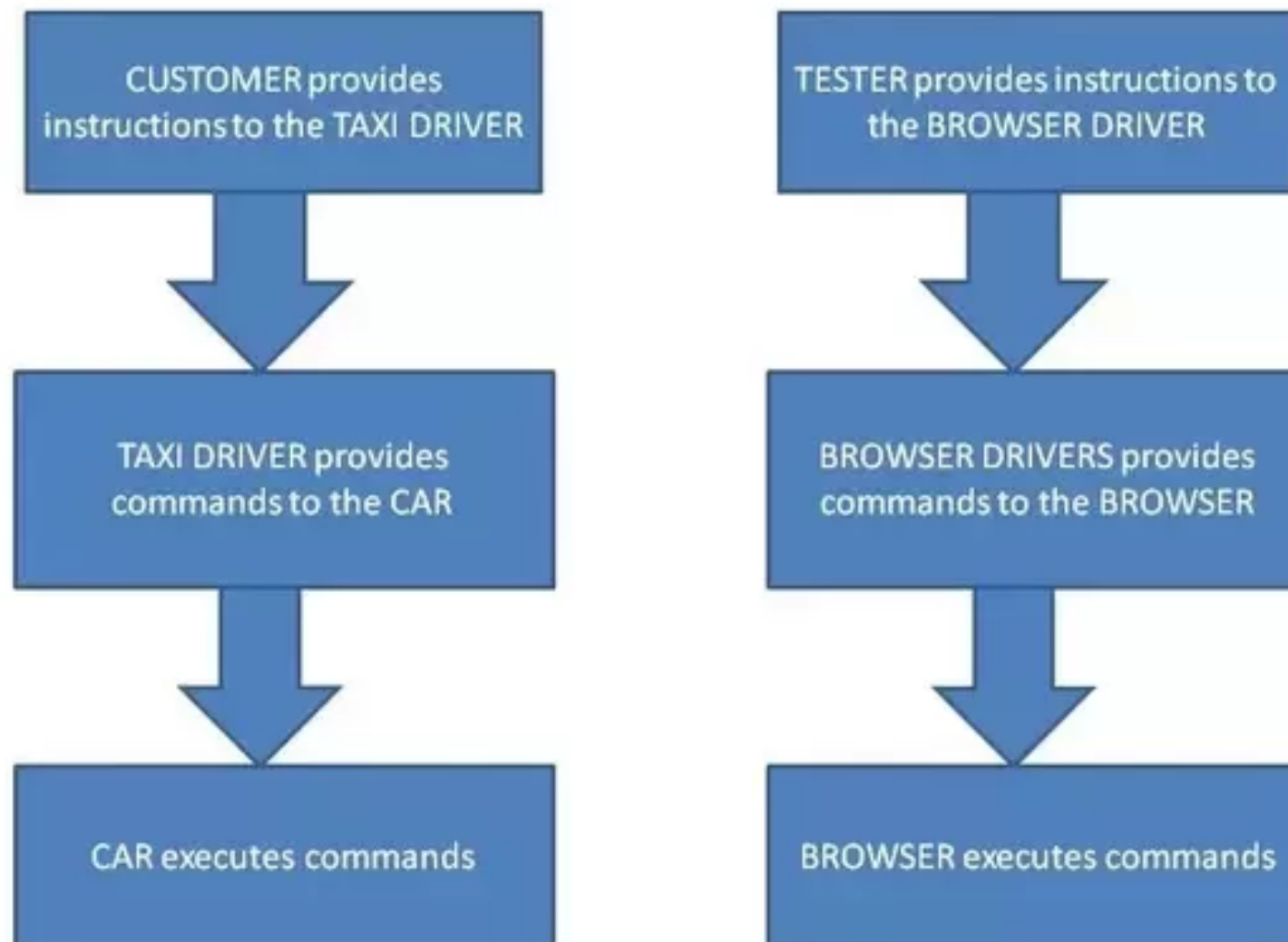
Jak działa Selenium



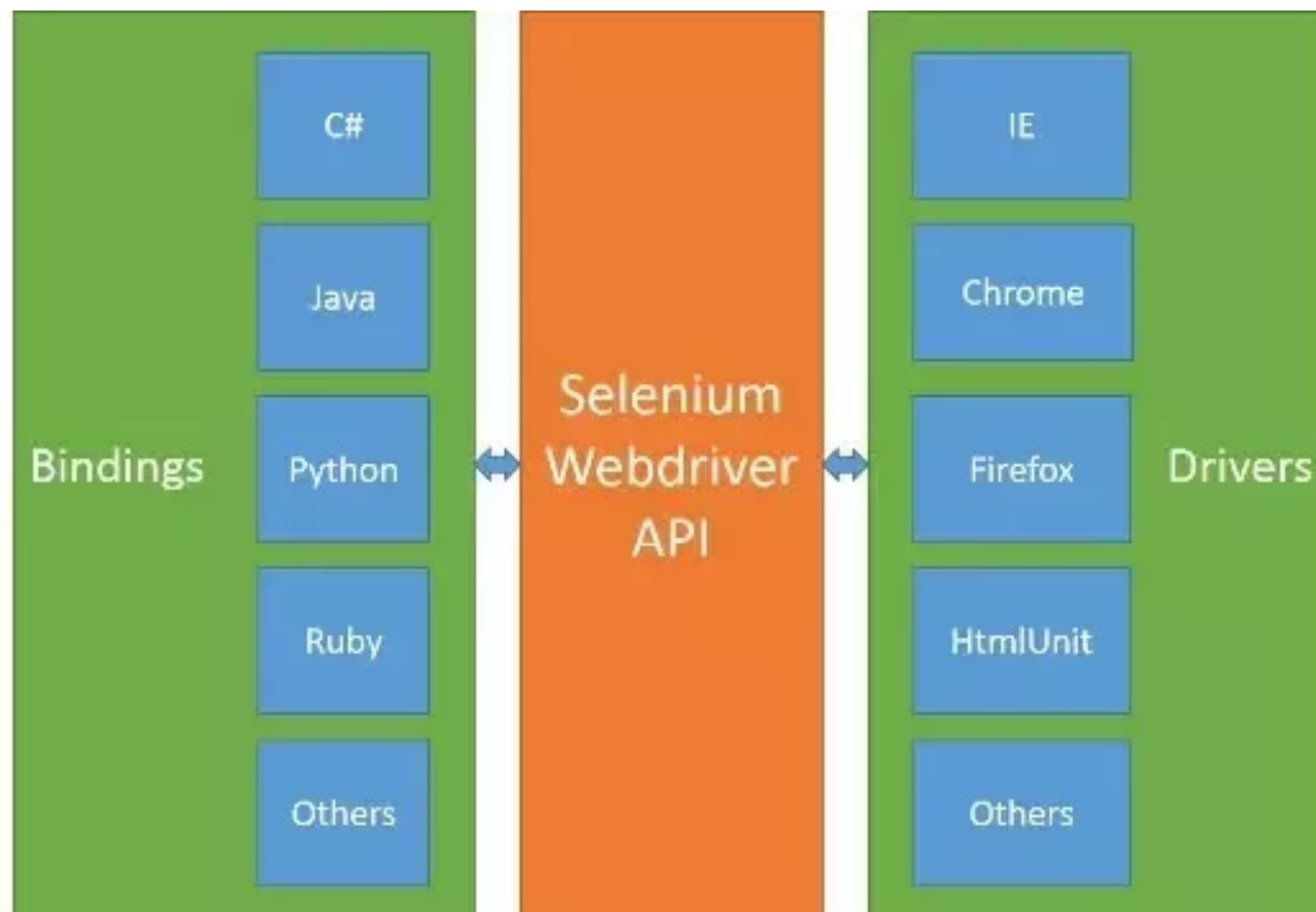
Jak działa Selenium



Jak działa Selenium



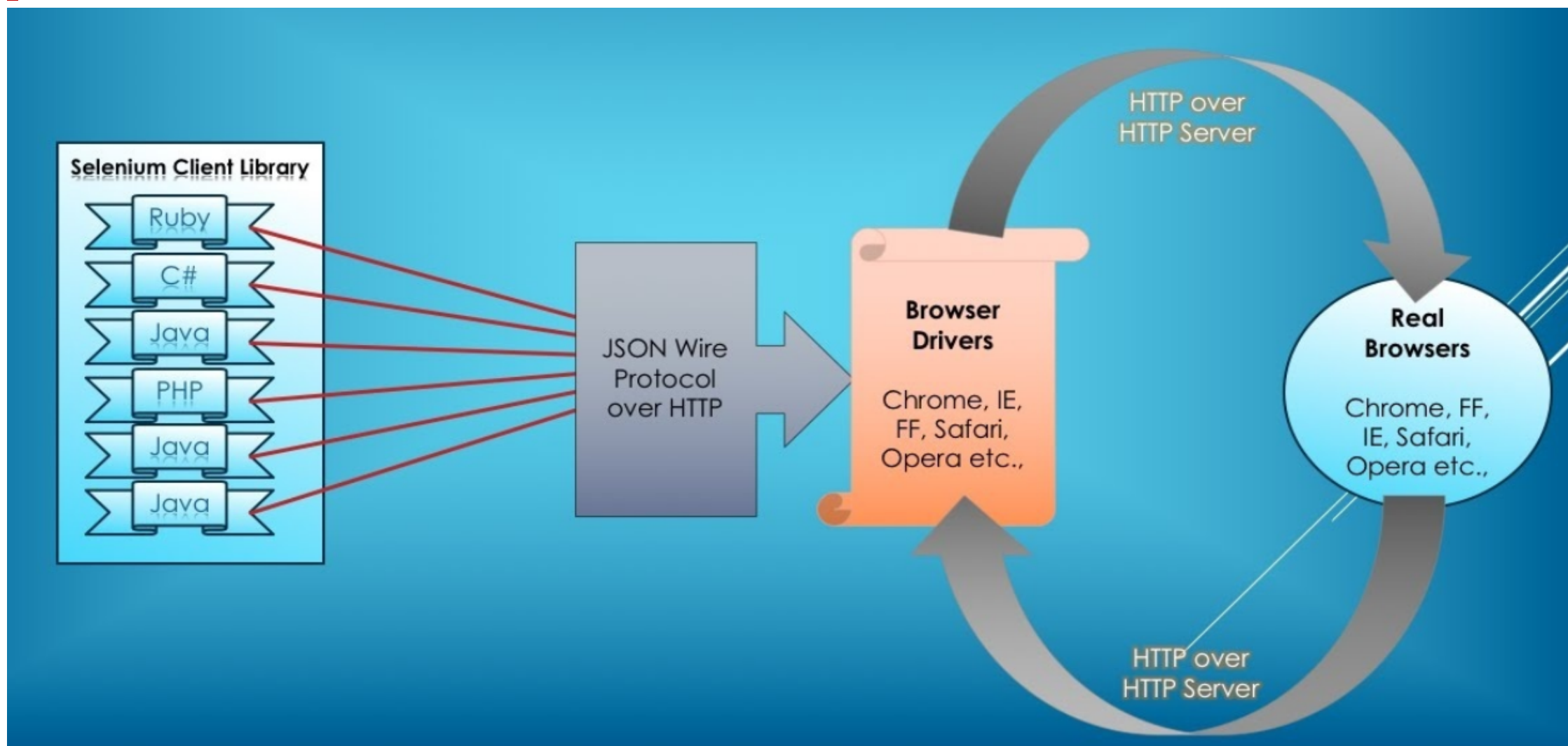
Jak działa Selenium



Jak działa Selenium

- Dla każdej komendy Selenium, jest tworzony HTTP request
- Request jest wysyłany do drivera przeglądarki
- Driver używa HTTP serwera by otrzymać request
- Serwer określa kroki zaimplementowane w kodzie
- Zaimplementowane kroki są wykonywane w przeglądarce
- Status wykonania wysyłany jest z powrotem do HTTP serwera
- Serwer HTTP wysyła status do skryptu testu automatycznego.

Jak działa Selenium



Lokatory

Lokatory

```
<div id="divontheleft" class="leftdiv">
  <input id="but1" value="Button with ID" type="button"></input>
  <br/>
</div>
```

- ID
- Name
- Link
- PartialLink
- Xpath
- CSS
- ClassName

```
<input name="but2" value="Button with name" type="button"></input>
```

```
<a href="link.html">Name of the Link</a>
```

```
xpath=//*[@id='username']
```

```
xpath=//input[@id='username']
```

```
xpath=//form[@name='loginForm']/input[1]
```

```
xpath=//*[@name='loginForm']/input[1]
```



**KEEP
CALM**

AND

**LETS WRITE
SOME CODE**

KeepCalmAndPosters.com

Zadanie #1

- Utwórz nowy projekt mavenowy.
- Napisz pierwszy test Selenium otwierający stronę google.com i sprawdzający czy jest na właściwej stronie.

Czas: 0.5h

XPath - podstawy

XPath – podstawy #1

- XPath to inaczej XML Path Language (Język ścieżek XML).
- Pozwala w elastyczny sposób wskazywać różne części dokumentu XML.
- XPath używa notacji ścieżkowej (ang. path notation) (tak jak adresy URL) do nawigacji po hierarchicznej strukturze dokumentu XML.



XPath – podstawy #2

- „/” - Ścieżka absolutna do elementu
 <AAA>
 <BBB/>
 <CCC/>
 <BBB/>
 <BBB/>
 <DDD>
 <BBB/>
 </DDD>
 <CCC/>
 </AAA>

/AAA/DDD/BBB

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Wybiera wszystkie te elementy BBB, które są dziećmi DDD, które z kolei są dziećmi elementu głównego AAA

XPath – podstawy #3

- „//” - Wybiera wszystkie elementy spełniające warunek po „//”

```
<AAA>  
  <BBB/>  
  <CCC/>  
  <BBB/>  
  <DDD>  
    <BBB/>  
  </DDD>  
  <CCC>  
    <DDD>  
      <BBB/>  
      <BBB/>  
    </DDD>  
  </CCC>  
</AAA>
```

//DDD/BBB

XPath – podstawy #4

- **/AAA/CCC/DDD/*** - Wybiera wszystkie elementy wewnątrz / AAA/CCC/DDD
- **/*/*/*/BBB** - Wybiera wszystkie te elementy BBB, które mają 3 przodków
- **//*** - Wybiera wszystkie elementy
- **/AAA/BBB[1]** - Wybiera pierwszy element BBB będący dzieckiem elementu AAA
- **/AAA/BBB[last()]** - Wybiera ostatni element BBB będący dzieckiem elementu AAA

```
<AAA>  
  <BBB/>  
  <BBB/>  
  <BBB/>  
  <BBB/>  
</AAA>
```

XPath – podstawy #5

- **//*[@id]** - Wybiera wszystkie atrybuty id
- **//BBB[@id]** - Wybiera elementy BBB posiadające atrybut id
- **//BBB[@*]** - Wybiera elementy BBB posiadające dowolny atrybut
- **//BBB[not(@*)]** - Wybiera elementy BBB nie posiadające ani jednego atrybutu

```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```

```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```

```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```

```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```

XPath – podstawy #6

- **//BBB[@id='b1']** - Wybiera te elementy BBB, które posiadają atrybut id z przypisaną wartością "b1"
- **//BBB[@name='bbb']** - Wybiera te elementy BBB, które posiadają atrybut name z przypisaną wartością "bbb"
- **//BBB[normalize-space(@name)='bbb']** - Wybiera elementy BBB które mają atrybuty "name" z przypisaną wartością "bbb". Przed porównaniem usuwa spacje z początku i końca porównywanej wartości.

```
<AAA>  
  <BBB id = "b1"/>  
  <BBB name = " bbb "/>  
  <BBB name = "bbb"/>  
</AAA>
```

```
<AAA>  
  <BBB id = "b1"/>  
  <BBB name = " bbb "/>  
  <BBB name = "bbb"/>  
</AAA>
```

```
<AAA>  
  <BBB id = "b1"/>  
  <BBB name = " bbb "/>  
  <BBB name = "bbb"/>  
</AAA>
```

XPath – podstawy #7

- **`//*[count(BBB)=2]`** - Wybiera elementy z dwojgiem dzieci BBB
- **`//*[count(*)=2]`** - Wybiera elementy z dwojgiem dzieci
- **`//*[count(*)=3]`** - Wybiera elementy mające troje dzieci

<AAA>	<AAA>	<AAA>
<CCC>	<CCC>	<CCC>
<BBB/>	<BBB/>	<BBB/>
<BBB/>	<BBB/>	<BBB/>
<BBB/>	<BBB/>	<BBB/>
</CCC>	</CCC>	</CCC>
<DDD>	<DDD>	<DDD>
<BBB/>	<BBB/>	<BBB/>
<BBB/>	<BBB/>	<BBB/>
</DDD>	</DDD>	</DDD>
<EEE>	<EEE>	<EEE>
<CCC/>	<CCC/>	<CCC/>
<DDD/>	<DDD/>	<DDD/>
</EEE>	</EEE>	</EEE>
</AAA>	</AAA>	</AAA>

XPath – podstawy #8

- **`//*[name()='BBB']`** - Wybiera wszystkie elementy o nazwie BBB;
zapis równoważny z `//BBB`
- **`//*[starts-with(name(),'B')]`** - Wybiera wszystkie te elementy,
których nazwy zaczynają się literą B
- **`//*[contains(name(),'C')]`** - Wybiera wszystkie te elementy,
których nazwy zawierają w sobie literę C

```
<AAA>
  <BCC>
    <BBB/>
    <BBB/>
    <BBB/>
  </BCC>
  <DDB>
    <BBB/>
    <BBB/>
  </DDB>
  <BEC>
    <CCC/>
    <DBD/>
  </BEC>
</AAA>
```

```
<AAA>
  <BCC>
    <BBB/>
    <BBB/>
    <BBB/>
  </BCC>
  <DDB>
    <BBB/>
    <BBB/>
  </DDB>
  <BEC>
    <CCC/>
    <DBD/>
  </BEC>
</AAA>
```

XPath – podstawy #9

- **`//*[string-length(name()) = 3]`** - Wybiera elementy o nazwach złożonych z trzech znaków
- **`//*[string-length(name()) < 3]`** - Wybiera elementy o jedno lub dwuznakowych nazwach
- **`//*[string-length(name()) > 3]`** - Wybiera elementy o nazwach dłuższych niż trzy znaki
- **`//CCC | //BBB`** - Wybiera wszystkie elementy CCC oraz BBB
- **`/AAA/EEE | //DDD/CCC | /AAA | //BBB`** - Nic nie ogranicza liczby kombinacji

```
<AAA>
  <BBB/>
  <CCC/>
  <DDD>
    <CCC/>
  </DDD>
  <EEE/>
</AAA>
```

```
<AAA>
  <BBB/>
  <CCC/>
  <DDD>
    <CCC/>
  </DDD>
  <EEE/>
</AAA>
```


XPath – podstawy #11

- **//FFF/ancestor::*** - Wybiera przodków elementów FFF
- **/AAA/BBB/following-sibling::*** - Oś "following-sibling" obejmuje rodzeństwo (elementy mające wspólnego rodzica), następujące po węźle w kontekście którego występuje.

```

<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
      <EEE>
        <DDD>
          <FFF/>
        </DDD>
      </EEE>
    </DDD>
  </CCC>
</AAA>

```

```

<AAA>
  <BBB>
    <CCC/>
    <DDD/>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
      <CCC/>
      <FFF/>
      <FFF>
        <GGG/>
      </FFF>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>

```

XPath – podstawy #12

- **/AAA/XXX/following::*** - Oś "following" obejmuje wszystkie węzły następujące po węźle w kontekście którego występuje. Wyłączone zostają węzły potomne tego elementu, węzły atrybutów oraz przestrzeni nazw.

```
<AAA>
  <BBB>
    <CCC/>
    <ZZZ>
      <DDD/>
      <DDD>
        <EEE/>
      </DDD>
    </ZZZ>
    <FFF>
      <GGG/>
    </FFF>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
      <CCC/>
      <FFF/>
      <FFF>
        <GGG/>
      </FFF>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```

XPath – podstawy #13

- **//BBB[$\text{position() mod } 2 = 0$]** - Wybiera elementy BBB na parzystych pozycjach
- **//BBB[$\text{position() = floor(last() div } 2 + 0.5)$ or $\text{position() = ceiling(last() div } 2 + 0.5)$]** - Wybiera środkowy element (lub elementy) BBB
- **//CCC[$\text{position() = floor(last() div } 2 + 0.5)$ or $\text{position() = ceiling(last() div } 2 + 0.5)$]** - Wybiera środkowy element (lub elementy) CCC

<AAA>	<AAA>	<AAA>
<BBB/>	<BBB/>	<BBB/>
< BBB />	<BBB/>	<BBB/>
<BBB/>	<BBB/>	<BBB/>
< BBB />	< BBB />	<BBB/>
<BBB/>	< BBB />	<BBB/>
< BBB />	<BBB/>	<BBB/>
<BBB/>	<BBB/>	<BBB/>
< BBB />	<BBB/>	<BBB/>
<CCC/>	<CCC/>	<CCC/>
<CCC/>	<CCC/>	< CCC />
<CCC/>	<CCC/>	<CCC/>
</AAA>	</AAA>	</AAA>

XPath – przykłady z projektów

- **`//div[@class = 'text']`** - elementy typu div, zawierające atrybut `class = text`
- **`//div[text() = 'Login']`** - elementy typu div o tekście "login"
- **`//div[contains(@title, 'Title')]/following-sibling::div/input`** - pole typu input, obok elementu div zawierającego atrybut `title = Title`
- **`//label[text() = 'Single']/following-sibling::input[@xpath = '1']`** - pole typu input, obok elementu label zawierającego o tekście "Single"
- **`//div[@class = 'mgnlEditorBarLabel' and @title = 'Header Main Menu']`** - elementy typu div z dwoma atrybutami
- **`//div[@id='acceptTerms-input-holder']/div[@class='checkbox']/*[not(../a)]`** - gdy element ma zagnieżdżony link, którego nie chcemy klikać.

Zadanie #2

- Napisz 20 Xpath selektorów do elementów na stronie i stronie logowania:

<https://www.phptravels.net/admin>

Email admin@phptravels.com

Password demoadmin

- Do zweryfikowania selektorów wykorzystać możesz narzędzie ChroPath lub terminal przeglądarki: `$x("//img[@id='hplogo']")`

Czas: 1h

Niektóre funkcje Selenium



**KEEP
CALM**

AND

**LETS WRITE
SOME CODE**

Sprawdzanie atrybutu

```
@Test
public void testElementAttribute()
{
    WebElement message = driver.findElement(By.id("message"));
    assertEquals("justify",message.getAttribute("align"));
}
```

Sprawdzanie wartości CSS

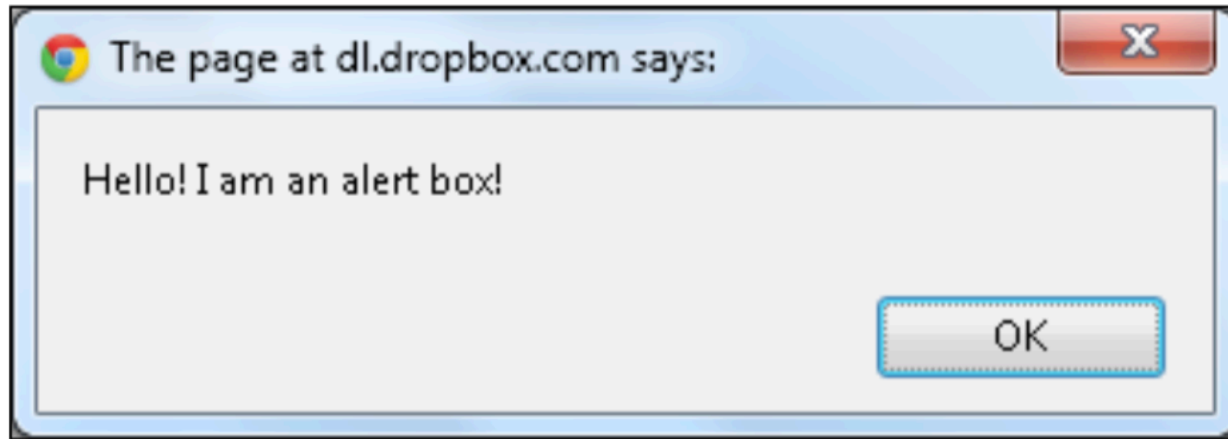
```
@Test
public void testElementStyle()
{
    WebElement message = driver.findElement(By.id("message"));
    String width = message.getCssValue("width");
    assertEquals("150px",width);
}
```

Przełączanie się między oknami

```
//Save the WindowHandle of Parent Browser Window  
String currentWindowId = driver.getWindowHandle();
```

```
Set<String> allWindows = driver.getWindowHandles();  
if(!allWindows.isEmpty()) {  
    for (String windowId : allWindows) {  
        driver.switchTo().window(windowId);  
    }  
}
```

Zamykanie alertów JSowych



```
//Get the Alert  
Alert alert = driver.switchTo().alert();
```

```
//Get the Text displayed on Alert using getText() method of  
Alert class  
String textOnAlert = alert.getText();
```

```
//Click OK button, by calling accept() method of Alert Class  
alert.accept();
```

Obsługa confirm boxów



```
//Get the Alert
Alert alert = driver.switchTo().alert();

//Click OK button, by calling accept() method of Alert
//Class
alert.accept();

//Click Cancel button, by calling dismiss() method of
//Alert Class
alert.dismiss();
```

Obsługa prompt box alertów

```
//Get the Alert  
Alert alert = driver.switchTo().alert();
```

```
//Enter some value on Prompt by calling sendKeys() method of  
//Alert Class  
alert.sendKeys("Foo");
```

```
//Click OK button, by calling accept() method of Alert Class  
alert.accept();
```



Praca z framami

```
<html>
    <frameset cols="25%,*,25%" FRAMEBORDER="NO" FRAMESPACING="0"
BORDER="0">
        <frame id="left" src="frame_a.htm" />
        <frame src="frame_b.htm" />
        <frame name="right" src="frame_c.htm" />
    </frameset>
</html>

//Activate the frame on left side using it's id attribute
driver.switchTo().frame("left");

//Activate the Page, this will move context from frame back to
//the Page
driver.switchTo().defaultContent();

//Activate the frame on right side using it's name attribute
driver.switchTo().frame("right");

//Activate the frame in middle using it's index. Index starts
//at 0
driver.switchTo().frame(1);
```

Zadanie #3

- Napisz test automatyczny do rejestracji na stronie:
<http://demoqa.com/registration/>

Czas: 1.5h





Waits

Implicit wait

- Czekamy zdefiniowany czas zanim wyrzucimy wyjątek, dotyczący **widoczności** elementu.
- Dotyczy całego testu, nie elementu na który czekamy.
- Staramy się go nie używać zbyt wiele i nie łączymy go z innymi waitami!
- Współtwórca Selenium przeprasza 😊

```
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

Explicit wait

- Czeka maksymalnie zdefiniowaną ilość czasu na spełnienie warunku
- Jeśli warunek spełniony jest wcześniej, test jest kontynuowany

```
1 WebDriverWait wait = new WebDriverWait(driver, 10);  
2  
3 WebElement element = wait.until(ExpectedConditions.elementToBeClickable(By.id("someid")));
```

Fluent wait

- We fluent wait definiujemy czas maksymalny by czekać na warunek
- Definiujemy także czas odpytywania
- Definiujemy własne warunki

```
1 // Waiting 30 seconds for an element to be present on the page, checking
2
3 // for its presence once every 5 seconds.
4
5 Wait wait = new FluentWait(driver)
6
7     .withTimeout(30, SECONDS)
8
9     .pollingEvery(5, SECONDS)
10
11     .ignoring(NoSuchElementException.class);
12
13 WebElement foo = wait.until(new Function() {
14
15     public WebElement apply(WebDriver driver) {
16
17         return driver.findElement(By.id("foo"));
18
19     }
20
21 });
```



**KEEP
CALM**

AND

**LETS WRITE
SOME CODE**

KeepCalmAndPosters.com

Zadanie #4

- <https://demos.telerik.com/kendo-ui/dragdrop/index>

Przenieś małą piłkę na dużą i upewnij się, że została przeniesiona.

- <https://jqueryui.com/datepicker/#dropdown-month-year>

Wybierz jedną konkretną datę z kalendarza i sprawdź czy wybrana data pojawiła się w polu Date.

- <https://jqueryui.com/demos/>

Sprawdź czy menu strony zawiera poniższe elementy:

Demos, Downloads, API Documentation, Themes, Development, Support, Blog, About.

Page Object / Page Factory

```
public class NoPOMTest99GuruLogin {
```

```
/**
```

```
 * This test case will login in http://demo.guru99.com/V4/  
 * Verify login page title as guru99 bank  
 * Login to application  
 * Verify the home page using Dashboard message  
 */
```

```
@Test(priority=0)
```

```
public void test_Home_Page_Appear_Correct(){
```

```
    WebDriver driver = new FirefoxDriver();
```

```
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

```
    driver.get("http://demo.guru99.com/V4/");
```

```
    //Find user name and fill user name
```

```
    driver.findElement(By.name("uid")).sendKeys("demo");
```

```
    //find password and fill it
```

```
    driver.findElement(By.name("password")).sendKeys("password");
```

```
    //click login button
```

```
    driver.findElement(By.name("btnLogin")).click();
```

```
    String homeText = driver.findElement(By.xpath("//table//tr[@class='heading3']")).getText();
```

```
    //verify login success
```

```
    Assert.assertTrue(homeText.toLowerCase().contains("guru99 bank"));
```

```
}
```

1 Find user name and fill it

2 Find password and fill it

3 Find Login button and click it

Find home
page text
and get it

4

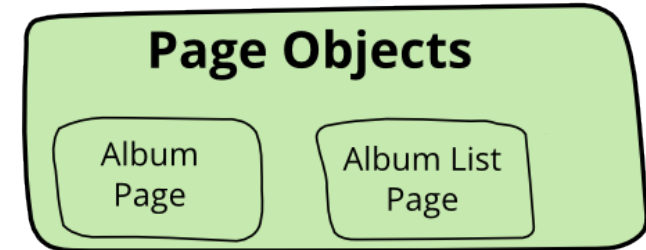
5 Verify home page has text 'Guru99 Bank'

Page Object

- Wzorzec projektowy
- Klasa odzwierciedla pojedynczą stronę / widok aplikacji
- Klasa zawiera lokatory jak i metody do obsługi strony czy widoku
- Jeśli coś się zmieni na stronie, zmieniamy jedną klasę a nie wszystkie testy.

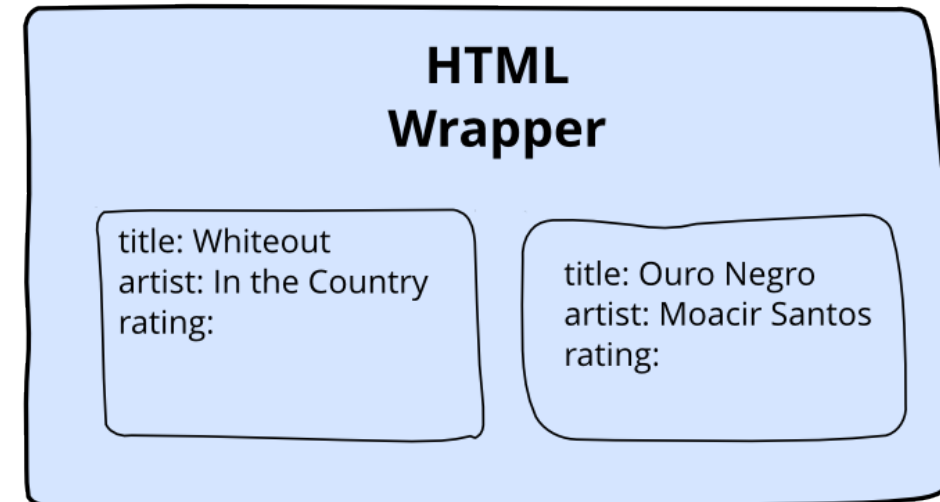
this API is about the application

```
selectAlbumWithTitle()
getArtist()
updateRating(5)
```



this API is about HTML

```
findElementsWithClass('album')
findElementsWithClass('title-field')
getText()
click()
findElementsWithClass('ratings-field')
setText(5)
```



Page Object - przykład

```
public class Guru99Login {  
    WebDriver driver;  
    By user99GuruName = By.name("uid");  
    By password99Guru = By.name("password");  
    By titleText = By.className("barone");  
    By login = By.name("btnLogin");  
  
    public Guru99Login(WebDriver driver){  
        this.driver = driver;  
    }  
    //Set user name in textbox  
    public void setUsername(String strUserName){  
        driver.findElement(user99GuruName).sendKeys(strUserName);  
    }  
}
```

1 Page class in object repository

Find Web Element

Performing operation on web element

2

3

Page Factory - przykład

WebElements are identify by
@FindBy Annotation

Static initElements method of
PageFactory class for
initializing WebElement

```
@FindBy(xpath="//table//tr[@class='heading3']")  
WebElement homePageUserName;  
  
public Guru99HomePage(WebDriver driver){  
    this.driver = driver;  
    //This initElements method will create all WebElements  
    PageFactory.initElements(driver, this);  
}
```

Page Object / Factory – dobre praktyki

- Klasa Page Object nie powinna zawierać asercji
- Nie wrzucaj lokatorów i metod, których nie potrzebujesz (YAGNI)
- Nazwy metod i klas powinny być znaczące

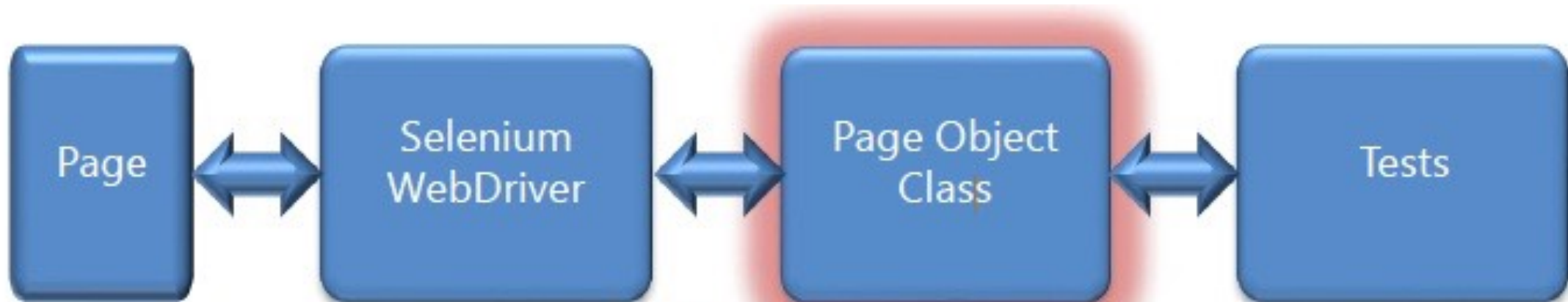


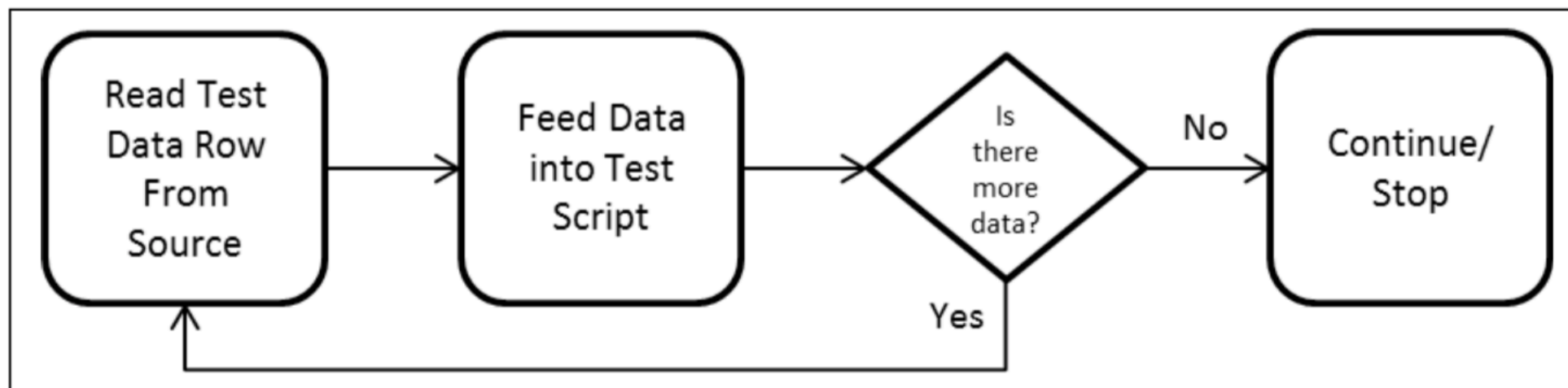
Figure 1

Zadanie #5

- Napisz prosty framework testowy dla strony <https://www.phptravels.net/>
- Email user@phptravels.com
Password demouser
- Scenariusze do pokrycia: Logowanie, sprawdzenie czy menu zawiera wszystkie elementy, czy działa guzik „Details” w hotelach, wyszukiwanie lotów.
- Zastosuj to czego się nauczyłeś – wzorce, dobre praktyki, znaczące nazwy, czysty kod, struktura projektu.

Data Driven Testing

DDT – Testy sterowane danymi



DDT – plusy i minusy



- Usuwamy duplikację – jedna metoda testowa a wiele przypadków
- W szybki sposób budujemy coverage
- Łatwe w utrzymaniu
- W przypadku zmiany funkcjonalności często wymagana jest zmiana w danych
- Dane mogą być pisane przez testerów manualnych
- Wiele narzędzi implementujących to podejście
- Generatory danych
- Wiele formatów danych
- Wymaga wiedzy z programowania
- Przygotowując dane testowe, w narzędziach typu notatnik, łatwo możemy się pomylić
- Nie zawsze warto implementować to podejście

Zadanie #6

- Wybierz jedno z przedstawionych podejść do DDT.
- Zaimplementuj testy obliczania wskaźnika BMI przy użyciu DDT dla aplikacji <http://bmi-online.pl/>
- Testy powinny być napisane dla wszystkich kategorii wagowych i dla obu płci.
- Rezultat testów też powinien być elementem DDT.

Czas: 2h

Pytania rekrutacyjne z Selenium

Pytania rekrutacyjne z Selenium

1. Czy jest automatyzacja testów?
2. Jakie są benefity testów automatycznych?
3. Dlaczego wybrałbyś Selenium jako narzędzie do automatyzacji?
4. Jakie są limitacje Selenium?
5. Czym są lokatory i jakie znasz?
6. Czym jest Xpath?
7. Jaka jest różnica między / a // w Xpath?
8. Jakie znasz typy driverów?
9. Jak można pobrać tekst z elementu przy pomocy Selenium?
10. Jak pobrać wartość z listy rozwijalnej?
11. Jakie znasz komendy nawigacyjne?
12. Kiedy używać findElement() a kiedy findElements()?
13. Jakie znasz anotacje Junit?
14. Czym jest framework?



"I get sea-sick, I can't swim but I rather fancy a little desk job like yours."