Rex Jones II

(Transcript) TestProject Built On Top Of Selenium

TestProject Built On Top Of Selenium

TestProject

Introduction

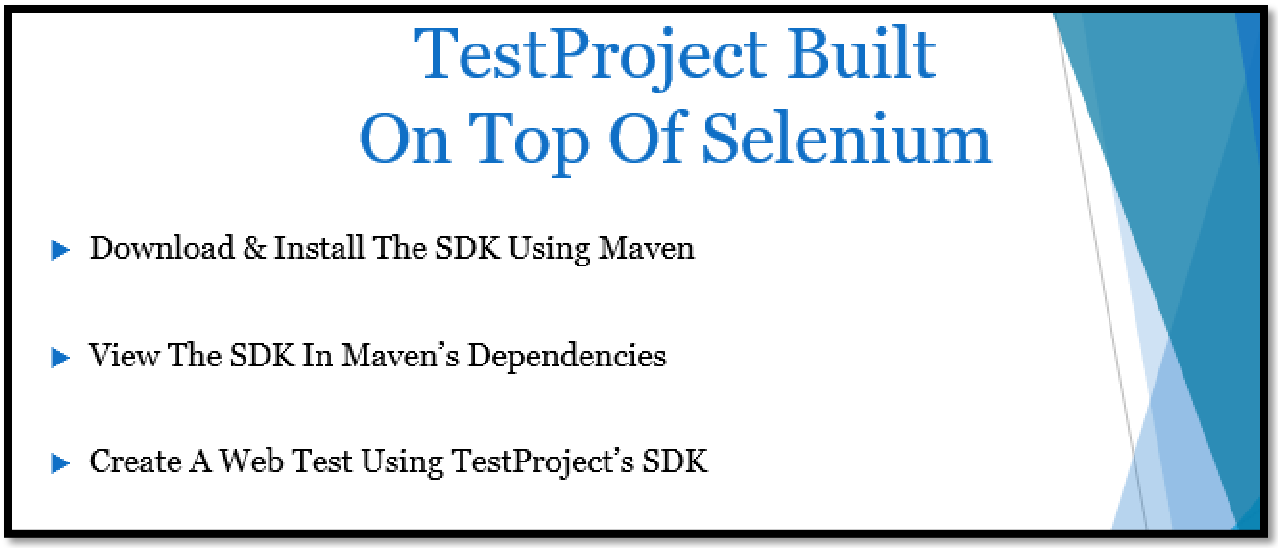
Two benefits of TestProject’s SDK are extending commands for Selenium and Appium. Selenium is used for Web Testing and Appium is used for Mobile Testing. Our focus will be Selenium. The SDK for Selenium is a wrapper meaning we have all of the benefits for Selenium plus more commands.

I wish a platform like TestProject was around when I started automation because it’s good for people who have no knowledge of programming and it’s good for people who have knowledge of programming. In this session, I’m going to write some commands that look like Selenium.

My name is Rex and I like to offer knowledge to our testing community through books, blogs, and videos. If you are new to my videos, consider subscribing to my YouTube channel and clicking the bell icon. You can also follow me on Twitter and connect with me on LinkedIn.

To get started with TestProject, we must have an account and agent. Then, we can download and install the SDK. The installation can be completed using Maven or Gradle. After the installation, we are going to look at the SDK jar file in Maven’s Dependencies. It’s going to show us all kinds of packages and classes. Last, we are going to create a Web Test using TestProject’s SDK.

Rex Jones II



TestProject Built On Top Of Selenium

- Download & Install The SDK Using Maven
- View The SDK In Maven's Dependencies
- Create A Web Test Using TestProject's SDK

Download & Install The SDK Using Maven

I'm already signed into TestProject and located at the Integrations tab. We see there are 5 coding languages: Java, C#, JavaScript, Python, and Groovy. These last 3 languages are gray because they are not available yet. Let's go ahead and download the SDK for Java. While it's downloading, let's go to GitHub for some examples. Scroll down to java-sdk-examples. This page has some good information. Copy the dependency for Maven and start using the SDK. Keep the SDK jar.

Go to Eclipse and in this pom.xml file, we see there are no other dependencies. Paste the dependency fpr TestProject.

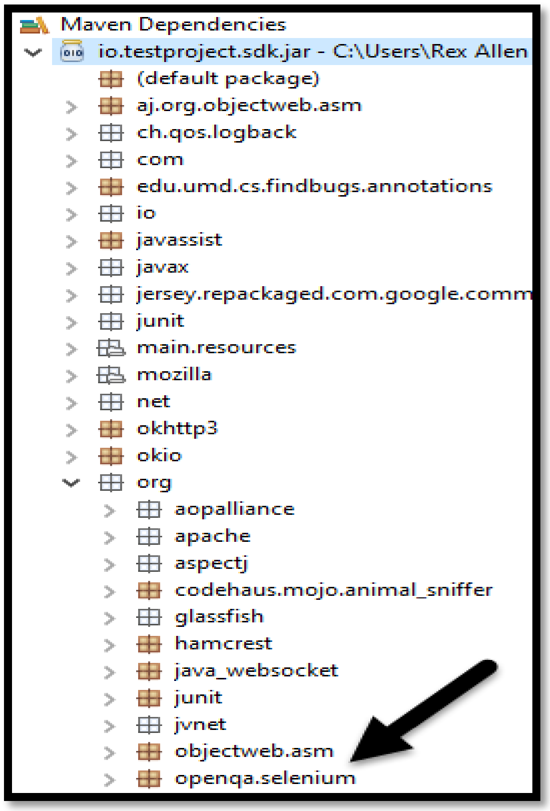
Go to my Downloads folder and we have to rename the jar file to io.testproject.sdk.jar. Copy the path by Shift and Right Click and we also have to change it for Gradle.

CTRL + SHIFT + F will format the pom.xml file. Do you see this systemPath? We have to change this to the location of our absolute path. Paste and remove the quotation marks and that's it. Go to the project, Right click, select Maven, then Update Project. Click OK

View The SDK In Maven's Dependencies

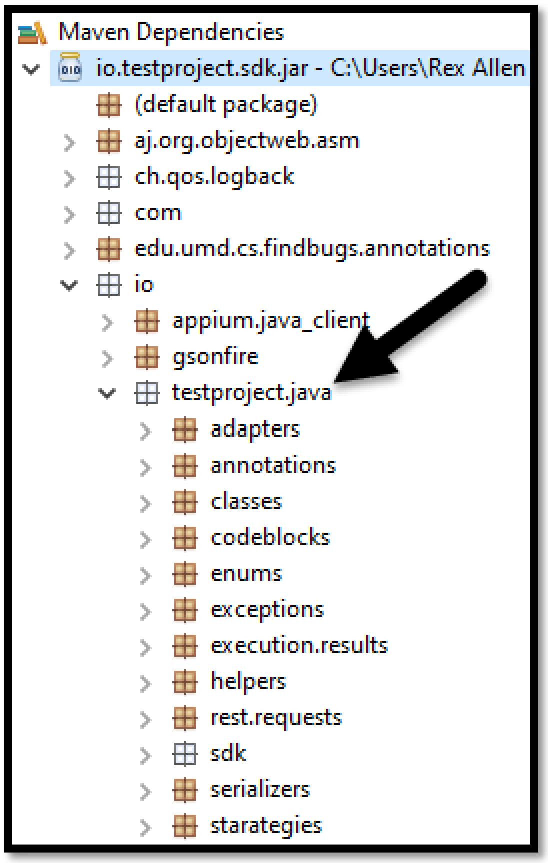
Look how Maven Dependencies has only 1 jar file which is TestProject's SDK. However, when I expand the jar file, we have packages inside of packages. Let's start by going to Selenium. org > openqa.selenium. There it is. We see WebDriver and WebElement. Although this jar file, has Selenium,

Although this jar file has Selenium, we are not going to use the Selenium package for writing our Test Script. We are going to use TestProject's SDK. This SDK jar file also has junit which is one of the Test Frameworks for Java. TestNG is the other Test Framework for Java.


```
Maven Dependencies
  io.testproject.sdk.jar - C:\Users\Rex Allen
      (default package)
    > aj.org.objectweb.asm
    > ch.qos.logback
    > com
    > edu.umd.cs.findbugs.annotations
    > io
    > javassist
    > javax
    > jersey.repackaged.com.google.comm
    > junit
    > main.resources
    > mozilla
    > net
    > okhttp3
    > okio
      org
      > aopalliance
      > apache
      > aspectj
      > codehaus.mojo.animal_sniffer
      > glassfish
      > hamcrest
      > java_websocket
      > junit
      > jvnet
      > objectweb.asm
      > openqa.selenium
```

io > testproject.java is the package we need and sdk > v2. By the end of this series, we are going to use most of these packages except for internal and support. Now, let's create our Web Test.

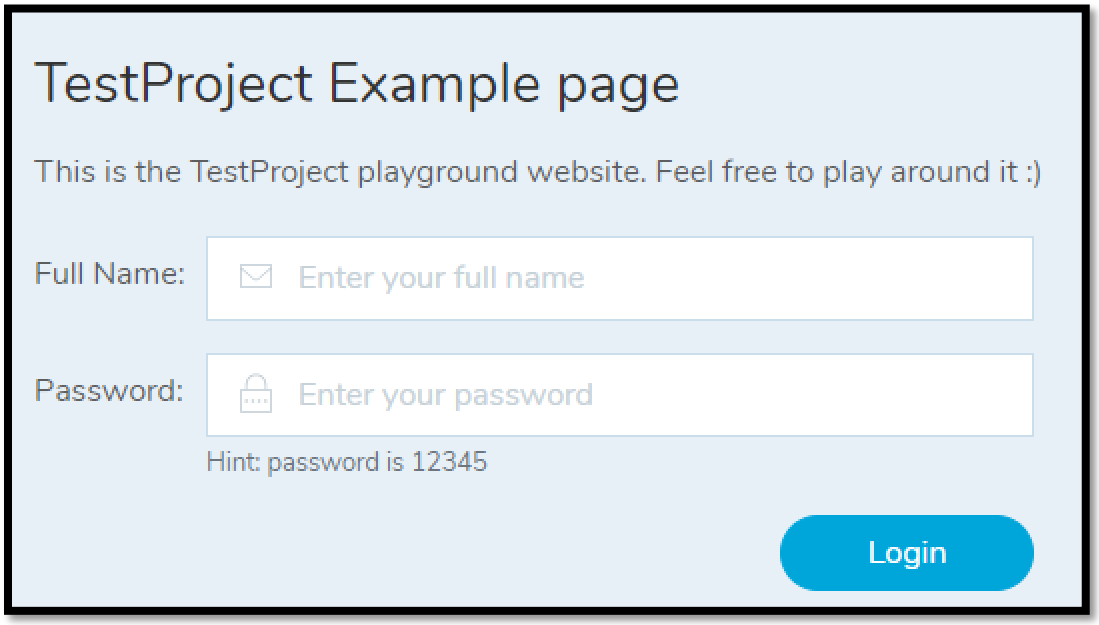
Rex Jones II



Create A Web Test Using TestProject's SDK

We are going to use TestProject's Example Page for our AUT. The Test Script will enter a name, password as 12345, then click the Login button. Inspect the WebElements. The Full Name field has a value for name as the id, inspect Password and we see password as the value for id. Inspect login button and login button has the value for id.

Rex Jones II


TestProject Example page

This is the TestProject playground website. Feel free to play around it :)

Full Name: Enter your full name

Password: Enter your password

Hint: password is 12345

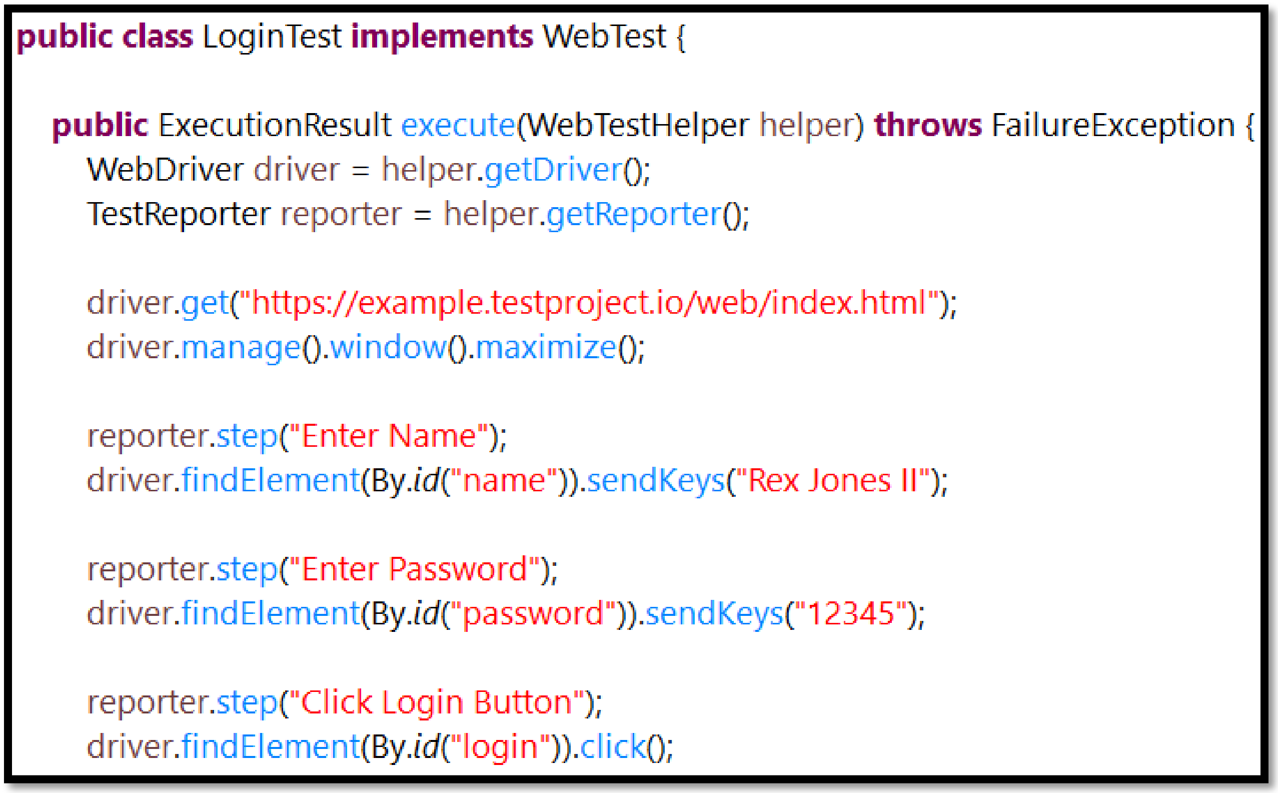
Test Class

In Eclipse, I have a Test Class and a class to run the Test Class. Every Test Class must implement an interface. We are testing a web application so LoginTest implements WebTest. Import WebTest then add the unimplemented method. This execute method returns ExecutionResult. Let's write a return statement that says ExecutionResult.PASSED.

If you know Selenium then the remaining steps will be a piece of cake. WebDriver driver; then import (CTRL+SHIFT+O). We see 2 types but here's the difference. Selenium is an interface and TestProject SDK is a class. No big deal but we are going to use the SDK. Do you see the parameter WebTestHelper helper? WebTestHelper is a class that provides access to the driver and test resources for the web. We write = helper.getDriver();.

Load the example page driver.get("https://example.testproject.io/web/index.html"). Maximize the window – driver.manage().window().maximize(). Next, we enter the Full Name and Password. To help with debugging, the SDK provides a TestReporter class that extends the Reporter class. Write TestReporter reporter object = helper.getReporter(); Import the class. The TestReporter class provides access to methods for reporting steps. getReporter is a method that provides a reporter instance. First step is to reporter.step: The description ("Enter Name") / driver.findElement(By.id("name")).sendKeys("Rex Jones II");. Second step is to enter the password. reporter.step("Enter Password") / driver.findElement(By.id("password")).sendKeys("12345"); Last step is reporter.step("Click Login Button") / driver.findElement(By.id("login")).click(); These steps will log into the application.

Rex Jones II



```
public class LoginTest implements WebTest {

  public ExecutionResult execute(WebTestHelper helper) throws FailureException {
    WebDriver driver = helper.getDriver();
    TestReporter reporter = helper.getReporter();

    driver.get("https://example.testproject.io/web/index.html");
    driver.manage().window().maximize();

    reporter.step("Enter Name");
    driver.findElement(By.id("name")).sendKeys("Rex Jones II");

    reporter.step("Enter Password");
    driver.findElement(By.id("password")).sendKeys("12345");

    reporter.step("Click Login Button");
    driver.findElement(By.id("login")).click();
```

Let’s confirm we actually logged into TestProject example page. Go back to the AUT. Name is James / Password is 12345 / click the Login Button. The name is here in this message. Inspect and we see greetings as the value for id

String greeting = driver.findElement(By.id(“greetings”)).getText(); Let’s also print the value. sysout(greeting). We are getting to the good part! Verify the greeting by writing an Assert statement dot (.) assertTrue and we see Assertions for TestNG and JUnit. We are going to use the Assert statement for TestNG by writing greeting.contains(“Rex”). Look at the import statements. The import statements have testproject. We do not see Selenium for the WebDriver but we see it for the By Class. We have one Selenium statement and one TestNG statement.

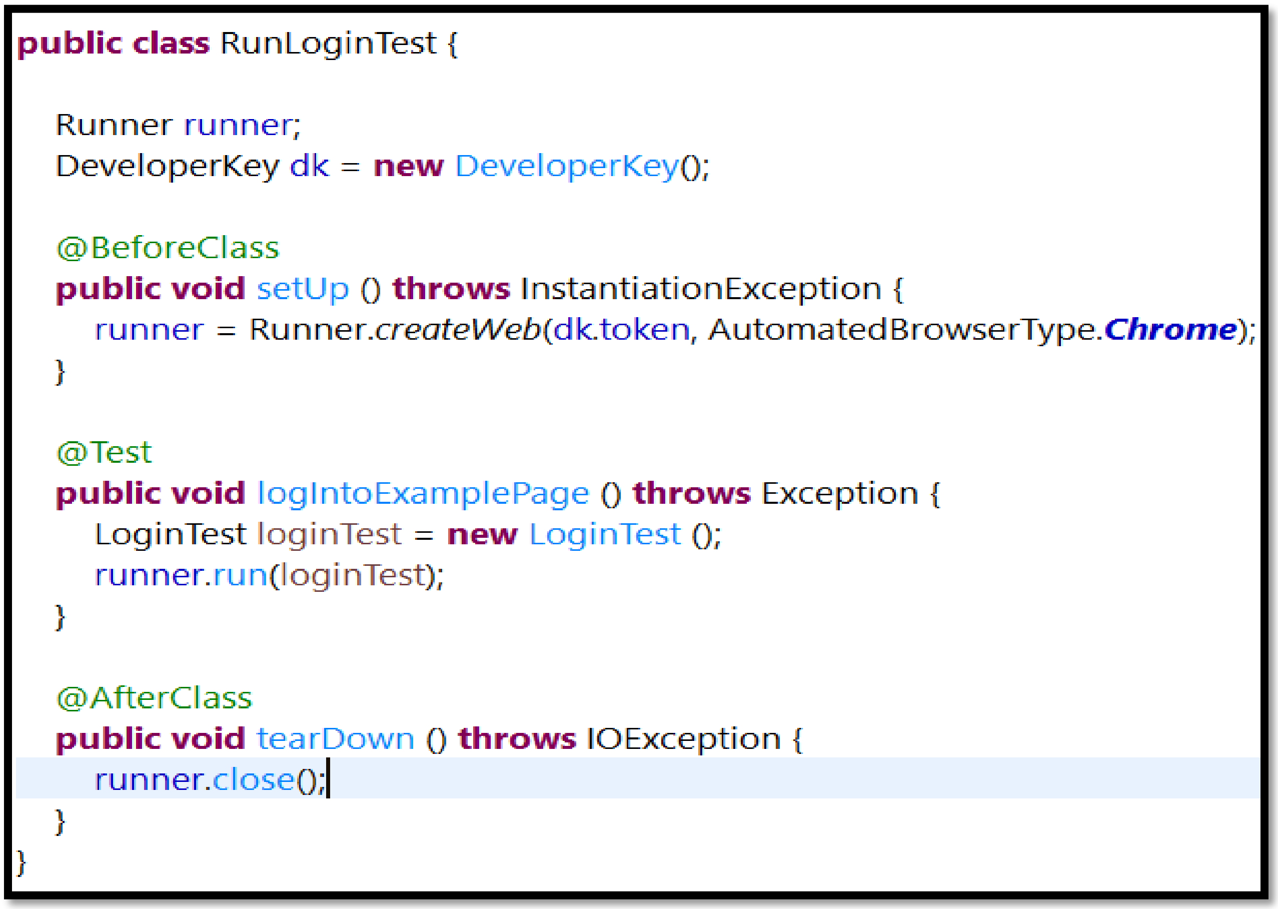
Run Test Class

The last class is the runner class. We have to make sure our Test Agent is running. So, let’s write Runner runner; Import Runner from the TestProject package sdk. @BeforeClass / public void setup () { runner = Runner.createWeb There it is. The Developer Token got to be retrieved from TestProject. This method create web - creates the runner and receives 2 parameters: A token and a browser type. Go back to Integrations tab and we get the Developer Key. Up under runner, I’m going to write DeveloperKey dk = new DeveloperKey(); This class has my token.

I added the token to this class because it’s a special token to whoever is logged into the application to do Addons and creating test. Feel free to paste your token in the developer key or put it write here under Developer token. dk.token, AutomatedBrowserType is Chrome. BeforeClass – TestNG annotation. Throws declaration for InstantiationException. Now, we have our Test Method.

@Test / public void logIntoExamplePage () { } / Import the Test Annotation from TestNG. / LoginTest loginTest = new LoginTest() / runner.run(loginTest); Add a throws declaration Exception.

Now, we have the @AfterClass / public void tearDown () / Import the annotation and we are going to close. runner.close; / Add a throws declaration for IOException.



```
public class RunLoginTest {

   Runner runner;
   DeveloperKey dk = new DeveloperKey();

   @BeforeClass
   public void setUp () throws InstantiationException {
      runner = Runner.createWeb(dk.token, AutomatedBrowserType.Chrome);
   }

   @Test
   public void logIntoExamplePage () throws Exception {
      LoginTest loginTest = new LoginTest ();
      runner.run(loginTest);
   }

   @AfterClass
   public void tearDown () throws IOException {
      runner.close();
   }
}
```

Now, let's run. We see the test Passed and 3 steps: Enter Name Passed, Enter Password Passed, and Click Login Button Passed. The Console show the print statement

Hello Rex Jones II, let's complete the test form. We will complete the test form in the next video. I'm going to show you how to take screenshots, reuse the same steps, and generate code without writing your own code. You can do it all through TestProject!!!