웹 애플리케이션 테스트 자동화

1. 소개

웹 애플리케이션 테스트는 웹에서 호스팅 되는 응용 프로그램을 테스트할 때 적합한 테스트 기법이다. 성능, 기능, 사용성, 인터페이스, 호환성 및 보안 테스트 등의 이슈로 나눌 수 있다. 이번 실습에서는 주로 기능 테스트에 초점을 맞춘다. 주요 목표는 시스템의 여러 기능을 커버하는 단위 테스트를 작성하는 방법을 이해하는 것이다. 이번 실습은 Window 와 Mac 환경에서 IntelliJ 기준으로 작성되었으므로, IntelliJ 사용을 추천한다.

2. 테스트 대상(SUT)

2.1. 테스트 대상은 간단한 온라인 스토어이다.

2.2. 제공되는 테스트 대상은 실습용으로 제작되어 기능이 제한적이다.

2.2.1. 사용자는 상품 구입이 가능하다.

2.2.2. 관리자는 상품을 추가 및 수정이 가능하다.

2.2.3. 관리자로 시스템에 액세스 하려면 직접 계정을 제작해야 한다.

3. 컴퓨터에서 웹 응용프로그램 설정 방법

3.1. Eclass에서 “Application Source Code.zip”을 다운로드 하고 압축 해제

3.2. 도커(Docker)를 설치 https://www.docker.com/products/docker-desktop/

※ Docker가 작동하려면 WSL(Windows Subsystem for Linux)이 필요하다.
컴퓨터에 설치되어 있지 않은 경우 관리자 권한으로 "wsl --install" 명령을 실행하여 설치

3.3. 명령 프롬프트(CMD)를 시작하고(관리자 모드) 압축 해제 한 애플리케이션 소스 코드 폴더 ("cd C:₩Users₩...₩OnlineStoreApplication")로 이동

3.4. "docker compose up" 명령어 입력 후 Enter 키 입력

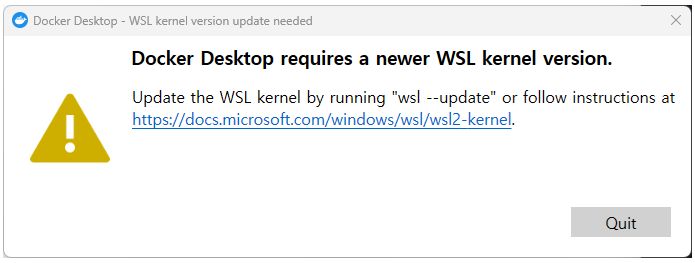
※ 이 명령어는 포트 5432, POSTGRES 데이터베이스 서비스를 실행한다.
따라서 POSTGRES 데이터베이스가 이미 설치되어 있고 동일한 포트를 사용하는 경우,
설치된 POSTGRES 데이터베이스의 포트를 변경해야 한다.

3.5. 명령 프롬프트에 텍스트 표시가 중지될 때까지 대기

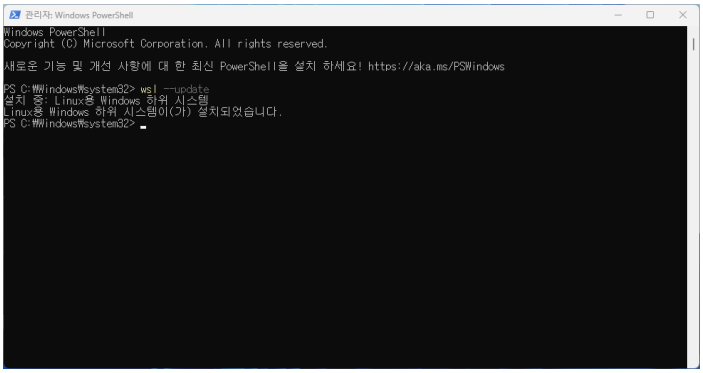
3.6. 웹 브라우저에서 http://127.0.0.1:3000 접속

※ 컴퓨터에서 웹 응용프로그램 설정 진행 화면

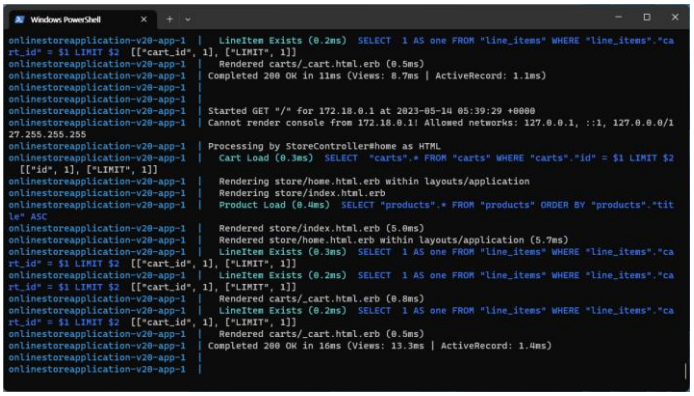
3.2.1 Docker 설치 후 실행 시 화면 **(WSL 커널 설치 필요)**



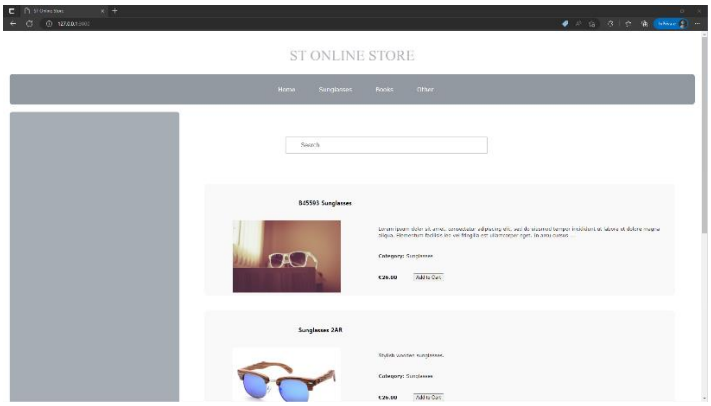
3.2.2 WSL 설치 화면



3.4.1 명령어 입력 후 화면



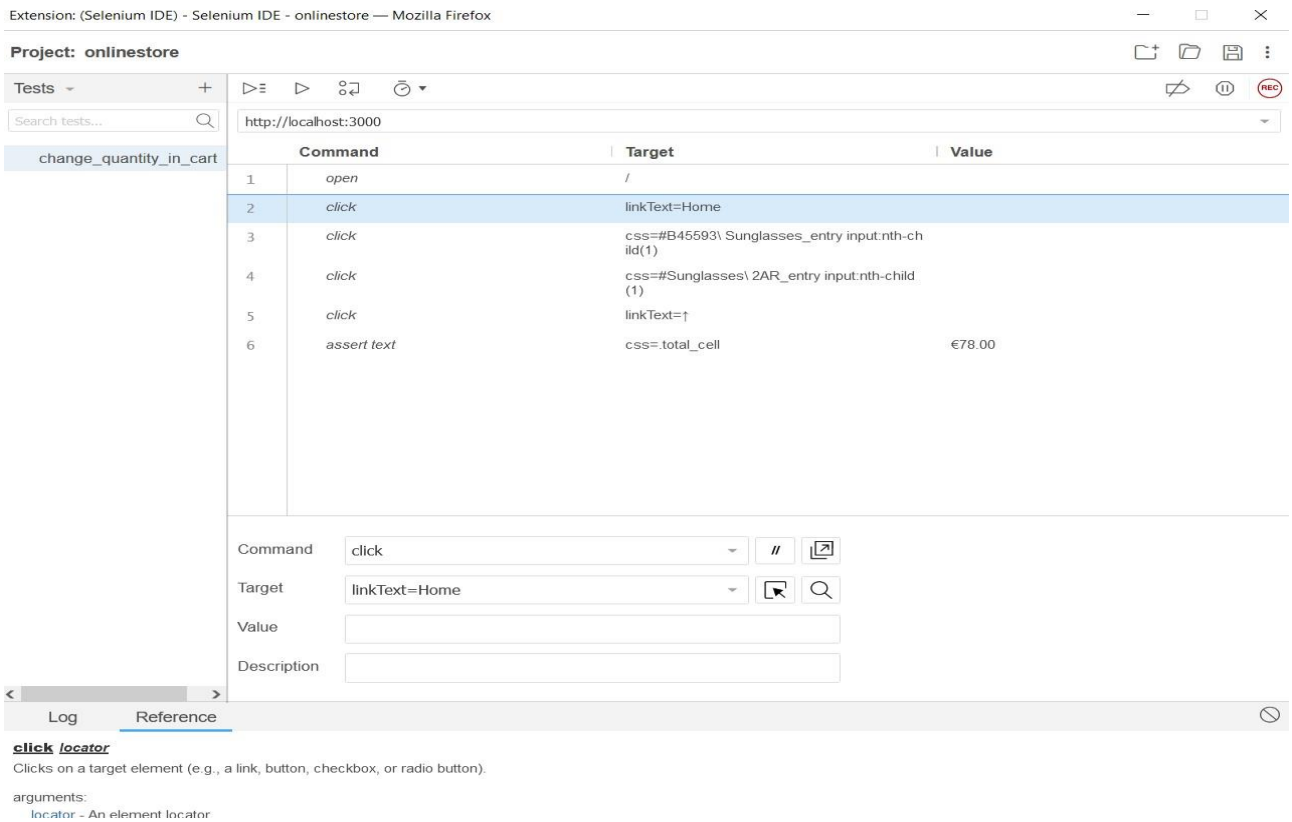
3.6.1 ST Online Store 실행 화면



4. Selenium IDE

Selenium은 웹 애플리케이션 테스트를 자동화하기 위하여 많이 사용하는 도구다. 사용이 간편하고 대부분의 웹 브라우저에서 테스트를 실행할 수 있다.

참고: 이번 실습에는 웹 기반 IDE를 설치할 필요가 없다. IDE를 사용하면 Selenium이 동작을 녹화하고 (스크립트로 기록) 테스트를 재실행 할 수 있다. 그러나 IDE에서 Selenium이 기록한 데이터는 사용자가 알아보기 어렵다. 아래 그림 참조.



실습을 위한 테스트 케이스로 두가지 제품을 카트에 추가한다. 그리고 한가지 제품의 수량을 올리고, 가격이 맞는지 확인하려고 시도한다. 그러나 가격이 정확히 계산되었다고 확인할 수 없다.

테스트를 위해 IDE를 사용하는 것은 더 간단해 보이지만, 적절한 단위 테스트를 사용하면 입/출력 및 가시적 데이터에 대한 제어를 더 잘할 수 있다. 또한 테스트를 수정하기가 쉽고 사람들이 더 쉽게 읽을 수 있다.

Selenium 설명 링크: https://www.selenium.dev/documentation/en/

5. IntelliJ에서 Selenium WebDriver 도구 셋업

※ 중요 - 과제는 Selenium 웹 드라이버를 사용하여 직접 스크립트를 작성해야 한다.

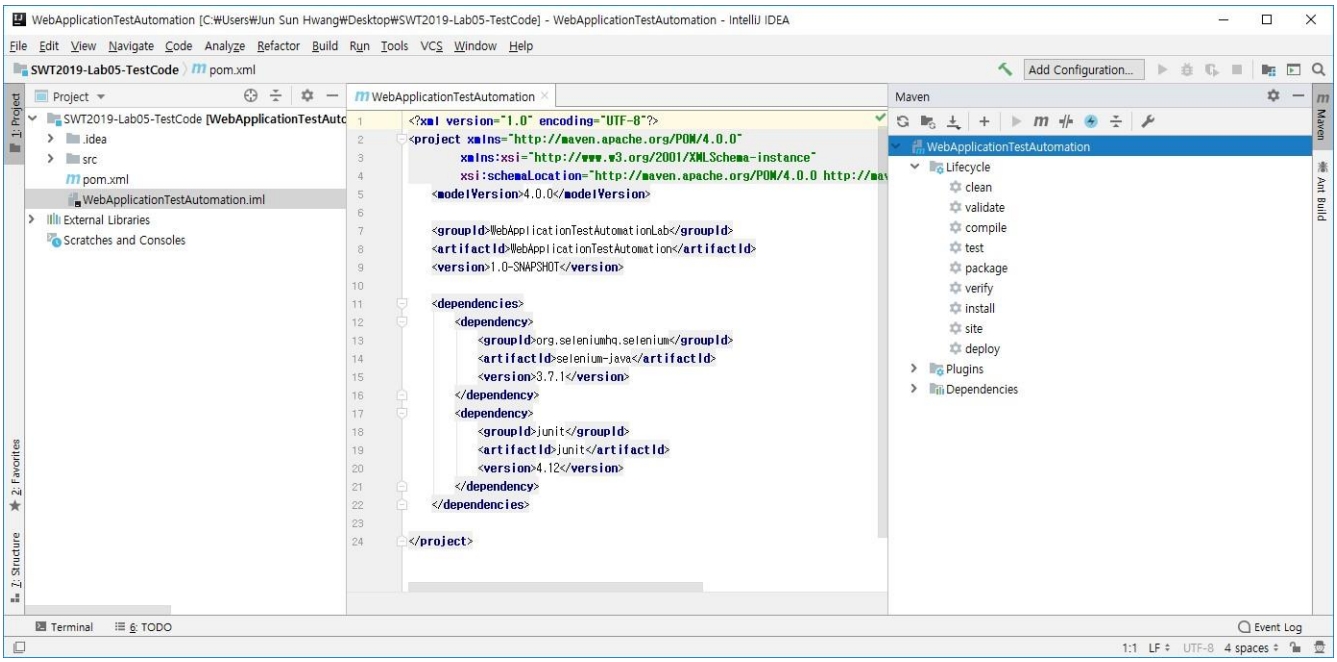
5.1. 링크에서 운영체제에 맞는 IntelliJ IDEA 다운로드.
https://www.jetbrains.com/idea/download/#section=windows

5.2. 시스템 재시작이 필요하면 재시작.

5.3. Eclass에서 Maven 프로젝트 "Lab8-TestCode.zip"을 다운로드하고 압축을 푼다.
Maven : 애플리케이션이 필요한 많은 라이브러리를 일괄적으로 관리 pom.xml 에서 설정

5.4. 프로젝트 열기(New-> Project from existing sources -> "pom.xml" 클릭) or (Open -> "pom.xml" 클릭)

5.5. 프로젝트 새로고침 (View -> Tool windows -> Maven projects -> "Reimport All Maven Projects" 클릭)
or (View -> Tool windows -> Maven 클릭) 아래 그림같이 나와야한다.



5.6. 웹 브라우저에서 selenium 테스트를 실행할 수 있도록 드라이버를 다운로드한다.
각 브라우저마다 다른 드라이버를 사용한다
(Geckodriver는 Firefox용 드라이버다. https://github.com/mozilla/geckodriver/releases)
(ChromeDriver는 Chrome용 드라이버다. https://chromedriver.chromium.org/downloads)
(다른 브라우저용 드라이버는 https://www.selenium.dev/downloads/ 에서 찾을 수 있다.)

5.7. TestHelper 클래스의 setUp() 메소드에 드라이버 경로를 추가한다.
(Testhelper 위치: test->java->TestHelper)

5.8. TestHelper 클래스에 "3.6."에서 만든 웹사이트의 URL을 추가한다.

5.9. 테스트 실행 (Build -> Build Project -> Run ->All in WebApplicationTestAutomation 클릭)

6. 테스트 코드 예제

예제 테스트 코드는 프로젝트 내부에 있다. 주어진 프로젝트에서 "TestHelper" 클래스는 setup() / tearDown() 메소드를 설정했으므로, 테스트 케이스를 확장하더라도 모든 테스트 케이스에 적용된다.

setup()과 tearDown() 메소드는 모든 테스트케이스 전/후에 호출된다. setUp()은 테스트가 실행되기전 드라이버를 초기화하며 테스트 케이스가 실패하더라도 tearDown()을 통해 드라이버를 종료한다.

만약 이전버전의 Firefox를 사용하고 있다면, geckodriver 가 호환되지 않을 수 있고, 테스트를 실행할 때 오류가 발생할 수 있다. (setUp()이 실행되지 않기 때문에 일반적으로 tearDown()에서 오류가 발생한다.) 이런 경우 Firefox 버전을 업데이트하면 해결된다.

7. 참고 문서 링크

- Assert statements are imported from jUnit:
  http://junit.sourceforge.net/javadoc/org/junit/Assert.html
- Locating Elements:
  https://www.selenium.dev/documentation/en/webdriver/locating_elements/
- Waits:
  https://www.selenium.dev/documentation/en/webdriver/waits/

8. Selenium 힌트 & 팁

8.1. 웹페이지에서 요소를 가져오는 방법

```
driver.findElement(By.id("<insert id here>"));
By.id/xpath/tagName/className
```

- XPath 소개

https://www.guru99.com/xpath-selenium.html

- Chrome Xpath Helper Extension:

https://chrome.google.com/webstore/detail/xpath-helper/hgimnogjllphhhkhlmebbmlgjoejdpjl

8.2. 웹 애플리케이션의 모든 요소에 항상 접근할 수 있지는 않다.
예를 들어 페이지가 reloads 중에 요소에 접근할 수 없음. Ex) StaleElementReferenceException(요소가 더이상 DOM에 연결되어 있지 않음). 해결방법은 드라이버가 3-4초 기다리게 하거나, 요소를 접근할 수 있을 때까지 기다리는 것이다.

```
new WebDriverWait(driver, waitForResposeTime).ignoring(
    StaleElementReferenceException.class).until(
    ExpectedConditions.elementToBeClickable(By.id(...)
);
```

9. 추가 정보

9.1. 시스템에는 몇몇 버그가 심어져 있다. 따라서 모든 기능이 정상 작동하지 않는다.

9.2. 아래에 열거된 기능(10.1)들은 별도로 테스트할 수 있다. 그러나 시스템이 최종 사용자에게 표시되는 내용은 관리자가 삽입/삭제하는 항목에 따라 다르다는 점에 유의한다.

9.3. 먼저 응용프로그램의 관리자 측면을 탐색한다. 이 응용프로그램에서 가능한 기능들을 배우고, 최종 사용자가 무엇을 보아야 하는지를 더 잘 이해할 수 있다.

9.4. 테스트는 self-cleaning 하게 작성되어야 한다. 쉽게 말해 테스트 전과 후에 테스트 대상에 변화가 있어서는 안 된다. 예를 들어 테스트가 제품 추가에 관한 것이라면 테스트 후에 제품을 삭제해야 동일한 테스트를 여러 번 실행할 수 있다.

10. 과제 (점수 15점)

10.1. 아래에 나열된 모든 기능을 다루는 단위 테스트를 작성한다. 테스트 케이스의 이름은 이름만 보고 명확히 알 수 있도록 작성하세요.

관리자 테스트:

1. 계정등록
2. 시스템에 로그인
3. 시스템에서 로그아웃
4. 계정 삭제
5. 제품 추가
6. 제품 편집
7. 제품 삭제

사용자 테스트:

1. 카트에 제품 추가
2. 카트의 제품 수량 증가 / 감소
3. 카트에서 항목 하나씩 삭제
4. 카트 물품 전체 삭제
5. 홈페이지에서 이름별로 제품 검색 및 범주별로 필터링
6. 물품 결제

10.2 만점을 받으려면 최소한 다음 사항을 제출해야 한다.

★ 20개 이상의 테스트 케이스에 5개 이상의 negative test 가 포함되어야 한다.

★ 최소 2개의 심어진 버그를 찾아야 한다. 즉. 테스트 케이스 중 2개는 fail이 나와야 한다.

★ negative test와 버그는 별도로 표시를 한다.