Prerequisite

* Java JDK 6 and up
* JRE
* Junit and selenium library
* chrome driver/ your browser driver.

I have chrome browser in my system so has to go with chrome driver. On how to set up the selenium and eclipse please refer to setup section.

The test scripts for UI are available in https://github.com/Ssrinivasan1/API/tree/master/TestScript

## Setup for automated test with Selenium in Java

1. Download and install Java:
   1. Download Java from <http://www.java.com/en/download/manual.jsp.(> Depending on the Operating system you are working, follow the respective instructions).
   2. Create a JAVA\_HOME system environment variable, and set it to the Java installation folder (Looks like C:\Program Files\Java for windows. For unix/mac set the JAVA\_HOME in .bsh\_profile - /usr/libexec/java\_home).
   3. Add the Java runtime bin folder to the PATH (looks like C:\Program Files\Java\jre7\bin or in unix/mac looks like /usr/bin/)

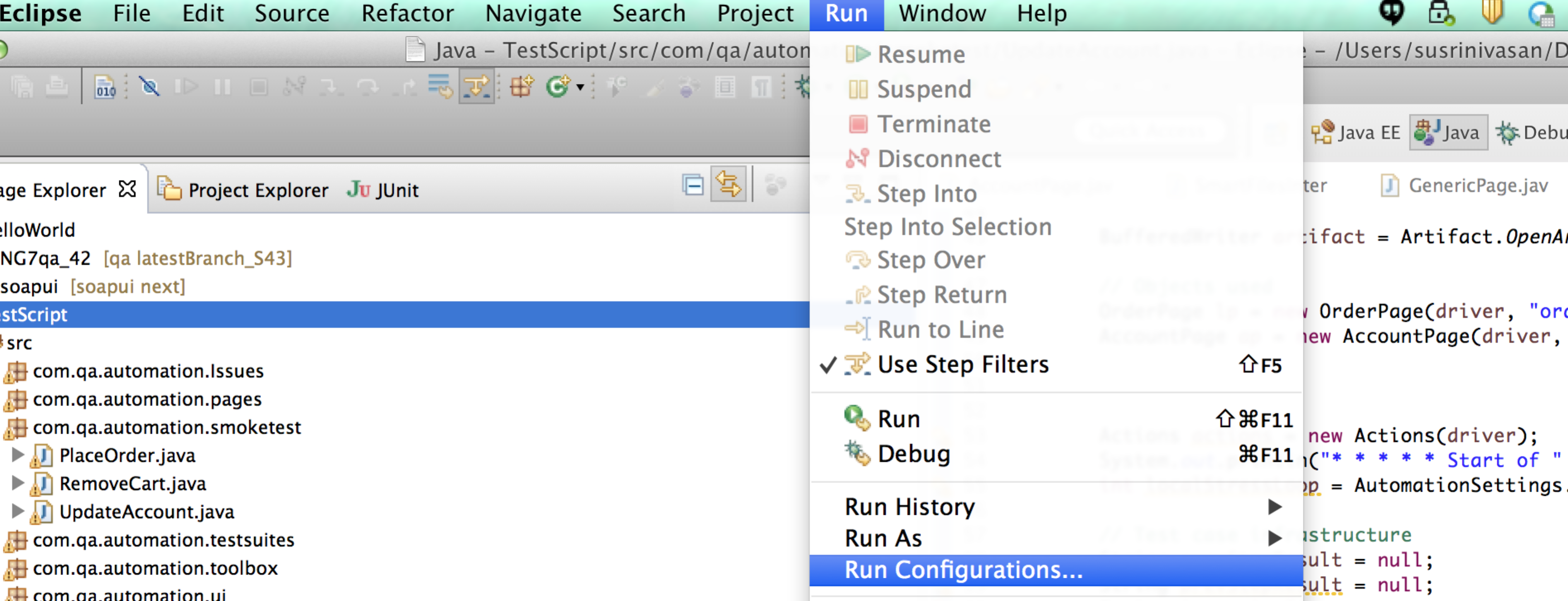
How to set path in unix/mac

* vi ~/.bash\_profile Append the following export command:
* export PATH=$PATH:/usr/local/sbin/
* Save and close the file. To apply changes immedialty enter: source $HOME/.bash\_profile

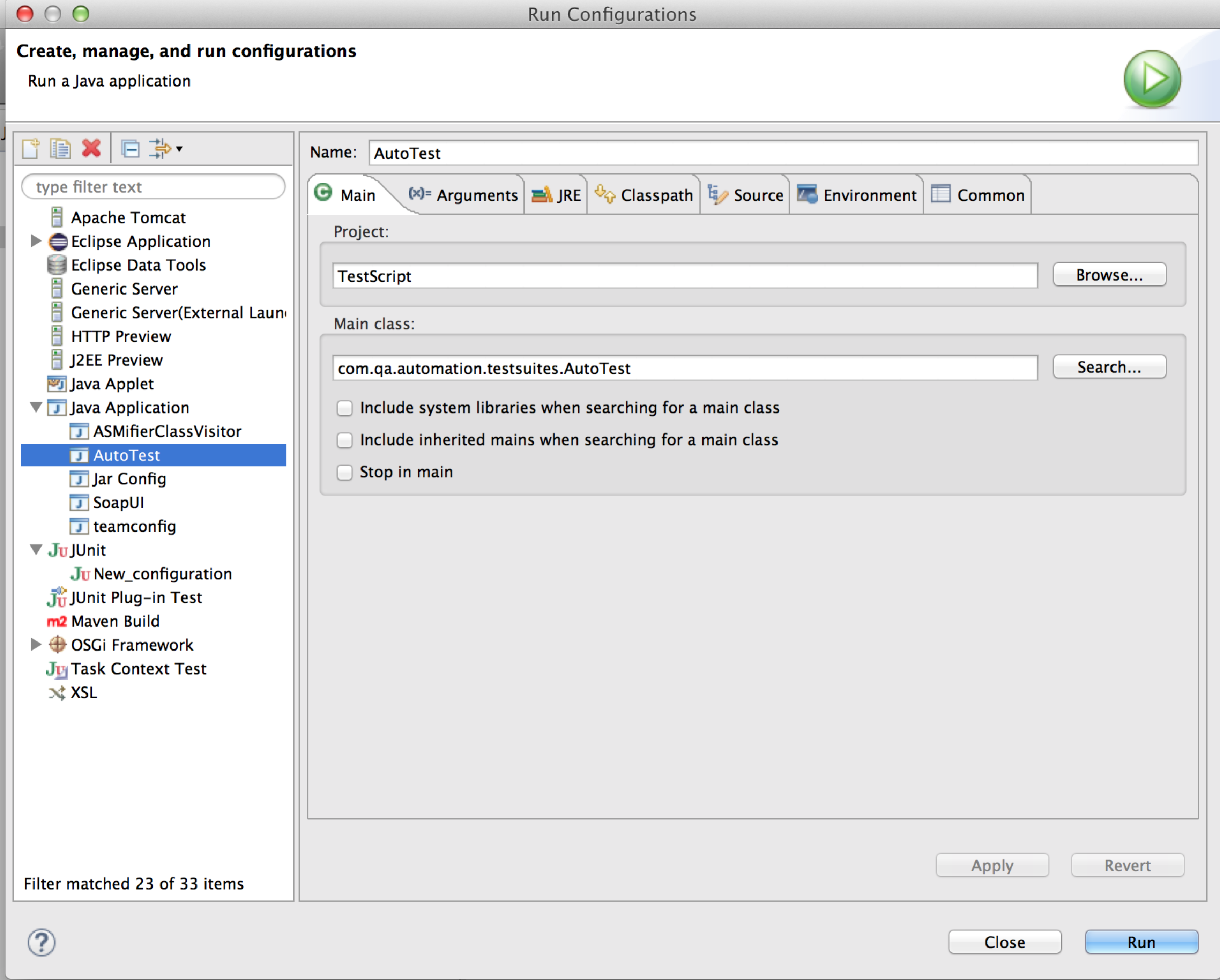
1. Download and install latest version of Eclipse IDE for Java EE Developers (e.g eclipse-jee-\*-win32-x86\_64.zip or the version depending on your Operating System) from <http://www.eclipse.org/downloads>.
2. Open Eclipse and install the e-Git plugin:
   1. "Help" | "Eclipse Marketplace..."
   2. Search for "egit"; click "Go"
   3. Click Install on package EGit - Git Team Provider
   4. Restart Eclipse
   5. Following steps 4-7 of this website to complete the installation of eGit in Eclipse: <http://crunchify.com/how-to-configure-bitbucket-git-repository-in-you-eclipse/>
3. Open Eclipse and clone the API repository () into a project.  Helpful references:

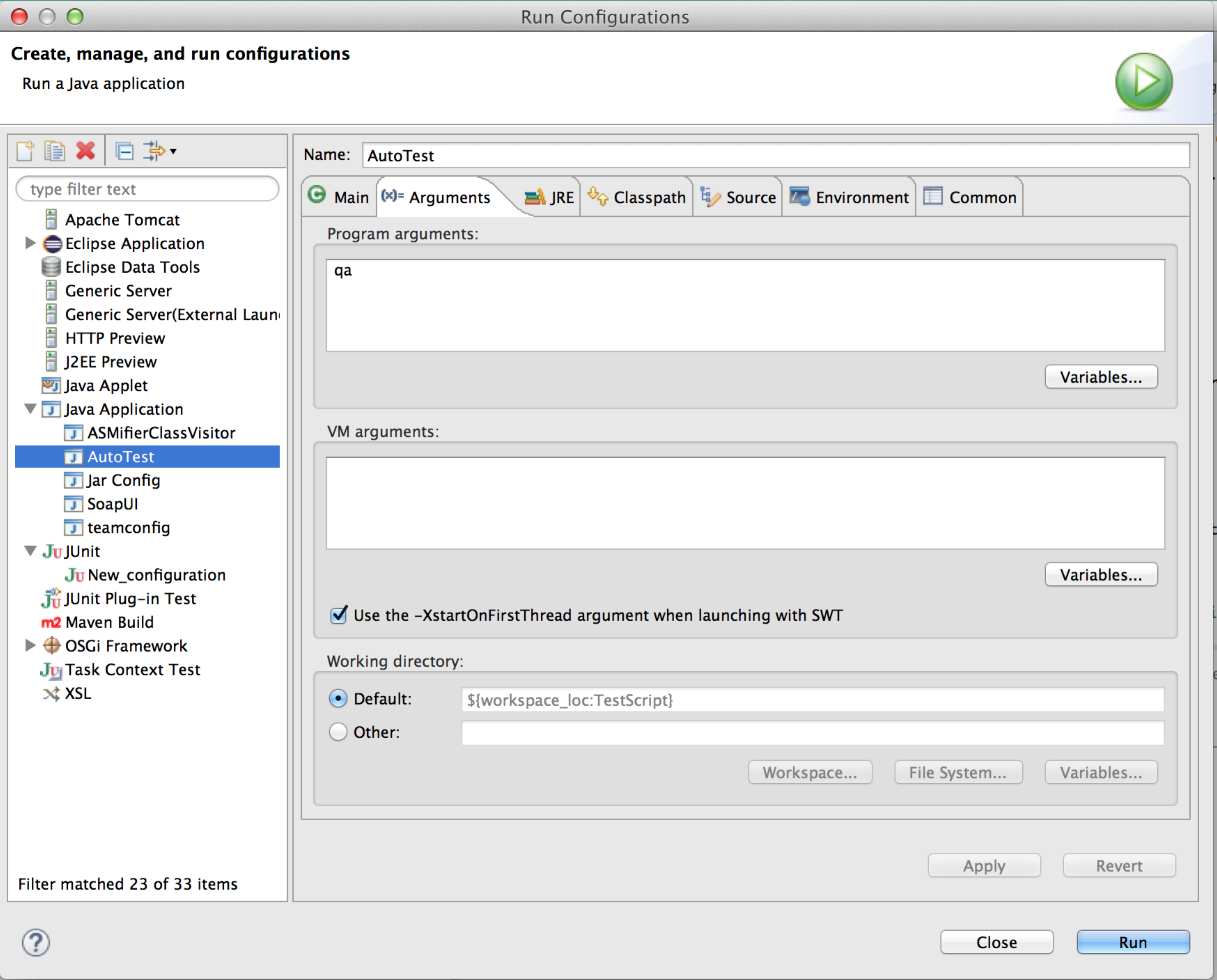
<http://crunchify.com/how-to-configure-bitbucket-git-repository-in-you-eclipse/>

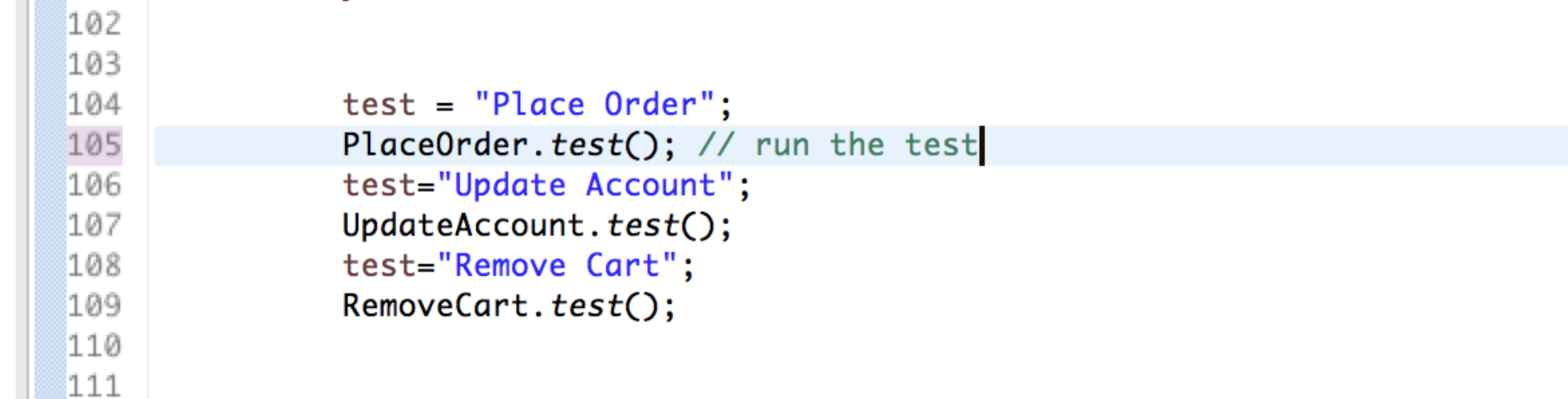
1. Create run configuration and execute tests:
2. From Eclipse, launch the Run Configurations wizard by clicking "Run | Run Configurations..."
3. Create a new run configuration by selecting the Java Application type and clicking the New button:



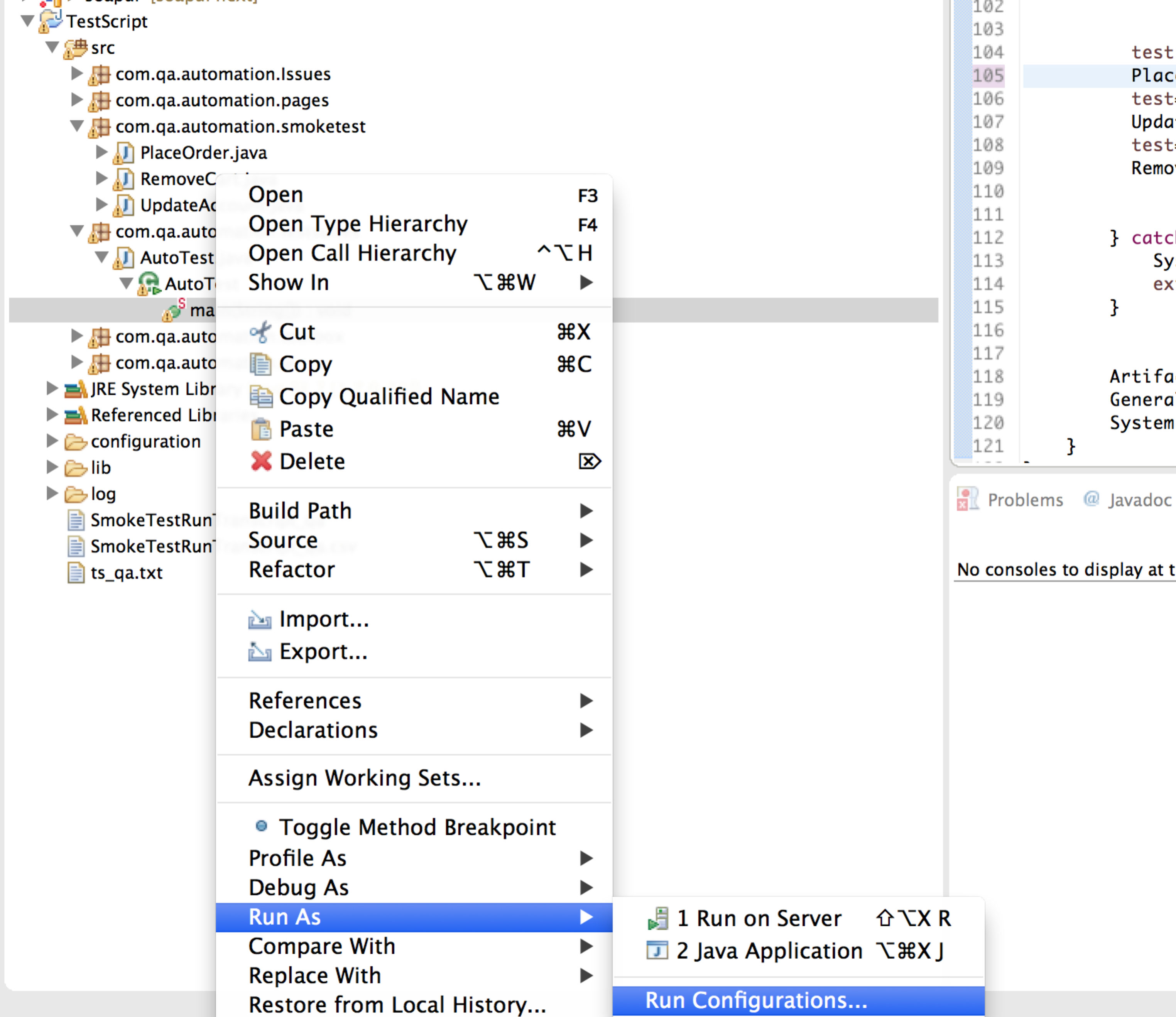
1. On the Main tab, populate the name of the configuration and browse to the local project.  Select the AutoTest.Java as your Main class.



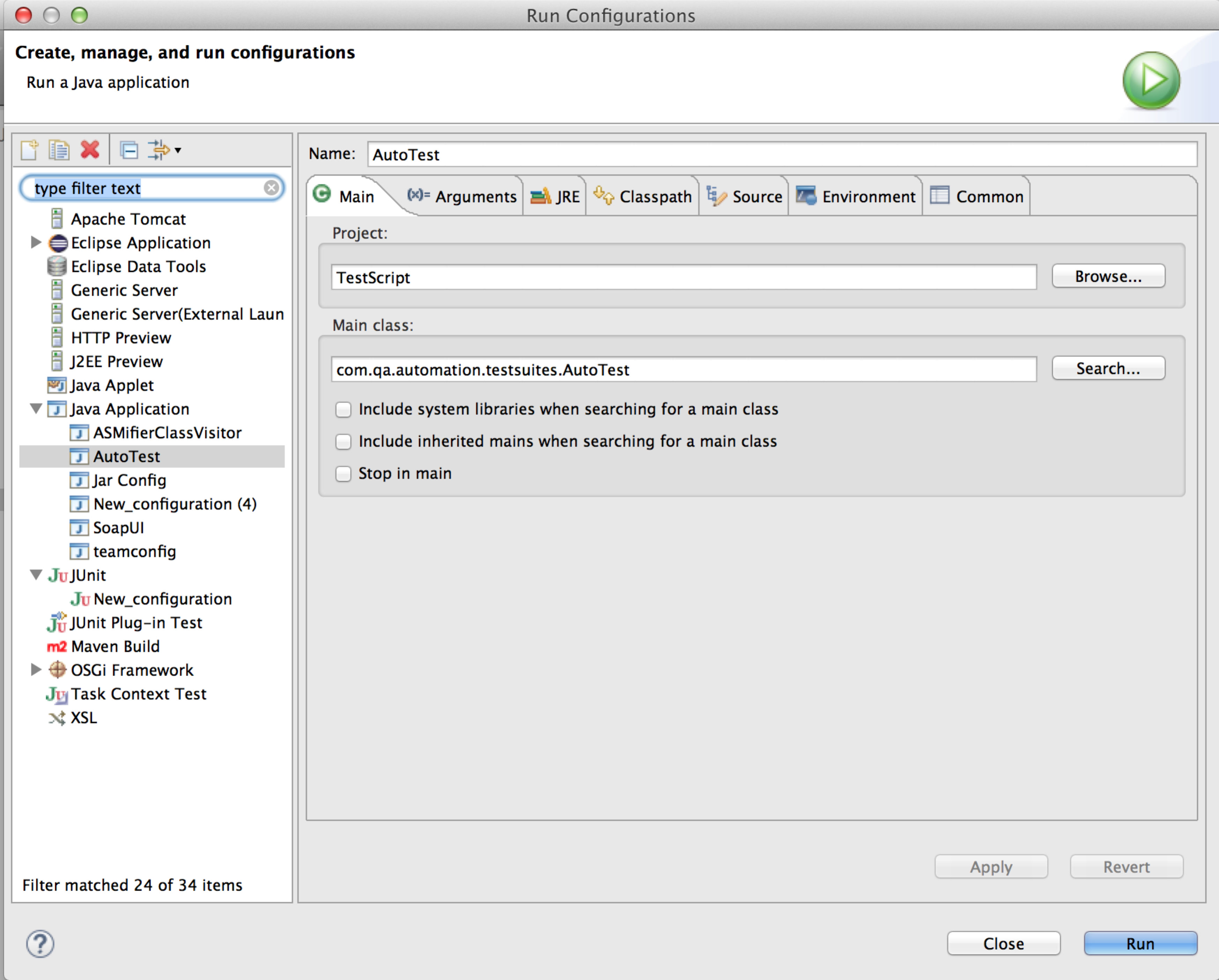
1. On the Arguments tab, select the region for e.g qa . Click "Apply" and "Close" or  choose step g when done:  
   
2. From the package/project explorer, open the AutoTest file and select the tests you wish to execute by commenting in/out of the set list (remember which set was selected in step e):



1. From the package/project explorer, open and select the AutoTest and click the Run button:



g. To run the program in an alternate way instead of choosing Close in step d choose Run.



Test execution

**Executing from Eclipse**

## Follow step 5 from Setup for automated test with Selenium in Java to execute the tests through Eclipse.

**Executing from the command line**

The AutoTest jar file can be used to execute the Test suite mode on all supported deployments.

From the windows command line:

C:\<local folder>\java -jar AutoTest.jar [ qa]

In mac or unix terminal ( Get the directory where the jar file has been saved)

cd /Users/susrinivasan/Documents/workspace/TestScript

java -jar ProductionFiles/AutoTest.jar qa

**Test execution artifact creation and analysis**

The Artifact.java class (in the toolbox package) contains methods that generate and manage test execution artifact files.  A test execution artifact can be found in two different locations:

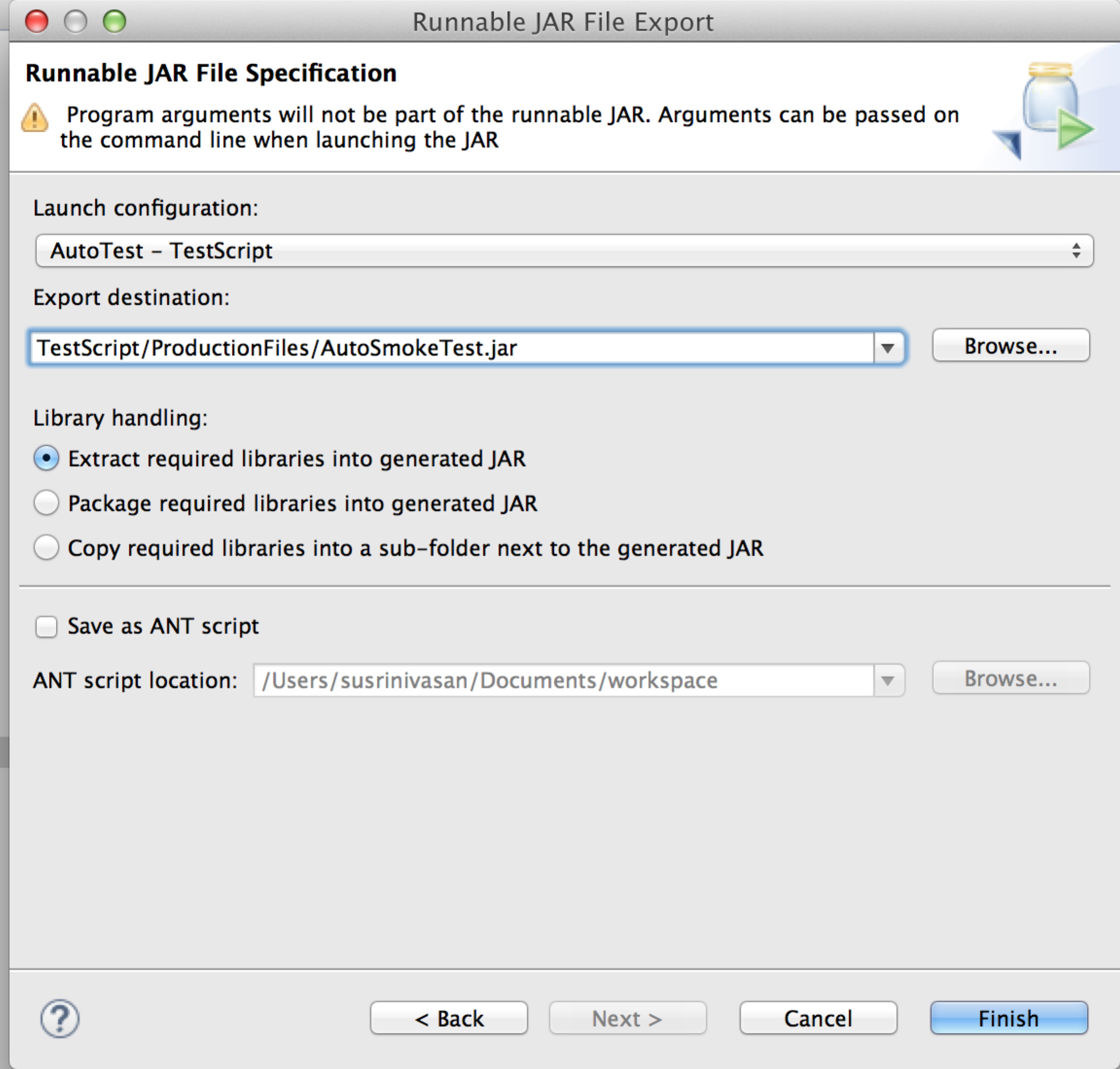
If a smoke test execution launched from the local machine, the artifact will be found locally at the directory where the automated tests are SmokeTestRunTranscript\_<region tested>.csv.

To obtain the latest test results, open the artifact and add filters to the top header line. The results from the latest execution will be displayed.

### Creating the executable jar file

Steps to create/update an executable jar file from Eclipse:

1. Regress all test code before check in with the AutoTest suite.
2. Check in all new code.
3. Click on **File | Export…**
4. Select Java | Runnable JAR file. Click **Next**.
5. In the wizard, select the created run configuration and the location for the created JAR.  The location should be gitdirectory\ProductionFiles\AutoTest.jar .
6. Select **Extract required libraries into generated JAR**.



1. Click**Finish**.
2. The generated file can be executed from NG7qa with the command java -jar ProductionFiles\AutoTest.jar <region> <set>
3. Push the created jar file into the master branch of the api project in Git.

**Guidance Notes:**

The browser driver should be placed in the directory where your automated test packages are under configuration folder.

The memory should be cleared of the previous browser runs and chrome driver. Multiple sessions if open should be closed before running the test.

**Prerequiste for Test**: Items are already in the cart for your account. If not please add items so that the account has items. The account I used is userid:’testuser1256’ password:’ jq6mBdOp829g’

##### The UI tests are developed in a Java JUnit framework, via an Eclipse IDE | Selenium 2 | ChromeDriver stack.  The developed test scripts are organized within Eclipse in the following packages of TestScript project (package names are preceded with com.qa.automation):

* **pages**
  + base classes developed for each tested page of the store demo application
  + base classes declare the web elements used in the testing and their related methods for interaction
* **smoketest**
  + test scripts that execute test steps and write out test results into an artifact
* **testsuites**
  + Collection of test scripts that are executed together
  + Each suite can execute scripts in any region (considering only the demo region as QA region run parameter for argument should be qa)
  + AutoTest.java executes the scripts in any region (considering only the demo region as QA region run parameter for argument should be qa)
* **toolbox**: base classes of general methods