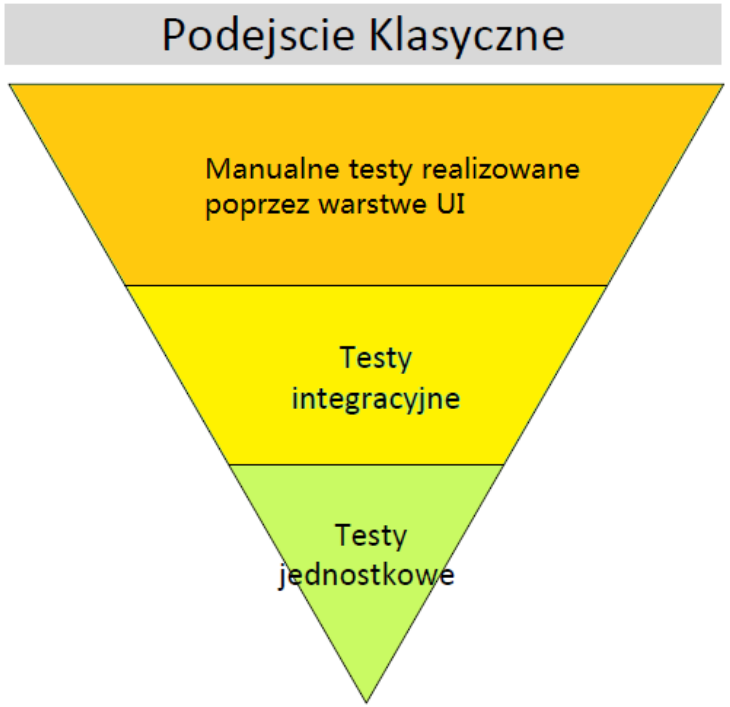
Automatyzacja testów w Javie

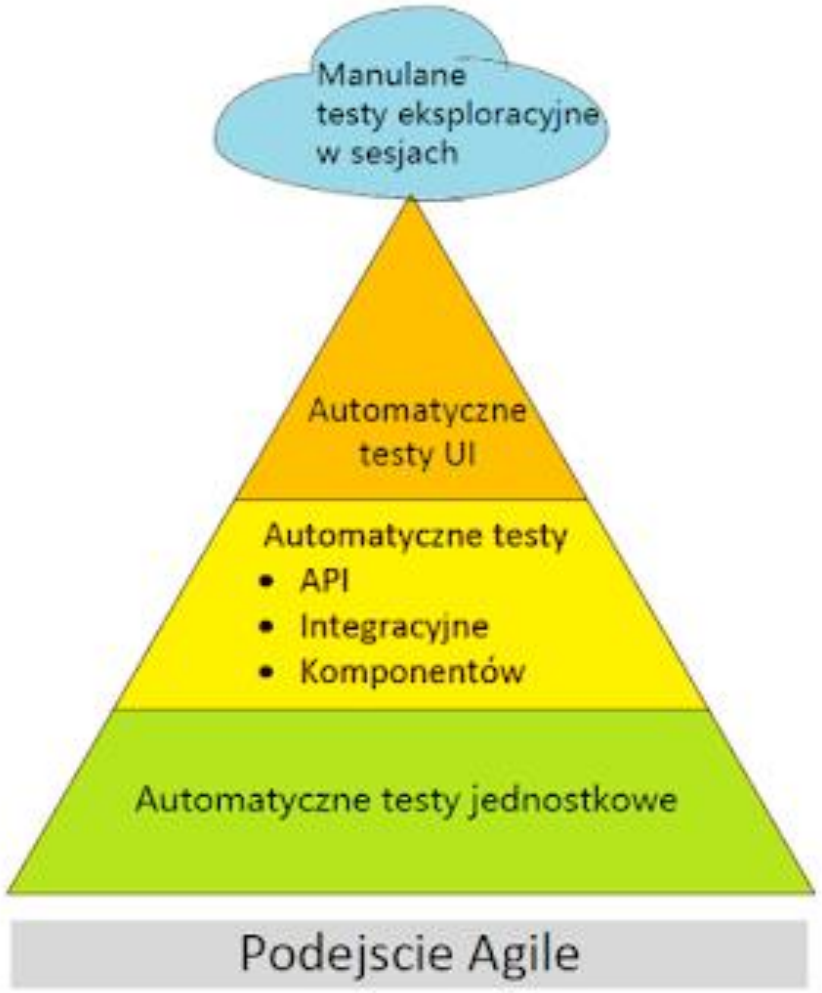
Autor: Paweł Dubaj

Warszawa 2019

Co, ile, gdzie, kiedy … ???

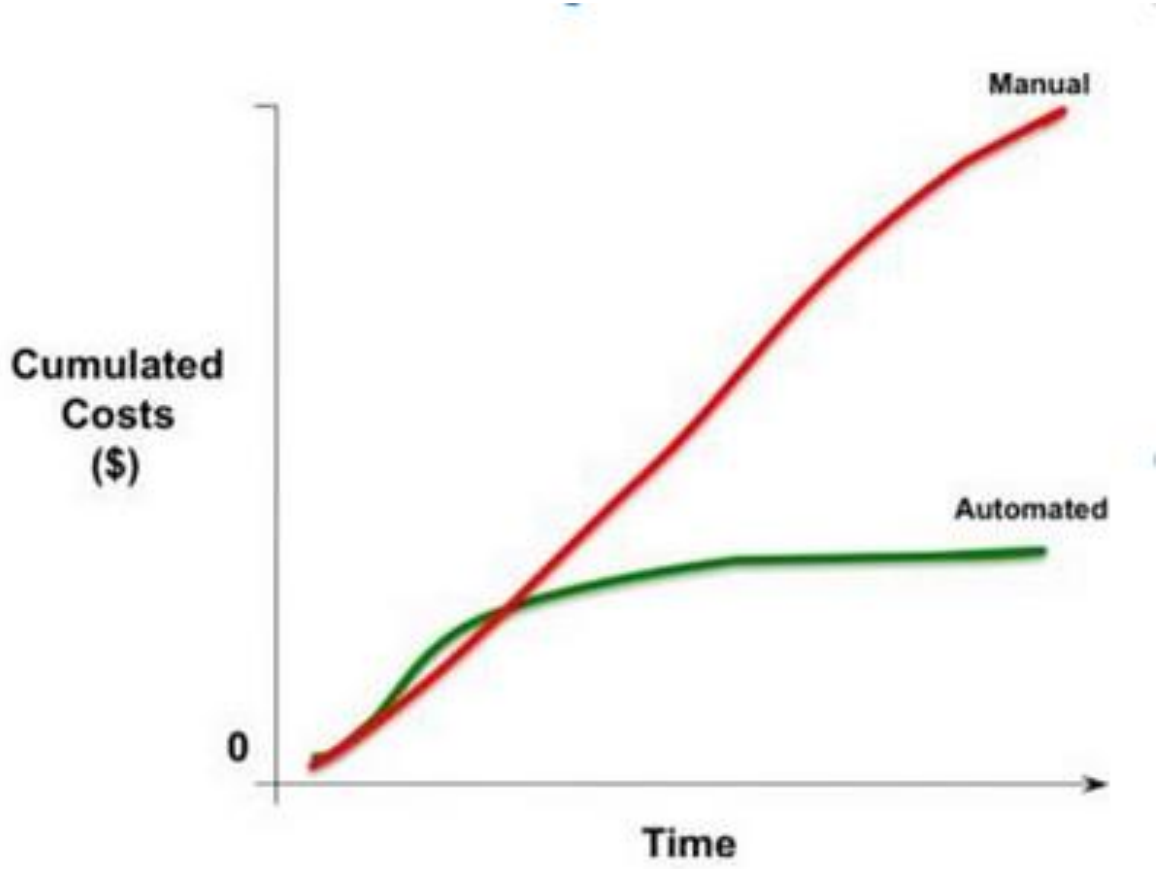
Piramida testów




Manulane
testy eksploracyjne
w sesjach
Automatyczne
testy UI
Automatyczne testy
API
Integracyjne
Komponentów
Automatyczne testy jednostkowe
Podejscie Agile

Koszt wykrycia i naprawy błędu
Zasada 1 – 10 – 100

Koszt automatyzacji

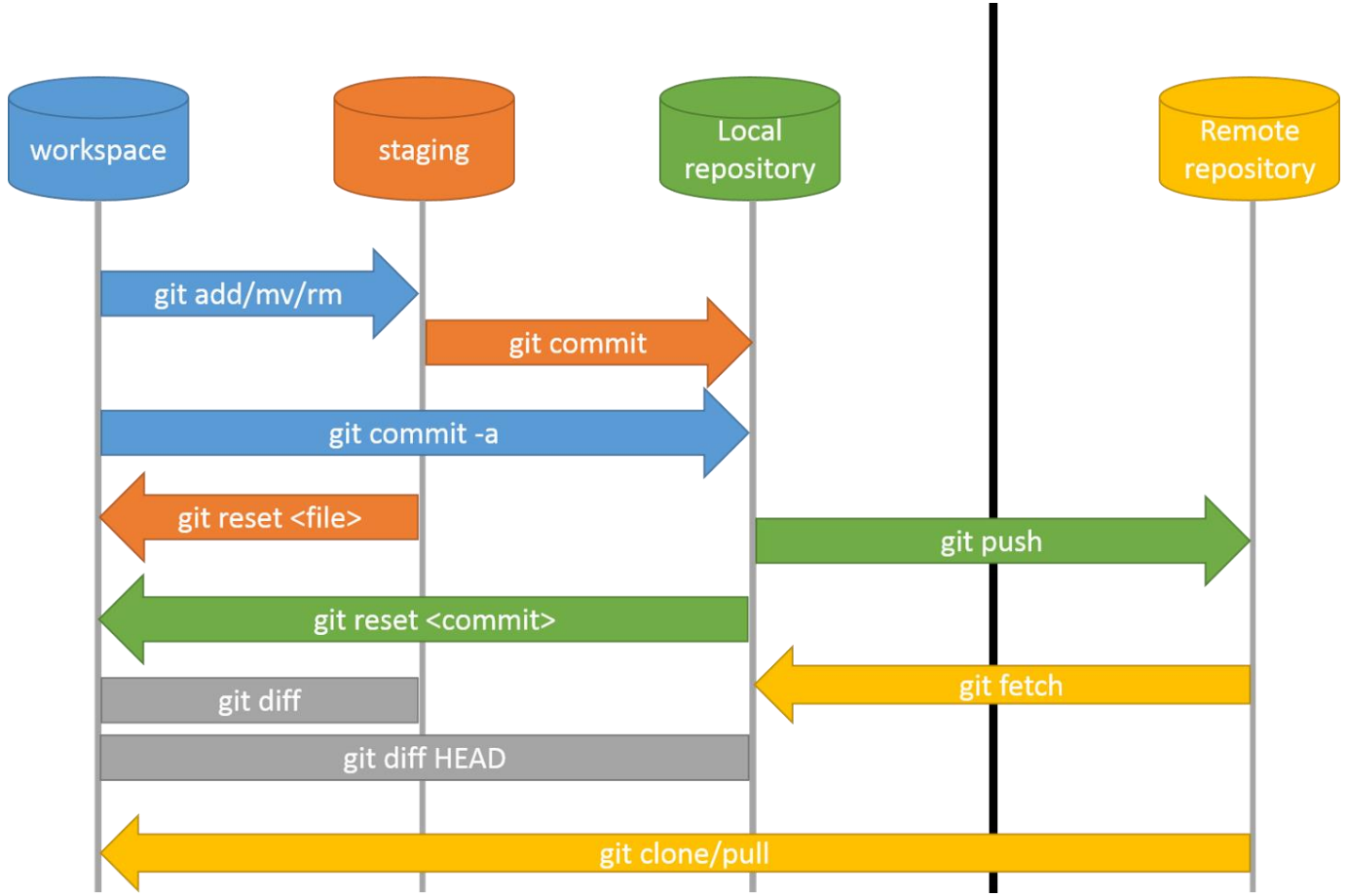


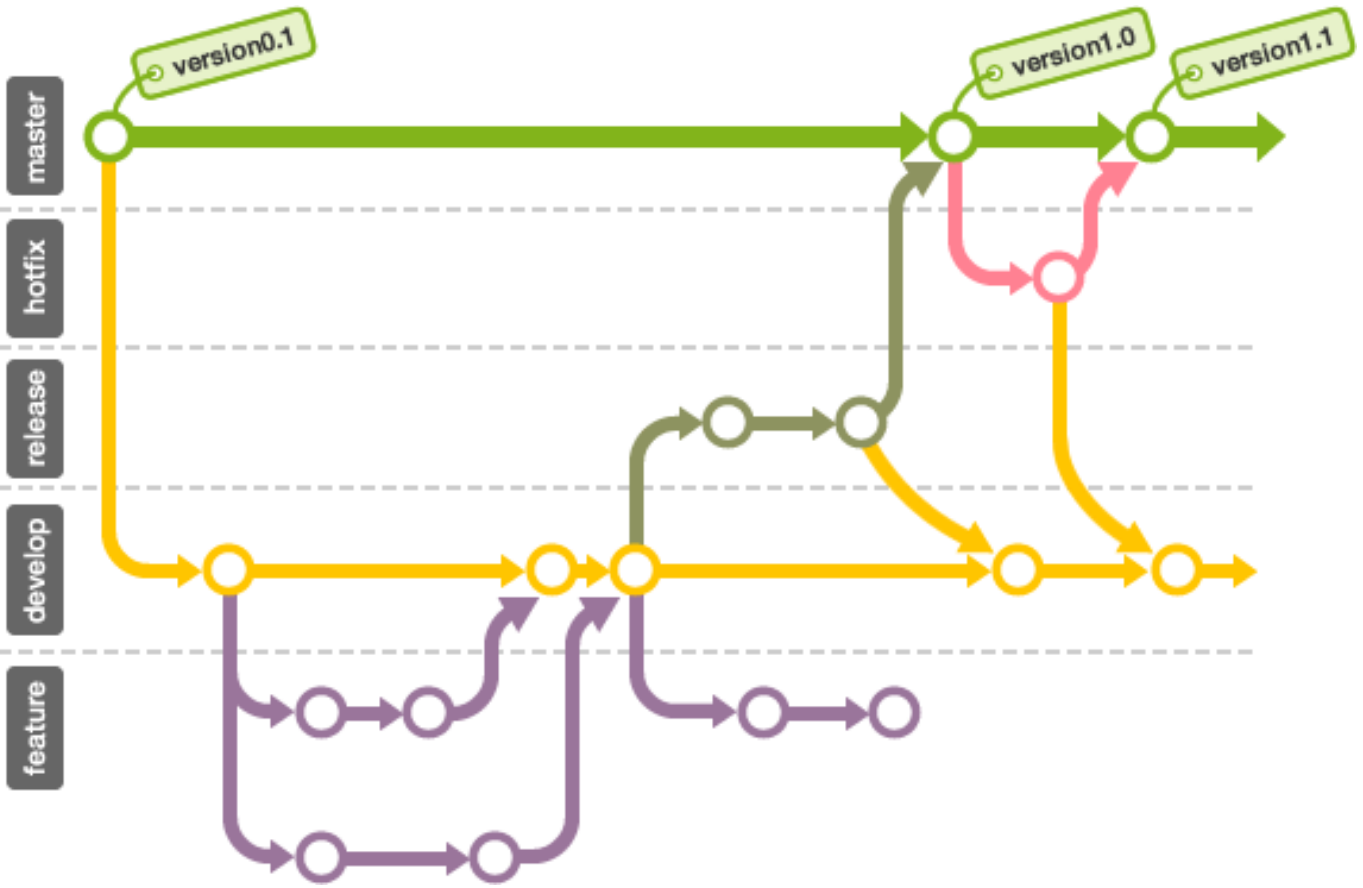
Sprawdźmy czy wszystko działa

- java -version
- git --version
- mvn --version

GIT – wersjonowanie kodu

https://git-scm.com/
https://github.github.com/training-kit/downloads/pl/github-git-cheat-sheet/




version0.1
version1.0
version1.1
master
hotfix
release
develop
feature

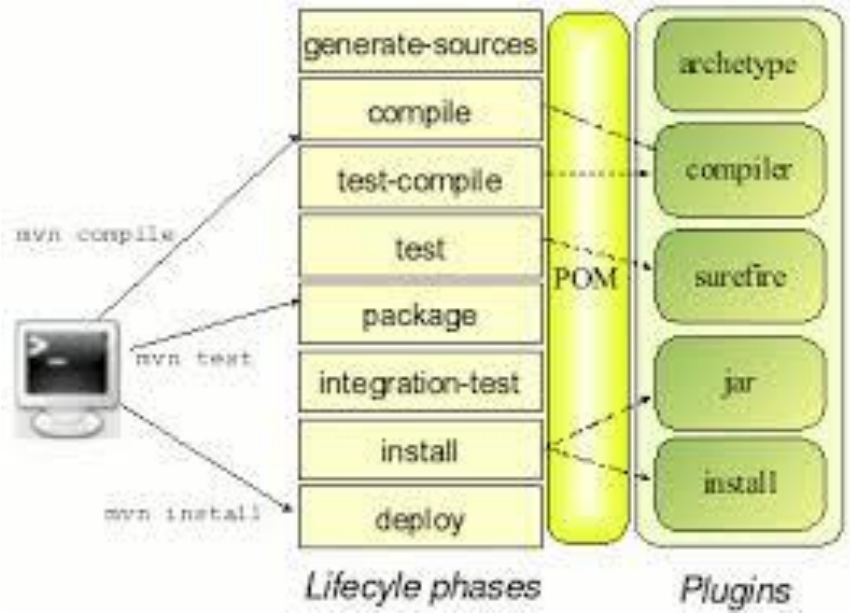
Kilka najważniejszych komend

- git init
- git pull
- git add <file name>
- git commit –m „some message”
- git diff
- git push
- git fetch
- git checkout <branch name/commit hash>
- git merge <branch name>

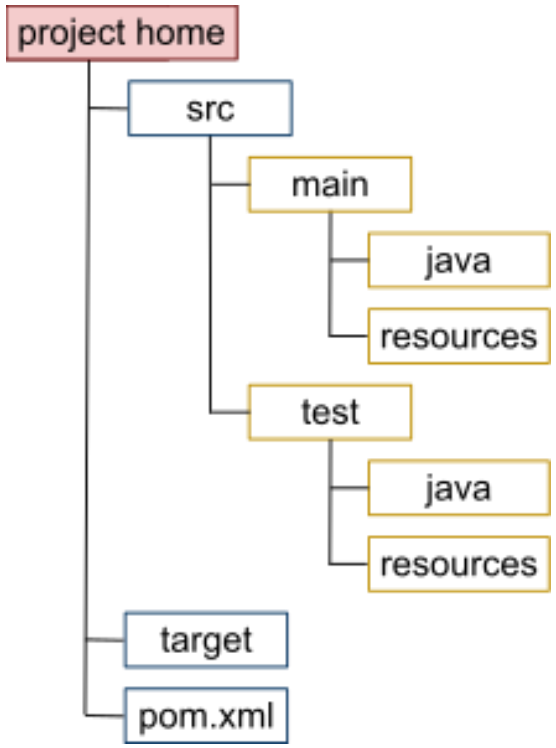
Maven – budowanie projektu

https://maven.apache.org/
https://www.baeldung.com/maven

- mvn clean
- mvn test
- mvn install
- mvn clean install -Dtest=className#testName -Dparam=value



Maven- Struktura projektu



pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <groupId>com.jsystems</groupId>
    <artifactId>qa</artifactId>
    <packaging>pom</packaging>
    <version>1.0-SNAPSHOT</version>

    <modules>
        <module>qaapi</module>
    </modules>

    <properties>
        <jdk.version>1.8</jdk.version>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    </properties>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.junit.jupiter</groupId>
                <artifactId>junit-jupiter-api</artifactId>
                <version>5.4.2</version>
                <scope>test</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.6.1</version>
```

Identyfikacja projektu i modułów

```
<groupId>com.jsystems</groupId>
<artifactId>qa</artifactId>
<packaging>pom</packaging>
<version>1.0-SNAPSHOT</version>

<modules>
    <module>qaapi</module>
    <module>qajunit</module>
    <module>frontend</module>
</modules>

<properties>
    <jdk.version>1.8</jdk.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
</properties>
```

```
<groupId>com.jsystems</groupId>
```
- Identyfikator grupy/projektu

```
<artifactId>qa</artifactId>
```
- Identyfikator modułu

```
<packaging>pom</packaging>
```
- sposób pakowania projektu

```
<version>1.0-SNAPSHOT</version>
```
- wersja modułu

```
<modules>
    <module>qaapi</module>
    <module>qajunit</module>
    <module>frontend</module>
</modules>
```
- zależne podmoduły

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.6.1</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.22.2</version>
            <dependencies>
                <dependency>
                    <groupId>org.junit.platform</groupId>
                    <artifactId>junit-platform-surefire-provider</artifactId>
                    <version>1.3.2</version>
                </dependency>
                <dependency>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
                    <version>5.4.2</version>
                </dependency>
                <dependency>
                    <groupId>org.junit.jupiter</groupId>
                    <artifactId>junit-jupiter-engine</artifactId>
                    <version>5.4.2</version>
                </dependency>
            </dependencies>
            <configuration>
                <includes>
                    <include>**/*Test.java</include>
                </includes>
            </configuration>
        </plugin>
    </plugins>
</build>
```

```xml
<build>
    …
</build>
```

- specyfikacja budowania projektu

```xml
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
```

plugin odpowiadający za specyfikację kompilatora

```xml
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
```

plugin odpowiadający za uruchamianie testów

```xml
<dependency>
    <groupId>org.junit.vintage</groupId>
    <artifactId>junit-vintage-engine</artifactId>
```

zależność odpowiadająca za uruchomienie testów junit4 na platformie junit5

```xml
<configuration>
    <includes>
        <include>**/*Test.java</include>
```

dodatkowa konfiguracja,
odpowiadająca za uruchomienie wszystkich klas
z końcówką *Test.java w nazwie
(testy sterowane przez Cucumber)

Maven Repository

https://mvnrepository.com/

C:\Users\user.m2

Junit 5 – Jupiter

https://junit.org/junit5/docs/current/user-guide/

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.4.2</version>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-params</artifactId>
    <version>5.4.2</version>
    <scope>test</scope>
</dependency>
```

Test, opisy i tagowanie

```java
@DisplayName("Junit tests")
@Tag("unit")
public class JunitTest {

    final String stringTestowy = "stringTestowy";
    final String testowy = null;

    @Test
    @DisplayName("First junit test")
    public void firstTest() {

        assertTrue(true, "message for test result");
        assertTrue(stringTestowy.equals("stringTestowy"), "message for test result");
        assertTrue(5 == 2 + 3, "message for test result");
        assertTrue(true);
        assertFalse(false);
        assertFalse(stringTestowy.matches("^s"));
        assertEquals("stringTestowy", stringTestowy);
        assertSame("stringTestowy", stringTestowy);
    }
}
```

JUnit Assercje

assertTrue(boolean condition)

assertFalse(boolean condition)

assertNull(Object actual)

assertNotNull(Object actual)

assertEquals(short expected, short actual)

assertArrayEquals(boolean[] expected, boolean[] actual)

assertIterableEquals(Iterable<?> expected, Iterable<?> actual)

assertLinesMatch(List<String> expectedLines, List<String> actualLines)

assertNotEquals(byte unexpected, byte actual)

assertSame(Object expected, Object actual)

assertNotSame(Object unexpected, Object actual)

assertAll(Executable... executables)

assertThrows(Class<T> expectedType, Executable executable)

assertDoesNotThrow(Executable executable)

assertTimeout(Duration timeout, Executable executable)

assertTimeoutPreemptively(Duration timeout, Executable executable)

Przed testami i po testach

```java
@BeforeAll
public static void setUpBeforeAll() {
    System.out.println("========== BeforeAll =========");
}

@BeforeEach
public void setUp(TestInfo testInfo) {
    System.out.println("========== BeforeEach =========");
    System.out.println("DisplayName: " + testInfo.getDisplayName());
    System.out.println("========== Test Name class name: " + testInfo.getTestClass().getClass().getSimpleName()
            + " \ntest name: " + testInfo.getTestMethod() );
}

@AfterEach
public void tearDown(TestInfo testInfo) {
    System.out.println("========== AfterEach =========");
    System.out.println("DisplayName: " + testInfo.getDisplayName());
    System.out.println("========== Test Name class name: " + testInfo.getTestClass()
            + " \ntest name: " + testInfo.getTestMethod() );
}

@AfterAll
public static void tearDownAfterAll() {
    System.out.println("========== AfterAll =========");
}
```

Matchery

hamcrest

```
<dependency>
    <groupId>org.hamcrest</groupId>
    <artifactId>hamcrest-all</artifactId>
    <version>1.3</version>
    <scope>test</scope>
</dependency>
```

```
assertThat("message from That", testowy, is("firstTest"));
assertThat("message from That", testowy, containsString("Test"));
assertThat("message from True", testowy, equalTo("firstTest"));
assertThat("message from True", testowy, endsWith("t"));
```

google.truth

```
<dependency>
    <groupId>com.google.truth</groupId>
    <artifactId>truth</artifactId>
    <version>0.45</version>
    <scope>test</scope>
</dependency>
```

```
assertThat(stringTestowy).isEqualTo("stringTestowy");
assertThat(stringTestowy).contains("s");
assertThat(stringTestowy).matches("^s");
```

Zadanie 1.

```
String resultString = "Wordpress powers 34% of the internet";
String expectedString = "Wordpress powers [number]% of the internet";
```

Proszę napisać test i sprawdzić czy:

1. Zwracany resultString jest taki jak expectedString, z wyjątkiem zmieniającego się numeru.
2. Czy numer jest zwracany poprawnie (liczba całkowita) i czy jest większy od < 0 >

Parametryzacja testów

```java
@DisplayName("First parameterized test")
@ParameterizedTest(name = "Parameterized test with value {0}")
@ValueSource(ints = {5, 15, 25})
public void paramFirstTest(int number) {
    assertTrue(number % 5 == 0);
}

@DisplayName("Second parameterized test")
@ParameterizedTest(name = "Parameterized test with value {0}")
@ValueSource(strings = {"Hello", "Hello Junit", "Hello students"})
public void paramSecondTest(String value) {
    assertTrue(value.contains("Hello"));
}

@DisplayName("Csv value parameterized test")
@ParameterizedTest(name = "Parameterized test with values name: {0} and value: {1}")
@CsvSource(value = {"Hello, 5", "HelloJunit 5, 15", "'Hello 5!', 25"}, delimiter = ',')
public void paramMultiArgTest(String param1, int param2) {
    assertTrue(param1.contains("Hello"));
    assertTrue(param2 % 5 == 0);
}

@DisplayName("Csv file source parameterized test")
@ParameterizedTest(name = "Parameterized test with data from csv file, name: {0} and value: {1}")
@CsvFileSource(resources = "/plik.csv", delimiter = ',')
public void csvFileSourceTest(String param1, int param2) {
    assertTrue(param1.contains("Hello"));
    assertTrue(param2 % 5 == 0);
}

@DisplayName("Csv file source parameterized test")
@ParameterizedTest(name = "Parameterized test with data from csv file, name: {0} and value: {1}")
@EnumSource(value = SimpleEnum.class)
public void csvFileSourceTest(SimpleEnum simpleEnum) {
    assertTrue(simpleEnum.toString().contains("A"));
}

enum SimpleEnum {
    A, AA, AAA
}
```

Zadanie 2

```
String resultString = "Wordpress powers 34% of the internet";
```

Proszę napisać test sparametryzowany

1. Proszę sprawdzić czy resultString zawiera wyrazy: „Wordpress” , „powers”, „internet”

Configuration

typesafe.config

```
<dependency>
    <groupId>com.typesafe</groupId>
    <artifactId>config</artifactId>
    <version>1.3.4</version>
</dependency>
```

Plik config.conf umieszczamy w resourcach: src/main/resources/config.conf

```
environment = "dev"
environment = ${?ENVIRONMENT}

environments {
    dev {
            baseUrl = "https://wordpress.com/"
            login = "test_login"
            password = "test_pass"

        }
}
```

```
private static final Config CONFIG = ConfigFactory.load("config.conf");
private static final String ENVIRONMENT = CONFIG.getString("environment");

private static final Config ENV = CONFIG.getConfig("environments").getConfig(ENVIRONMENT);

public static final String BASE_URL = ENV.getString("baseUrl");
public static final String LOGIN = ENV.getString("login");
public static final String PASSWORD = ENV.getString("password");
```

Selenium WebDriver

https://www.seleniumhq.org/projects/webdriver/

Drivery

I. Ustawiamy drivery przeglądarek poprzez odwołanie do zmiennych środowiskowych

```
static String chromePath;
static String fireFoxPath;

private void setDrivers() {
    try {
        chromePath = Paths.get(getClass().getClassLoader().getResource("driver/chromedriver.exe")
                .toURI()).toFile().getAbsolutePath();
        fireFoxPath = Paths.get(getClass().getClassLoader().getResource("driver/geckodriver.exe")
                .toURI()).toFile().getAbsolutePath();
    } catch (URISyntaxException e) {
        e.printStackTrace();
    }
}

@BeforeEach
public void setUpEach() {
    setDrivers();
    System.setProperty("webdriver.chrome.driver", chromePath);
    System.setProperty("webdriver.gecko.driver", fireFoxPath);
```

II. Używamy WebDriverMenagera

```
<dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
    <version>3.7.1</version>
</dependency>
```

```
@BeforeAll
public static void setUpAll() {
    WebDriverManager.chromedriver().setup();
    WebDriverManager.firefoxdriver().setup();
}
```

WebDriver

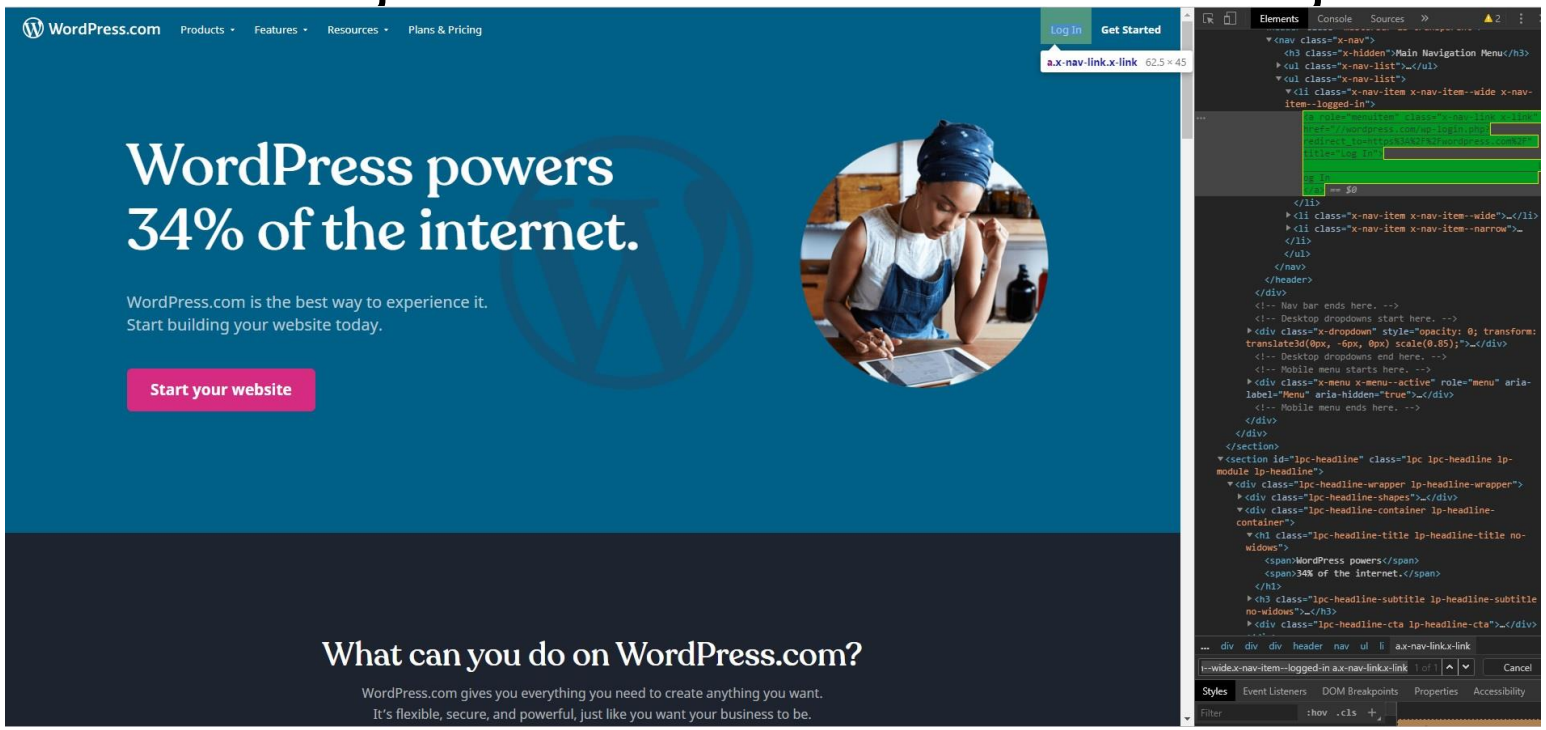
```
WebDriver driver;


@BeforeEach
public void setUpEach() {
String browser = Configuration.getBROWSER();

    if(browser.equals("chrome")){
        driver = new ChromeDriver();
    } else if(browser.equals("firefox")){
        driver = new FirefoxDriver();
    }

    driver.manage().window().maximize();
    driver.manage().deleteAllCookies();
    driver.manage().timeouts().pageLoadTimeout(120, TimeUnit.SECONDS);

}

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

UI Test

```
@Test
public void firstFrontTest() {
    driver.get("https://wordpress.com/");
    WebElement login = driver.findElement(By.cssSelector(".x-nav-item.x-nav-item--wide.x-nav-item--logged-in a.x-nav-link.x-link"));
    WebElement buildEWebsite = driver.findElement(By.cssSelector("#lpc-headline .lpc-headline-container.lp-headline-container h1 span:nth-child(1)"));
    assertTrue(buildEWebsite.isDisplayed());
    assertEquals(buildEWebsite.getText(), "Build a website,");
    assertTrue(login.isDisplayed());
    assertEquals(login.getText(), "Log In");
}
```

Page Object Pattern

Każde okno jest odzwierciedlone w osobnej klasie




```
public class MainWordporessPage extends BasePage {

    public MainWordporessPage(WebDriver driver) {
        super(driver);
    }

    public WebElement login = driver.findElement(By.cssSelector(".x-nav-item.x-nav-item--wide.x-nav-item--logged-in" +
            " a.x-nav-link.x-link"));

    public WebElement buildEWebsite = driver.findElement(By.cssSelector("#lpc-headline " +
            ".lpc-headline-container.lp-headline-container h1 span:nth-child(1)"));
}
```

Page Factory

```
public class MainWordporessPage extends BasePage {

    public MainWordporessPage(WebDriver driver) {
        super(driver);
        PageFactory.initElements(driver, this);
    }

    @FindBy(css = ".x-nav-item.x-nav-item--wide.x-nav-item--logged-in a.x-nav-link.x-link")
    public WebElement login;

    @FindBy(css = "#lpc-headline .lpc-headline-container.lp-headline-container h1 span:nth-child(1)")
    public WebElement buildEWebsite;

}
```

```
MainWordporessPage wordporessPage;

@Test
public void firstFrontTest() {
    driver.get("https://wordpress.com/");
    wordporessPage = new MainWordporessPage(driver);
    assertTrue(wordporessPage.buildEWebsite.isDisplayed());
    assertEquals(wordporessPage.buildEWebsite.getText(), "Build a website,");
    assertTrue(wordporessPage.login.isDisplayed());
    assertEquals(wordporessPage.login.getText(), "Log In");
    wordporessPage.login.click();
}
```

Lokatory

Zalecane jest odwoływać się na stronie po ‚Id' lub ‚Name' ponieważ są one najszybciej znajdowane na stronie.

Zalecane jest używać lokatorów w następującej hierarchii

- Id,
- Name,
- ClassName,
- XPatha,
- CssSelectora
- Nadać elementowi id'ka lub name,
- LinkText
- PartialLinkText
- TagName

WebElement

WebElement element = driver.findElement(By.Id(„elementId”))

element.isDisplayed();

element.isEnabled();

element.isSelected();

element.clear();

element.click();

element.getText();

element.sendKeys();

element.sendKeys();

element.submit();

element.getTagName();

element.getAttribute();

element.getLocation();

element.getSize();

Zadanie 3

Proszę napisać test w którym użytkownik będzie się logował na swoje konto i po zalogowaniu wejdzie na profil użytkownika i sprawdzi czy nazwa użytkownika się zgadza.

WebDriverWait

```
WebDriverWait wait = new WebDriverWait(driver, 30);
wait.until(ExpectedConditions.visibilityOf(element));
```

Alert

```
Alert alert = driver.switchTo().alert();
alert.accept();
driver.switchTo().alert()
```

Action

```
Actions action = new Actions(driver);
action
        .moveToElement(loginPage.emailInput)
        .sendKeys(Configuration.LOGIN)
        .sendKeys(Keys.chord(Keys.ENTER))
        .build()
        .perform();
```

JavaScriptExecutor

```
JavascriptExecutor jsexecutor = (JavascriptExecutor) driver;
jsexecutor.executeScript("arguments[0].scrollIntoView(true);", windowFrame);
```

Window

```
String firstPageWindowHandle;
String secondTestWindowHandle = null;

firstPageWindowHandle = driver.getWindowHandle();

Set<String> testPageWindowHandle = driver.getWindowHandles();

for (String windowHandle : testPageWindowHandle) {
    if (!firstPageWindowHandle.equals(windowHandle)) {
        secondTestWindowHandle = windowHandle;
    }
}

driver.switchTo().window(secondTestWindowHandle);
driver.switchTo().window(secondTestWindowHandle).close();
driver.switchTo().window(firstPageWindowHandle);
```

Frame

```
driver.switchTo().frame(testframe);
driver.switchTo().parentFrame();
```

Location

```
int hyperlinkYCoordinate = windowFrame.getLocation().getY();
int hyperlinkXCoordinate = windowFrame.getLocation().getX();
```

BDD - Cucumber

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>4.5.2</version>
</dependency>

<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>4.5.2</version>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-picocontainer</artifactId>
    <version>4.5.2</version>
    <scope>test</scope>
</dependency>
```

Gherkin i scenariusze BDD

Scenariusze testowe piszemy w plikach z rozszerzeniem .feature

np. testscenario.feature

Pliki .feature lokalizujemy w folderze ‚resources' w części ‚test'

src/test/resources/testscenario.feature

Do pisania scenariuszy używamy składni języka Gherkin

```
Feature: User panel setup

  @wordpress @login @userProfile
  Scenario: Setup user profile
    Given User start on main page
    When User log In to the user page
    Then User can modified user profile
```

RunTest

Konfiguracja

```
@RunWith(Cucumber.class)
@CucumberOptions(
        features = "src/test/resources",
        glue = "classpath:com.jsystems.qa.frontend.cucumber",
        plugin = {"html:target/cucumber-html-report", "rerun:target/rerun.txt"},
        tags = {
                "@wordpress",
//                "@login",
//                "@userProfile",
//                "@notification"
        }
)
public class RunTest {
}
```

```
features = "src/test/resources",
```
- lokalizacja features

```
glue = "classpath:com.jsystems.qa.frontend.cucumber",
```
- lokalizacja stepów

```
plugin = {"html:target/cucumber-html-report", "rerun:target/rerun.txt"},
```
- pluginy np. raporty

```
tags = {
                "@wordpress",
//                "@login",
        }
```
- tagi wywołujące otagowane scenariusze

Cucumber Before & After

```
WebDriver driver;

@Before
public void setUpAll() {
    WebDriverManager.chromedriver().setup();
    WebDriverManager.firefoxdriver().setup();
}

public WebDriver setUp() {

    String browser = Configuration.getBROWSER();
    if(browser.equals("chrome")){
        driver = new ChromeDriver();
    } else if(browser.equals("firefox")){
        driver = new FirefoxDriver();
    }

    driver.manage().window().setSize(new Dimension(1920,1080));
    driver.manage().deleteAllCookies();
    driver.manage().timeouts().pageLoadTimeout(120, TimeUnit.SECONDS);
    return driver;
}

@After
public void tearDown(Scenario scenario) {
    String status;
    if(!scenario.isFailed()) {
        status = "( ͡° ͜ʖ ͡°)";
        scenario.write("Scenario passed");
    } else {
        status = "(✖∩✖)";
        scenario.embed(((TakesScreenshot)driver).getScreenshotAs(OutputType.BYTES),"images/png");
        scenario.write("Scenario failed");
    }
    System.out.println("\n"+status+" End of: " + scenario.getName() + " scenario.");
    driver.quit();
    driver = null;
}
```

Steps

```
public LoginSteps(CucumberStepConfig stepConfig) {
    driver = stepConfig.setUp();
}

@Given("^User start on main page$")
public void userStartOnMainPage() {
    driver.get(Configuration.BASE_URL);
}

@When("^User log In to the user page$")
public void userLogInToTheUserPage() {
    login();
    userPage = new UserPage(driver);
    userPage.waitForVisibilityOfElement(userPage.userAvatar, 30);
    assertTrue(userPage.userAvatar.isDisplayed());
}

@Then("^User can modified user profile$")
public void userCanModifiedUserProfiles() {
    userPage.userAvatar.click();
    userProfilePage = new UserProfilePage(driver);
    userProfilePage.waitForVisibilityOfElement(userProfilePage.buttonSave, 120);
    JavascriptExecutor js = (JavascriptExecutor) driver;
    js.executeScript("arguments[0].scrollIntoView(true);", userProfilePage.buttonSave);
    userProfilePage.waitForVisibilityOfElement(userProfilePage.buttonSave, 120);
    assertFalse(userProfilePage.buttonSave.isEnabled());
}
```

```
Feature: User panel setup

  @wordpress @login @userProfile
  Scenario: Setup user profile
    Given User start on main page
    When User log In to the user page
    Then User can modified user profile
```

Proszę napisać test w którym użytkownik będzie się logował na swoje konto , po zalogowaniu wejdzie na profil użytkownik, przejdzie na „ustawienia powiadomień” sprawdzi czy notyfikacja pierwsza z góry jest włączona, wyłączy ją i sprawdzi czy jest wyłączona, na koniec włączy ją ponownie.

HTTP – protocol

https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

Request methods

GET

HEAD

POST

PUT

DELETE

TRACE

OPTIONS

CONNECT

PATCH

Status codes

2xx Success

200 OK.

201 Created

3xx Redirection

300 Multiple Choices

4xx Client errors

400 Bad Request

401 Unauthorized

404 Not Found

5xx Server errors

500 Internal Server Error

API

http://rest-assured.io/

```
<dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <version>4.0.0</version>
</dependency>
```

Klasycznie

```
@Test
public void firstApiTest() {
    RestAssured
            .given()
            .get("http://www.mocky.io/v2/5a6b69ec3100009d211b8aeb")
            .then()
            .assertThat()
            .statusCode(200)
            .body("name", equalTo("Piotr"))
            .body("surname", equalTo("Kowalski"));
}
```

Poprzez rzutowanie na model

```
    @Test
    public void azureUser(){

        UserAzure userAzure = RestAssured.given()
                .baseUri("http://fakerestapi.azurewebsites.net")
                .contentType(ContentType.JSON)
                .when()
                .get("/api/Users/{id}", 1)
                .andReturn()
                .then()
                .extract()
                .body()
                .as(UserAzure.class);

        assertThat(userAzure.id).isEqualTo(1);
        assertThat(userAzure.userName).isEqualTo("User 1");
        assertThat(userAzure.password).isEqualTo("Password1");
    }
```

Specyfikatory

RequestSpecBuilder

```
private static final String V2 = "v2";

public static RequestSpecification requestSpecBuilder() {
    return new RequestSpecBuilder()
            .setContentType(ContentType.JSON)
            .setBaseUri(ApiConfig.BASE_URL)
            .setBasePath(V2)
            .build();
}

public static RequestSpecification requestSpecBuilderWithAuthorisation(String auth) {
    return new RequestSpecBuilder()
            .addHeader("Authorise", "ApiKey, " + auth)
            .setContentType(ContentType.JSON)
            .setBaseUri(ApiConfig.BASE_URL)
            .setBasePath(V2)
            .build();
}
public static RequestSpecification fakeAzureSpecBuilder() {
    return new RequestSpecBuilder()
            .setContentType(ContentType.JSON)
            .addCookie(Cookie.HTTP_ONLY)
            .addHeader("name", "value")
            .addHeader("Authorise", "ApiKey, " + "encodeAuthorization")
            .setSessionId("sessionId")
            .setBaseUri("http://fakerestapi.azurewebsites.net")
            .build();
}
public static RequestSpecification sampleSpecBuilder() {
    return new RequestSpecBuilder()
            .setContentType(ContentType.JSON)
            .addCookie(Cookie.HTTP_ONLY)
            .addHeader("name", "value")
            .addHeader("Authorise", "ApiKey, " + "encodeAuthorization")
            .setSessionId("sessionId")
            .setBaseUri("http://fakerestapi.azurewebsites.net")
            .build();
}
```

Json inputStream

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.9.9</version>
</dependency>

<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-annotations</artifactId>
    <version>2.9.9</version>
</dependency>

<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.9.9</version>
</dependency>
```

Model obiektu

```java
public class DeviceModel {

    @JsonProperty( required = true)
    public String produce;

    @JsonProperty(value = "screen.size", required = true)
    public int screenSize;

}
```

```java
public class MyUser {

    @JsonProperty(required = true)
    public String name;
    @JsonProperty(required = true)
    public String surname;

    public MyUser(String name, String surname) {
        this.name = name;
        this.surname = surname;
    }

    public MyUser() {
    }

}
```

```java
public class UserAzure {

    @JsonProperty(value = "ID")
    public int id;

    @JsonProperty(value = "UserName")
    public String userName;

    @JsonProperty(value = "Password")
    public String password;

}
```

```java
public class ErrorResponse {

    @JsonProperty("Error")
    public ErrorBody error;

    public static class ErrorBody {
        @JsonProperty("error.code")
        public int errorCode;

        @JsonProperty("validation_erro")
        public String validationError;

        public String message;
    }

}
```

Mappowanie response body na model obiektu

```
MyUser myUser = RestAssured
                .given()
                .spec(Specification.requestSpecBuilder())
                .get(USER)
                .then()
                .extract()
                .body()
                .jsonPath()
                .getObject("", MyUser.class);
```

```
UserAzure userAzure = RestAssured.given()
                .spec(Specification.fakeAzureSpecBuilder())
                .baseUri("http://fakerestapi.azurewebsites.net")
                .contentType(ContentType.JSON)
                .when()
                .get("/api/Users/{id}", id)
                .andReturn()
                .then()
                .extract()
                .body()
                .as(UserAzure.class);
```

Mapowanie response body na listę obiektów

```
List<User> users = RestAssured
                .given()
                .spec(Specification.requestSpecBuilder())
                .get(USERS_LIST)
                .then()
                .extract()
                .body()
                .jsonPath()
                .getList("", User.class);
```

```
List<Book> books = Arrays.asList(
        given()
        .spec(requestSpecBuilderFaker)
        .when()
        .get("/api/Books")
        .andReturn()
        .then()
        .extract()
        .body()
        .as(Book[].class));
```

Parametryzowanie requestów

QueryParam

```
MyUser myUser = RestAssured
                .given()
                .spec(Specification.requestSpecBuilder())
                .queryParam("name", name)
                .queryParam("name", surname)
                .get(USER)
                .then()
                .extract()
                .body()
                .jsonPath()
                .getObject("", MyUser.class);
```

PathVariable

```
MyUser myUser = RestAssured
                .given()
                .spec(Specification.requestSpecBuilder())
                .get("/5a6b69ec3100009d211b8aeb/{id}/urlsa/{deviceId}", 1, 5)
                .then()
                .extract()
                .body()
                .jsonPath()
                .getObject("", MyUser.class);
```

Service

Service

```
public class ApiService {
    private static final String USER = "/5a6b69ec3100009d211b8aeb";

public static MyUser getUser() {
        return RestAssured
                .given()
                .spec(Specification.requestSpecBuilder())
                .get(USER)
                .then()
                .assertThat()
                .statusCode(200)
                .extract()
                .body()
                .jsonPath()
                .getObject("", MyUser.class);
    }
}
```

Test

```
@DisplayName("Api tests")
public class ApiTest {

    @Test
    @DisplayName("First test with mapping to user obiect")
    public void jsonPathTest() {
        MyUser user = ApiService.getUser();

        assertThat(user.imie).isEqualTo("Piotr");
        assertThat(user.nazwisko).isEqualTo("Kowalski");
    }
}
```

Testowanie headera

```
@Test
@DisplayName("GET /api/Books  - Tests of Books")
public void getBookById() {
    Response response = given()
            .spec(requestSpecBuilderFaker)
            .when()
            .get("/api/Books/{id}", 1)
            .andReturn();

    assertThat(response.contentType()).isEqualTo(ContentType.JSON);
    assertThat(response.getSessionId()).isEqualTo("sessionId");
    assertThat(response.getCookie("cookie")).isEqualTo("firstCookie");
    assertThat(response.getHeader("name")).isEqualTo("value");

    Book books = response
            .then()
            .extract()
            .body()
            .as(Book.class);

    assertTrue(books.id == 1);
}
```

Zadanie 4

Proszę napisać test na sprawdzenie responsa dla zapytania API ze strony

http://fakerestapi.azurewebsites.net/Help

Dokumentacja Swagger wystawiona jest na:

http://fakerestapi.azurewebsites.net/swagger/ui/index

Proszę napisać test dla API Users dla metody GET api/users

Podłączenie do bazy danych

```
<dependency>
    <groupId>oracle</groupId>
    <artifactId>ojdbc6</artifactId>
    <version>11.2.0.3</version>
    <scope>system</scope>
    <systemPath>${basedir}/lib/ojdbc6-11.2.0.3.jar</systemPath>
</dependency>
```

Jeżeli dependency jest niedostępne do ściągnięcia a mamy pobrany .jar

to możemy go umieścić bezpośrednio w projekcie

qaproject/

/qamodule/

/lib/ojdbc6-11.2.0.3.jar

Database connector

```
public class DatabaseConnector {

    private static Connection connection = null;

    public final synchronized static Connection getConnection() {
        if(connection == null) initConnection();
        return connection;
    }

    private static void initConnection() {
        try {
            Class.forName((ApiConfig.DB_CLASS));
            String url = ApiConfig.DB_URL;
            String user = ApiConfig.DB_USER;
            String pass = ApiConfig.DB_PASSWORD;
            connection = DriverManager.getConnection(url, user, pass);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Dostęp do danych

```
public static UserDb getOneById(Long id) {
    String sql = "select * from testuser where id = " + id;

    UserDb userDb = new UserDb();
    try {
        Statement statement = DatabaseConnector.getConnection().createStatement();
        ResultSet wynik = statement.executeQuery(sql);

        while (wynik.next()) {
            userDb.setId(wynik.getLong(1));
            userDb.setName(wynik.getString(2));
            userDb.setSurname(wynik.getString(3));
        }
        wynik.close();
        statement.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return userDb;
}
```

```
@Test
@Disabled
public void dbTest() {
    UserDb userDb = UserDao.getOneById(1L);
    assertThat(userDb.getName()).isEqualTo("Piotr");
}
```

Zakończenie

Dziękuję wszystkim za uwagę.

pdubaj@interia.pl