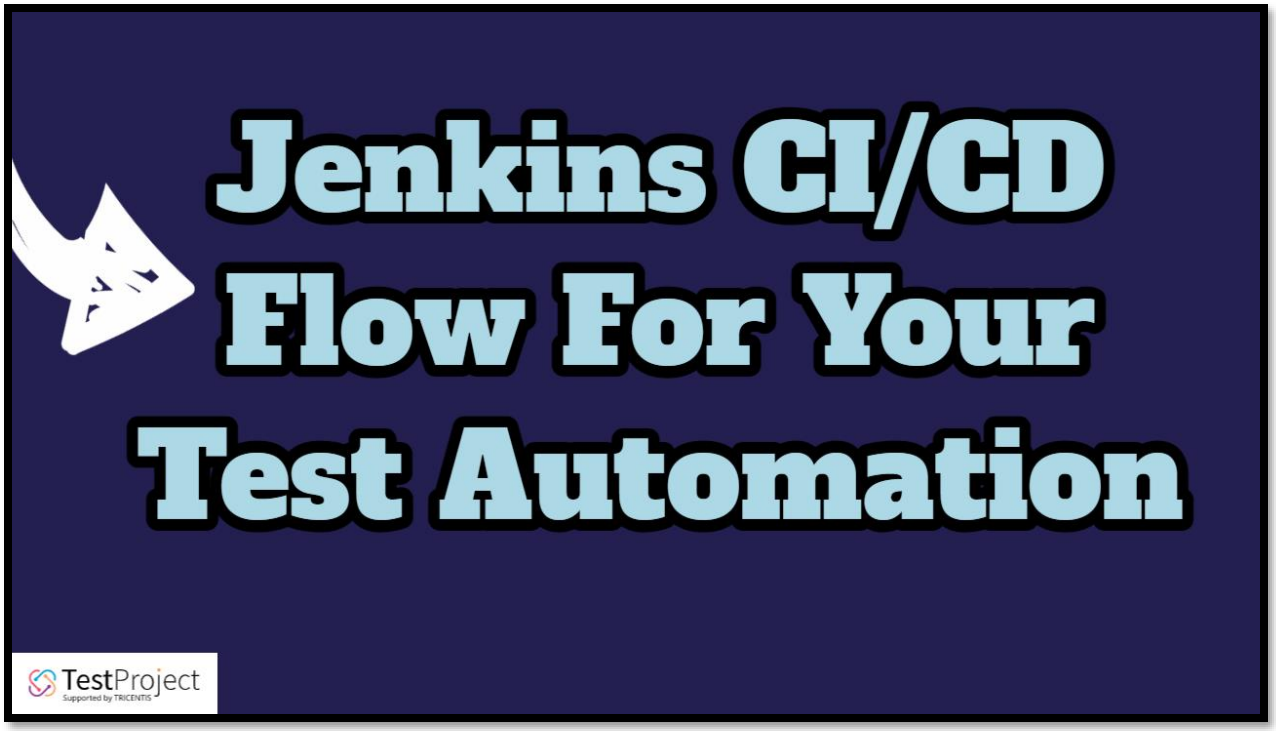
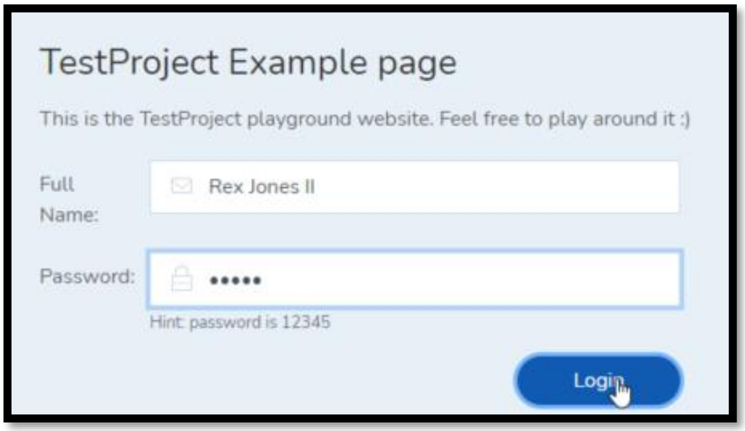
How To Achieve CI/CD Pipeline Using TestProject & Jenkins



In this session, we are going to look at the complete CI/CD Flow for Test Automation. I will demonstrate how to use and achieve the CI/CD pipeline using TestProject's OpenSDK and Jenkins.

First, we will walkthrough my Page Object Model. After the code, we will see how to install plugins for TestProject, TestNG, and Gradle followed by setting up the pipeline environment, pulling the test from Git, then running the test. This video is for Java but Bas has a webinar for Python.

Our Application Under Test is TestProject's Example Page. We are going to enter the Full Name "Rex Jones II", Password "12345", then click the Login button. On the next page, we select a country like "Algeria", enter address "1234 Test", email can be "Rex@GMail.com", and phone "214-225-1234" then click Save. Last is to verify the Saved message.



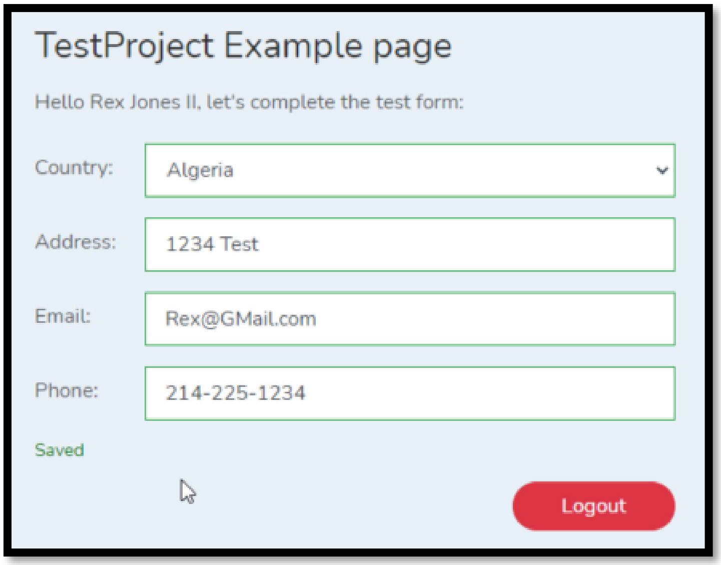
TestProject Example page

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

Full Name: Rex Jones II

Password: •••••

Hint: password is 12345



TestProject Example page

Hello Rex Jones II, let's complete the test form:

Country: Algeria

Address: 1234 Test

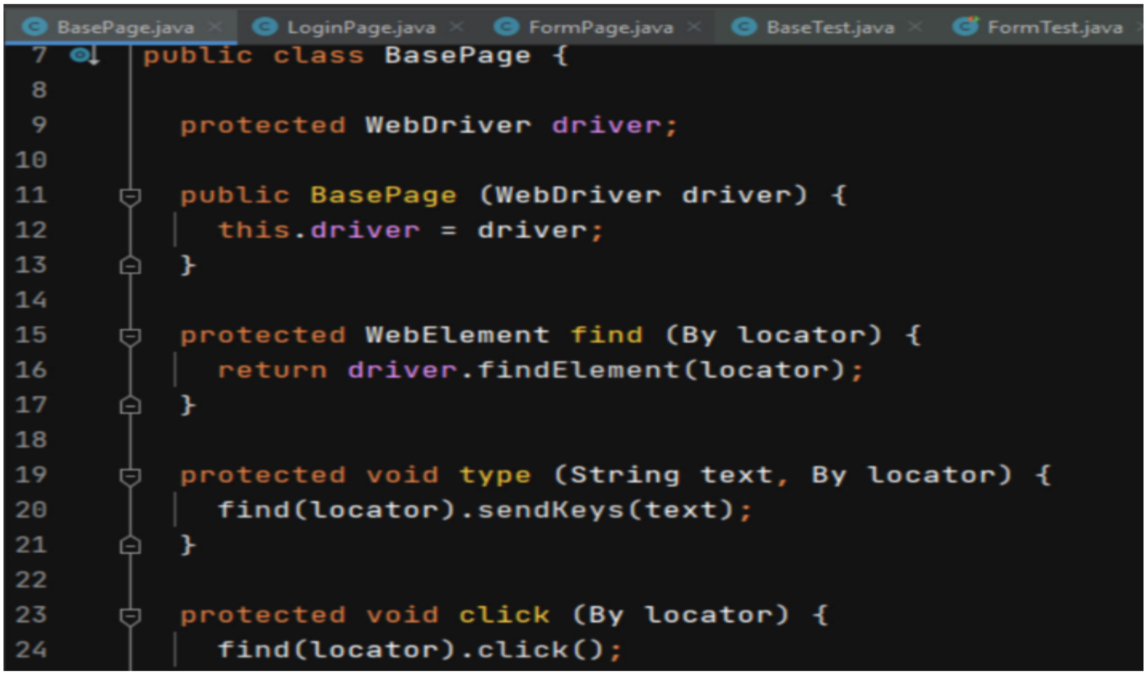
Email: Rex@GMail.com

Phone: 214-225-1234

Saved

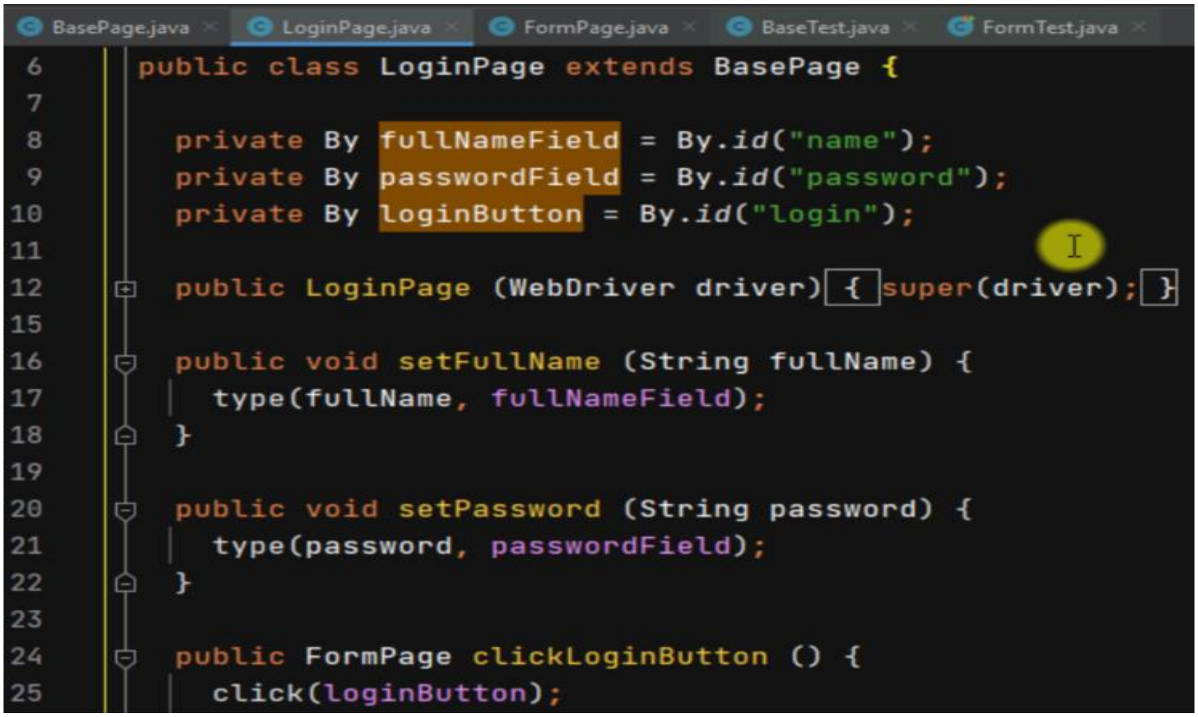
Page Object Model

Next is the Page Object Model. I have 5 classes. The BasePage has methods to find, type, and click an element.



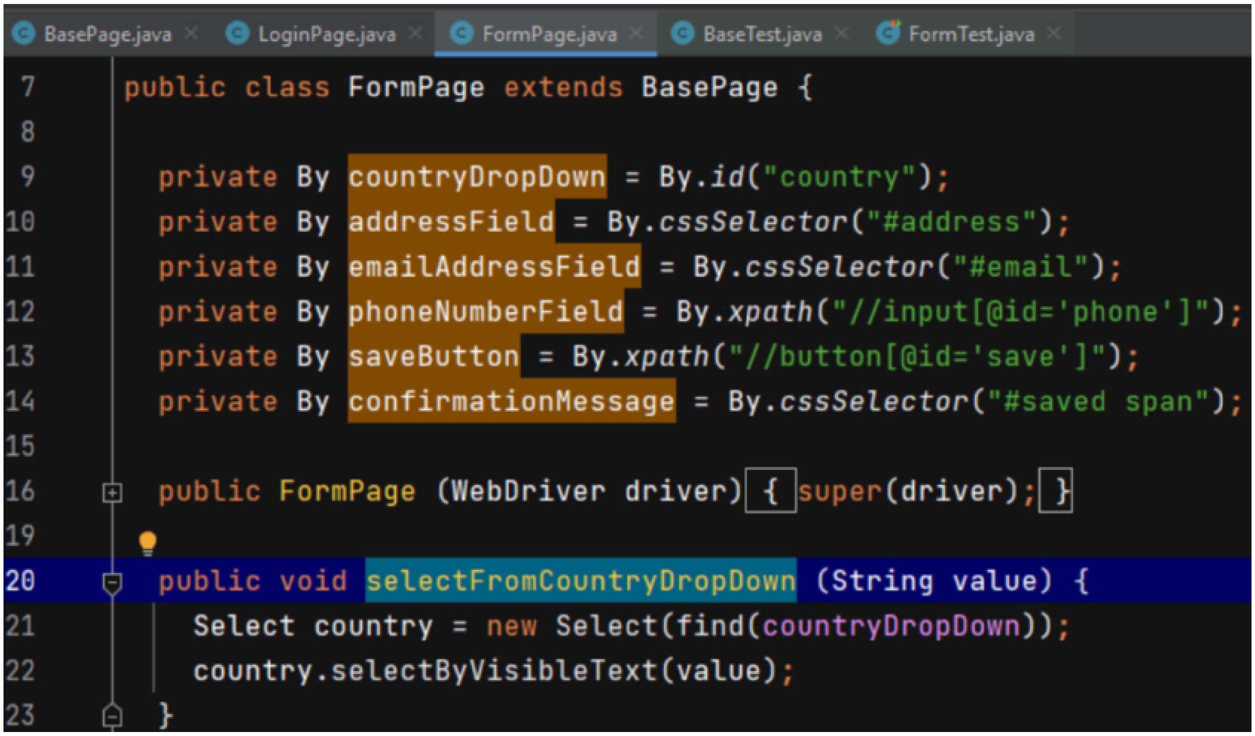
```java
public class BasePage {

  protected WebDriver driver;

  public BasePage (WebDriver driver) {
    this.driver = driver;
  }

  protected WebElement find (By locator) {
    return driver.findElement(locator);
  }

  protected void type (String text, By locator) {
    find(locator).sendKeys(text);
  }

  protected void click (By locator) {
    find(locator).click();
```

The Login Page has the fields and methods to set the Full Name, set the Password, and click the Login button.

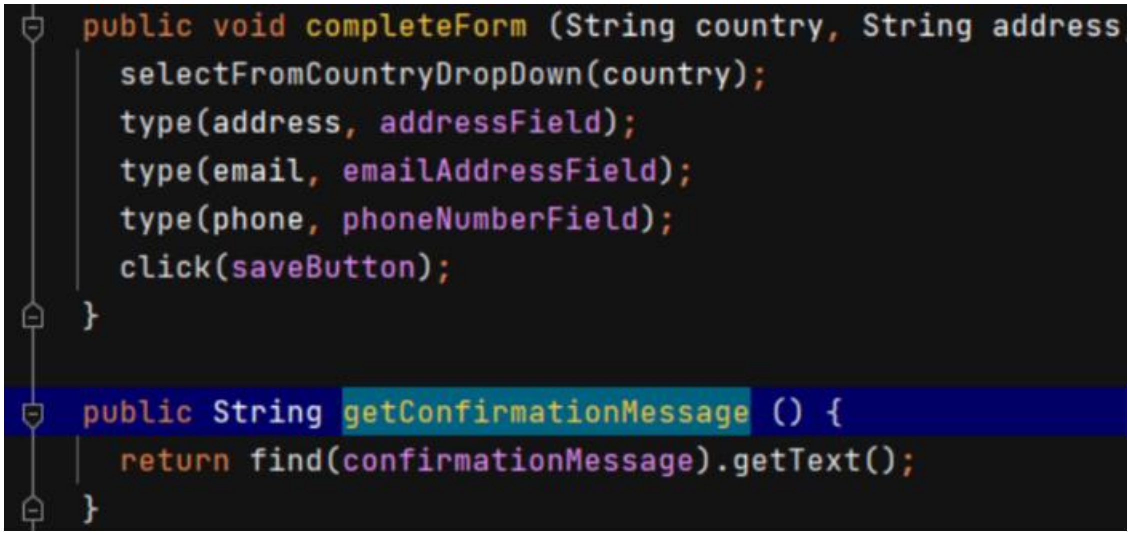


```java
public class LoginPage extends BasePage {

  private By fullNameField = By.id("name");
  private By passwordField = By.id("password");
  private By loginButton = By.id("login");

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

  public void setFullName (String fullName) {
    type(fullName, fullNameField);
  }

  public void setPassword (String password) {
    type(password, passwordField);
  }

  public FormPage clickLoginButton () {
    click(loginButton);
```

The Form Page also has fields and methods. These methods include selectFromCountryDropDown which I use in the complete form method as the first one to select a country. Also, get the confirmation message.

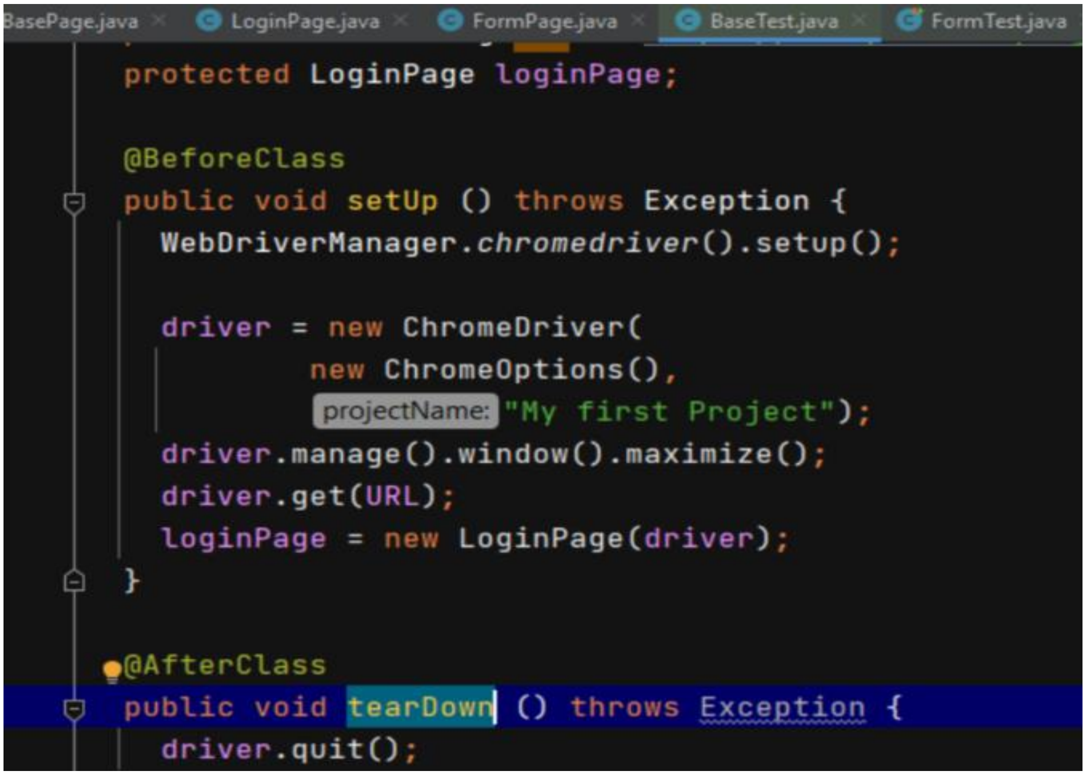


```java
public class FormPage extends BasePage {

  private By countryDropDown = By.id("country");
  private By addressField = By.cssSelector("#address");
  private By emailAddressField = By.cssSelector("#email");
  private By phoneNumberField = By.xpath("//input[@id='phone']");
  private By saveButton = By.xpath("//button[@id='save']");
  private By confirmationMessage = By.cssSelector("#saved span");

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

  public void selectFromCountryDropDown (String value) {
    Select country = new Select(find(countryDropDown));
    country.selectByVisibleText(value);
  }
```



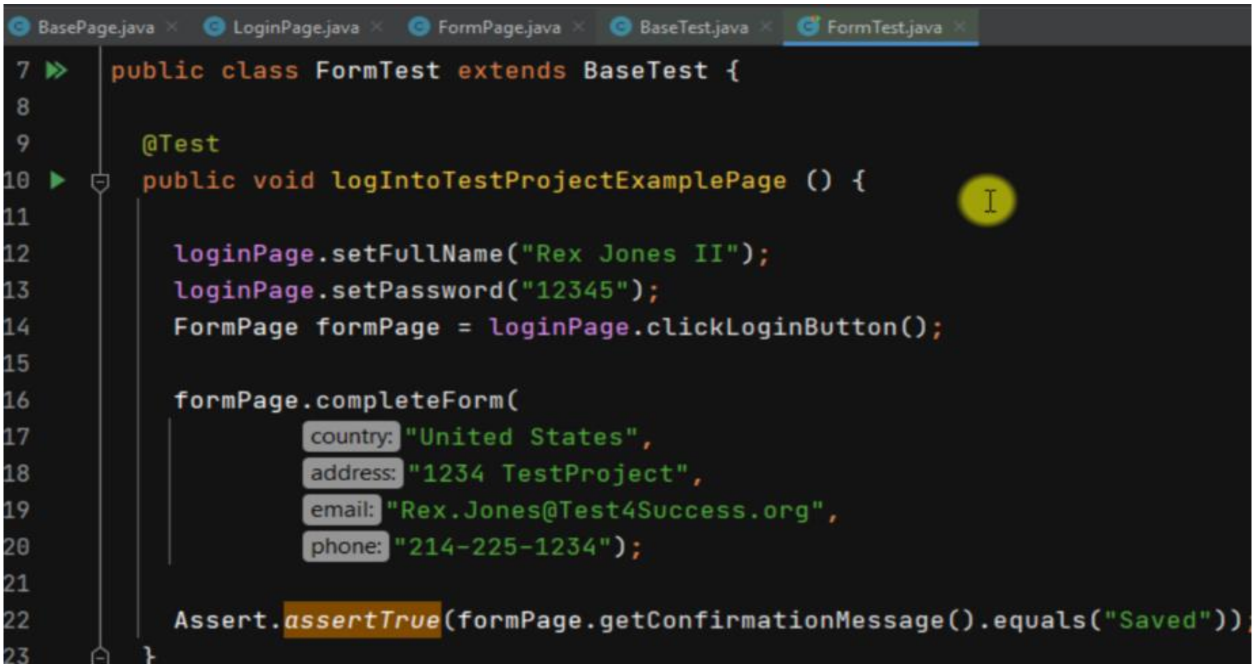
```java
public void completeForm (String country, String address
  selectFromCountryDropDown(country);
  type(address, addressField);
  type(email, emailAddressField);
  type(phone, phoneNumberField);
  click(saveButton);
}

public String getConfirmationMessage () {
  return find(confirmationMessage).getText();
}
```

Next is the BaseTest which sets up and tear down our test. The Project Name is My first Project.



```
    protected LoginPage loginPage;

    @BeforeClass
    public void setUp () throws Exception {
      WebDriverManager.chromedriver().setup();

      driver = new ChromeDriver(
              new ChromeOptions(),
              projectName: "My first Project");
      driver.manage().window().maximize();
      driver.get(URL);
      loginPage = new LoginPage(driver);
    }

    @AfterClass
    public void tearDown () throws Exception {
      driver.quit();
```

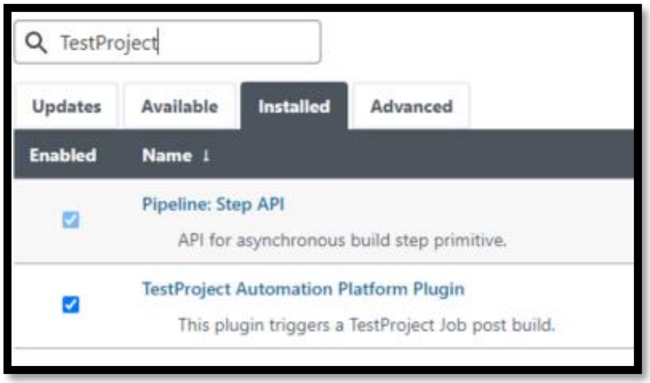
Last class is the FormTest where we add the logic for our Test Script. On the login page, we set the full name, set the password, and click the login button. Now we also do something else in this same Test Script. We are going to complete the form by selecting a country, entering an address, email the email and phone. We also are going to assert the Confirmation Message.



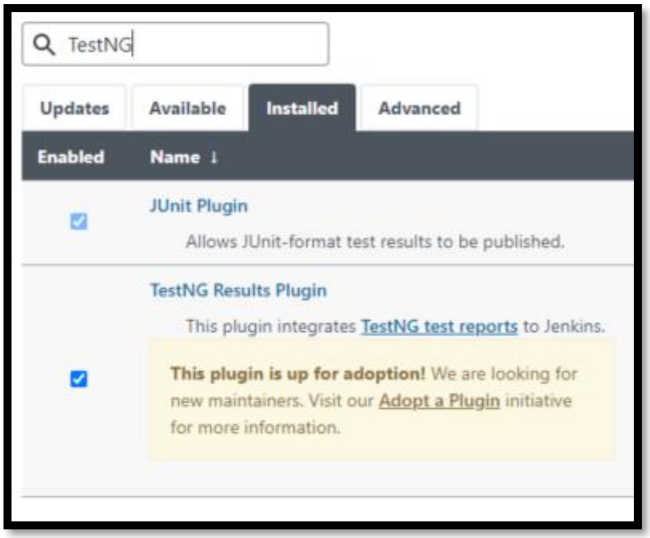
```
public class FormTest extends BaseTest {

  @Test
  public void logIntoTestProjectExamplePage () {

    loginPage.setFullName("Rex Jones II");
    loginPage.setPassword("12345");
    FormPage formPage = loginPage.clickLoginButton();

    formPage.completeForm(
            country: "United States",
            address: "1234 TestProject",
            email: "Rex.Jones@Test4Success.org",
            phone: "214-225-1234");

    Assert.assertTrue(formPage.getConfirmationMessage().equals("Saved"));
  }
```

Install Plugins

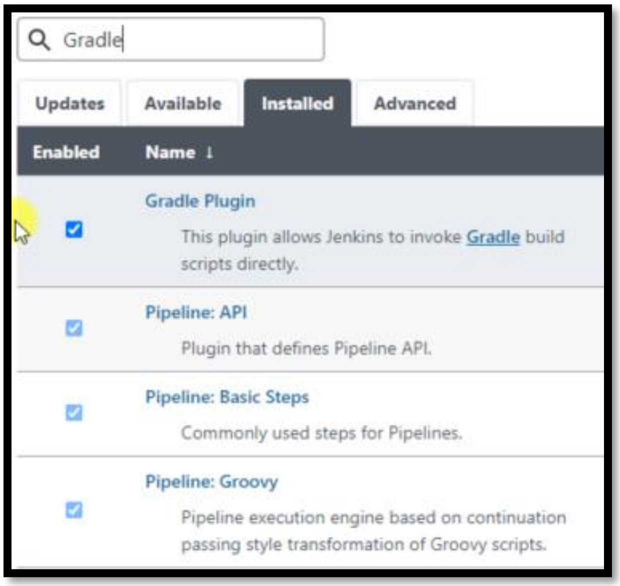
Next is the plugins. We must make sure Jenkins is up and running. In the browser, we have Jenkins running on our local host which is 8080. We go to Manage Jenkins and select Manage Plugins. If you do not have the required plugins then select Available. After selecting your plugins, you have the option of selecting Install without restart or Download now and install after restart. I'm going to the Installed tab since I have the plugins and you will see the plugins we need. Type TestProject and we see the plugins. Type TestNG and we see 2 more plugins. Last is Gradle and we have more plugins. Out of the box, Jenkins is great but plugins help enhance the functionality of Jenkins.



Enabled	Name ↓
☑	Pipeline: Step API API for asynchronous build step primitive.
☑	TestProject Automation Platform Plugin This plugin triggers a TestProject Job post build.



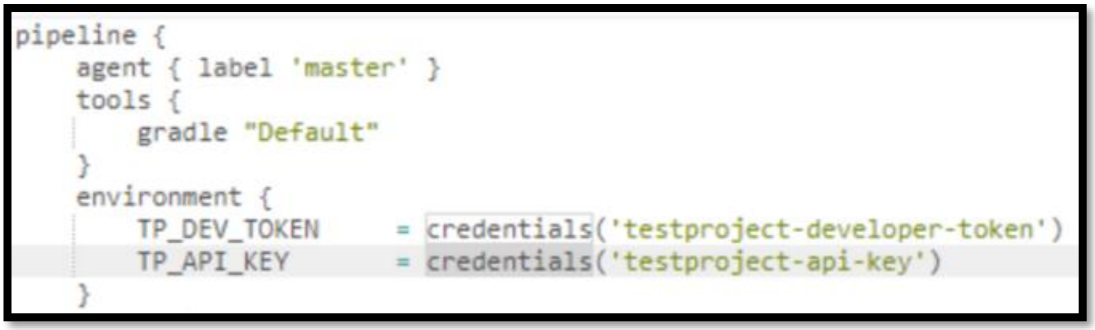
Enabled	Name ↓
☑	JUnit Plugin Allows JUnit-format test results to be published.
☑	TestNG Results Plugin This plugin integrates TestNG test reports to Jenkins. **This plugin is up for adoption!** We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.



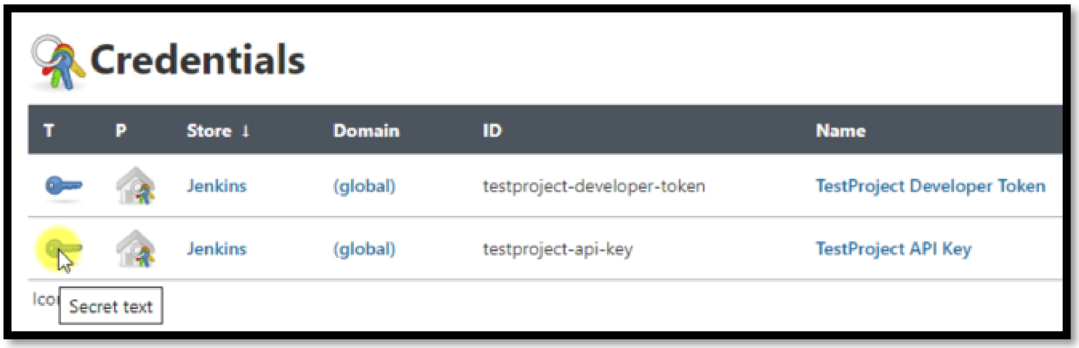
Enabled	Name ↓
☑	Gradle Plugin This plugin allows Jenkins to invoke Gradle build scripts directly.
☑	Pipeline: API Plugin that defines Pipeline API.
☑	Pipeline: Basic Steps Commonly used steps for Pipelines.
☑	Pipeline: Groovy Pipeline execution engine based on continuation passing style transformation of Groovy scripts.

Set-up Pipeline Environment

Next we are going to set up the pipeline environment. Go back to the Dashboard and view my TestProject Pipeline. Configure will show you my set up then select Pipeline. Do you see the environment section? It has TP_DEV_TOKEN and TP_API_KEY as the environment variables equal to credentials. Credentials is the helper method to support secret information. Let me show you. Go back to the Dashboard select Manage Jenkins and Manage Credentials. This key shows Secret text for testproject-developer-token and testproject-api-key. We get the Token and API Key from TestProject. In this Integrations menu. We see #1 is where we get the Developer Token. For the API Key, we select API then Create API Key. The purpose of setting up an environment is so our Jenkins job has knowledge of our Token and API Key.



```
pipeline {
    agent { label 'master' }
    tools {
        gradle "Default"
    }
    environment {
        TP_DEV_TOKEN     = credentials('testproject-developer-token')
        TP_API_KEY       = credentials('testproject-api-key')
    }
```



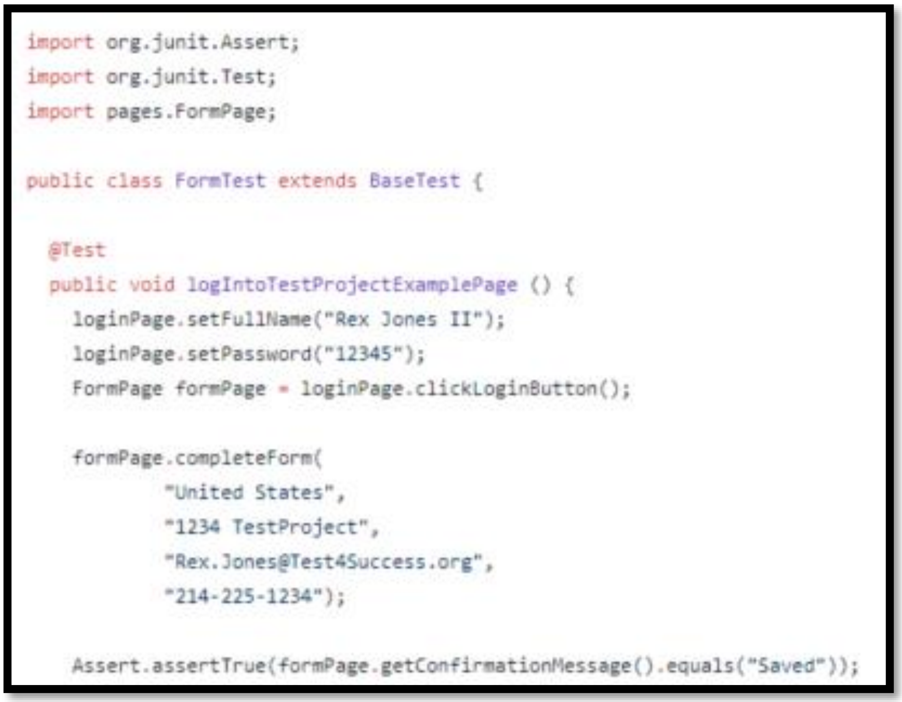
Credentials

T	P	Store ↓	Domain	ID	Name
		Jenkins	(global)	testproject-developer-token	TestProject Developer Token
		Jenkins	(global)	testproject-api-key	TestProject API Key

Ico Secret text

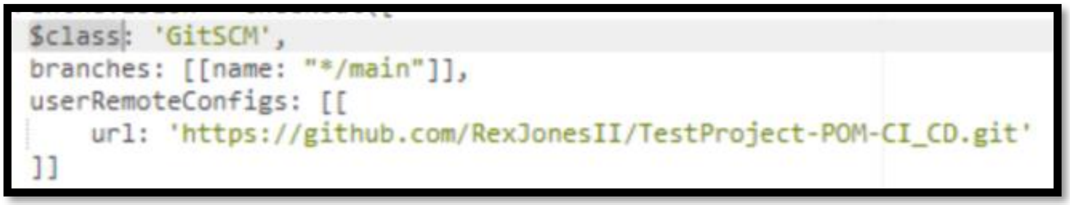
Pull Test From Git

Next is pulling the test from Git. Jenkins support different Source Control Management Tools like Subversion and Git. They help us to keep track of our source code history. A couple of benefits include collaboration with teammates and access to the latest code version. When it comes to the code version, if something is wrong, we can always go back to the previous version. This is GitHub which allows us to manage Git repositories. The Page Object Model code is located in the src folder. You can watch videos 144 – 146 to see step-by-step how I create the Page Object Model. The Page Objects are located in main/java/pages and the tests are in test/java. Here's the BaseTest and FormTest. This is the same Test Script from Intelli J.



```
import org.junit.Assert;
import org.junit.Test;
import pages.FormPage;

public class FormTest extends BaseTest {

  @Test
  public void logIntoTestProjectExamplePage () {
    loginPage.setFullName("Rex Jones II");
    loginPage.setPassword("12345");
    FormPage formPage = loginPage.clickLoginButton();

    formPage.completeForm(
            "United States",
            "1234 TestProject",
            "Rex.Jones@Test4Success.org",
            "214-225-1234");

    Assert.assertTrue(formPage.getConfirmationMessage().equals("Saved"));
```

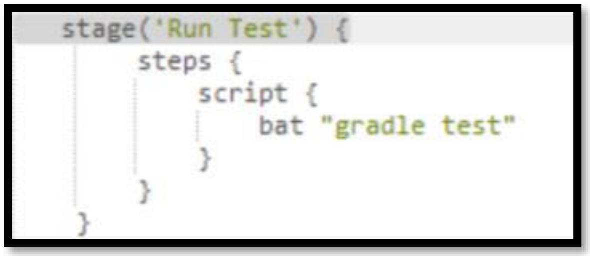
The TestProject-POM-CI_CD url is important for my Jenkins Pipeline. Go back to Dashboard for Jenkins and I will show you the TestProject Pipeline, Configure then Pipeline and we see th url matches the same url from GitHub. Do you see GitSCM for $class? GitSCM stands for Git Source Control Management. These are the steps for pulling our test from Git.



```
$class: 'GitSCM',
branches: [[name: "*/main"]],
userRemoteConfigs: [[
    url: 'https://github.com/RexJonesII/TestProject-POM-CI_CD.git'
]]
```

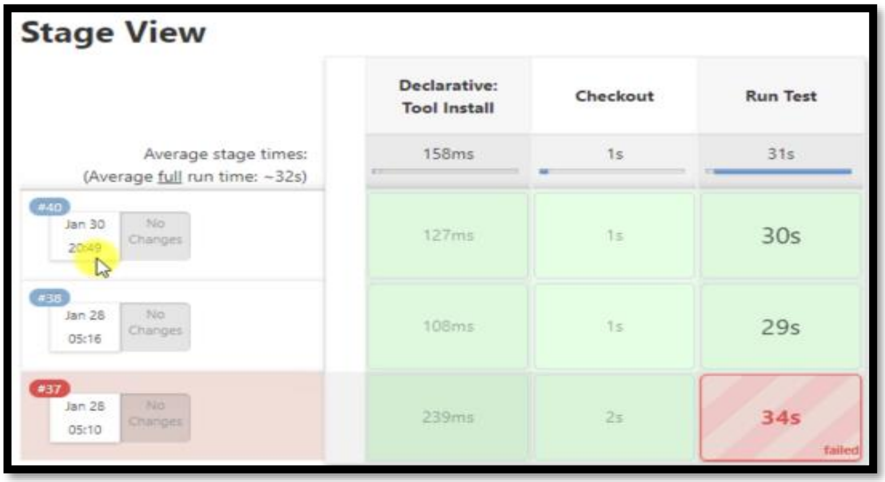
Run The Test

Next is to run the test from Jenkins. An example of the pipeline is available on TestProject's Documentation page. It's called "Using OpenSDK within CI/CD" and go to Java Example Test. This example is similar to my pipeline. Thanks to Ran from TestProject. He helped with my configuration. That's a bonus with TestProject. They are happy to help, just contact them and they will make time to resolve any problem. Great Support. Have a look around and ask us anything. We are always here to help. For the Jenkins pipeline, the Run Test stage. You will see it shows gradle test set up.



```
stage('Run Test') {
    steps {
        script {
            bat "gradle test"
        }
    }
}
```

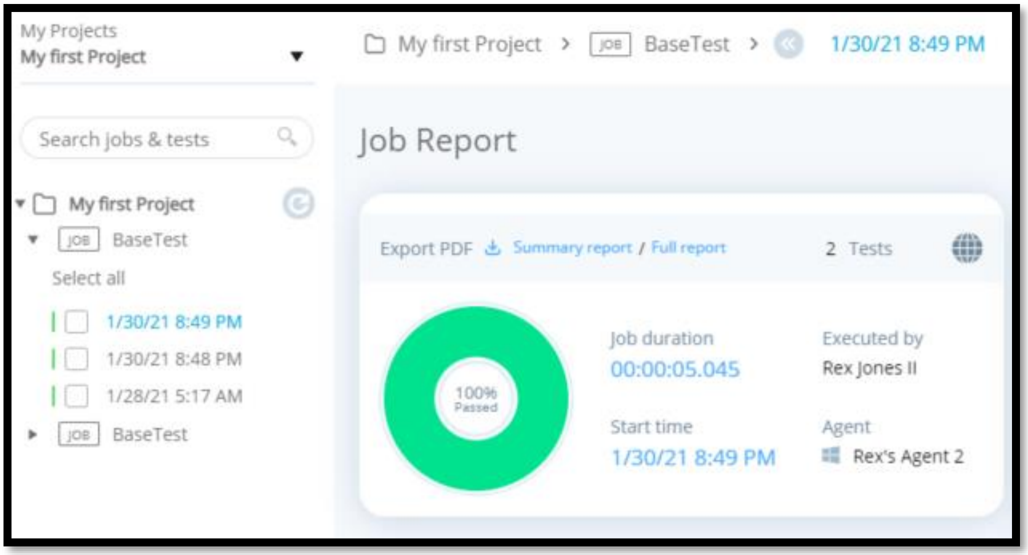
We are going to run by going back to the Dashboard. We can select the drop down for TestProject Pipeline and click Build Now or just click TestProject Pipeline drop then we see Click Build Now. The last execution we saw happened January 30 at 8:49 PM. We see the last step #37 Failed. It all shows Green and it Passed.



Stage View

	Declarative: Tool Install	Checkout	Run Test
Average stage times: (Average full run time: ~32s)	158ms	1s	31s
#40 Jan 30 20:49 No Changes	127ms	1s	30s
#38 Jan 28 05:16 No Changes	108ms	1s	29s
#37 Jan 28 05:10 No Changes	239ms	2s	34s failed

Let's go to TestProject then Reports. We see My first Project has a 100% Pass rate. Now, we are going to click an individual test and drill down and we see the 8:49 when the text executed and pass.



□ 1/30/21 8:49 PM
□ 1/30/21 8:48 PM
□ 1/28/21 5:17 AM

Job Report

2 Tests

Job duration
00:00:05.045

Executed by
Rex Jones II

Start time
1/30/21 8:49 PM

Agent
Rex's Agent 2

In the next session, we are going to walk through the reports from TestProject. Thanks for watching and I will see you in the next session.

Contact Info

- ✔ Email Rex.Jones@Test4Success.org
- ✔ YouTube https://www.youtube.com/c/RexJonesII/videos
- ✔ Facebook https://facebook.com/JonesRexII
- ✔ Twitter https://twitter.com/RexJonesII
- ✔ GitHub https://github.com/RexJonesII/Free-Videos
- ✔ LinkedIn https://www.linkedin.com/in/rexjones34/