Automation Testing Framework Build Steps

1. Create Maven test environment
   1. In terminal, move to preferred directory, then enter the following:
      1. mvn archetype:generate –DgroupId={project-packaging-name} –DartifactId={project-name} –DarchetypeArtifactId=maven-archetype-quickstart –DinteractiveMode=false
2. Connect project to Eclipse
   1. In terminal, move into project directory, then enter the following:
      1. mvn eclipse:eclipse
3. Import project in Eclipse
   1. Open Eclipse
   2. Click File in top navigation
   3. Click Import in dropdown menu
   4. Click Existing Projects into Workspace option or Existing Maven Projects
   5. Click Next
   6. Click Browse, find/click project directory and click OK
   7. Click Finish
4. Configure framework dependencies (.jar files) in pom.xml file
   1. Go to <https://mvnrepository.com>
   2. Search for selenium java
   3. Select latest stable version and copy dependency xml
   4. In Eclipse project, open pom.xml file
   5. If included, remove junit dependency within dependencies tag
   6. Add selenium dependency to dependencies tag
   7. Repeat steps 4a-4e regarding testng dependency
5. Create page object file Base class for re-useable methods
   1. Direct back to Eclipse project
   2. Create a Base class within src/main/java package
   3. Create a File named data.properties within src/main/java package
   4. Within data.properties add the following:
      1. Browser=chrome
   5. Within Base class, create a initializeDriver method as follows:

**public** static WebDriver driver;

**public** Properties props;

**public** WebDriver initializeDriver() **throws** IOException {

props = **new** Properties();

FileInputStream fis = **new** FileInputStream("C:\\Users\\zzmar\_000\\Desktop\\IT\_Related\\AutomationTestingFrameworkBuild\\src\\main\\java\\Academy\\data.properties");

props.load(fis);

String browserName = props.getProperty("browser");

**if**(browserName.equals(“chrome")) {

System.*setProperty*("webdriver.chrome.driver", "C://chromedriver.exe");

driver = **new** ChromeDriver();

} **else** **if** (browserName.equals("firefox")) {

System.*setProperty*("webdriver.gecko.driver", "C://geckodriver.exe");

driver = **new** FirefoxDriver();

}

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

**return** driver;

}

1. Create first test case to test initializeDriver method
   1. Within src/test/java package create HomePage class and add the following:

**public** **class** HomePage **extends** Base{

@Test

**public** **void** basePageNavigation() **throws** IOException {

driver = initializeDriver();

driver.get("http://www.qaclickacademy.com/");

}

}

1. Create testing.xml file
   1. Right click project directory
   2. Hover over TestNG
   3. Click Convert to TestNG
   4. Click Finish
2. Configure TestNG with Maven using Maven Surfire
   1. Google <https://maven.apache.org/surefire/maven-surefire-plugin/examples/testng.html>
   2. Under Using Suite XML Files, copy the xml plugins tag
   3. Within project pom.xml file, create a build tag above the dependencies tag
   4. Add the copied Maven Surfire plugins tag to the build tag
   5. Now you can run Maven test
      1. In the terminal, enter the following to run the test:
         1. mvn test
3. Add Log4j with Maven
   1. Google <https://mvnrepository.com/> and search log4j
   2. Select and copy latest Log4j Core and Log4j API
   3. Add dependency xml tag to project pom.xml dependencies tag
   4. Add the following resources tag within build tag, but above plugins tag:

<resources>

<resource>

<directory>src/main/java/resources</directory>

<filtering>true</filtering>

</resource>

</resources>

1. Integrate Log4j in project
   1. Create a log4j2.xml xml file with in src/main/java resources package
   2. Add the following configuration in log4j2xml:

<Configuration status=*"WARN"*>

<Properties>

<Property name=*"basePath"*>./logs</Property>

</Properties>

<Appenders>

<RollingFile name=*"File"* fileName=*"${basePath}/prints.log"* filePattern=*"${basePath}/prints-%d{yyyy-MM-dd}.log"*>

<PatternLayout pattern=*"%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"*/>

<SizeBasedTriggeringPolicy size=*"500"*/>

</RollingFile>

<Console name=*"Console"* target=*"SYSTEM\_OUT"*>

<PatternLayout pattern=*"%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"*/>

</Console>

</Appenders>

<Loggers>

<Root level=*"trace"*>

<AppenderRef ref=*"File"*/>

</Root>

</Loggers>

</Configuration>

1. Add logs to every test class
   1. public static Logger log = LogManager.getLogger(Base.class.getName());
   2. Import the following in each class:
      1. import org.apache.logging.log4j.LogManager;
      2. import org.apache.logging.log4j.Logger;
2. Add screenshot capabilities
   1. Google <https://mvnrepository.com/> and search Apache Commons IO
   2. Copy dependency and add to projects pom.xml
   3. Within Base class, add the following getScreenshot method:

**public** **void** getScreenshot(String resultName) **throws** IOException {

File src = ((TakesScreenshot)*driver*).getScreenshotAs(OutputType.***FILE***);

FileUtils.*copyFile*(src, **new** File("C:\\Users\\zzmar\_000\\Desktop\\IT\_Related\\AutomationTestingFrameworkBuild\\testScreenshots\\screenshot-” + resultName + “.png"));

}

* 1. Create a Listeners class within src/test/java Academy package
  2. Implement ITestListener interface within Listeners class
  3. Create new Base object within Listeners class
  4. Within the onTestFailure method within Listeners class, render Base class getScreenshot() method as follows:

**public** **void** onTestFailure(ITestResult result) {

// **TODO** Auto-generated method stub

**try** {

base.getScreenshot(result.getName());

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

* 1. Add the following listeners tag within the suite tag, but above the test tag inside testing.xml:

<listeners>

<listener class-name=*"Academy.Listeners"*/>

</listeners>