# Developer Setup

This document will detail steps for the setting up of a development environment form a generic perspective to a MuleSoft specific perspective. The generic tools selected for discussion have a wide acceptance throughout the development community plus the MuleSoft tools that are part of the MuleSoft development philosophy.

The following tools products will be discussed:

* Eclipse Integrated Development Environment
* MuleSoft AnyPoint Studio version build on the Eclipse platform.
* Maven
* Git and GitHub.

# Generic Developer Setup

## Maven

This section of the document is intended to give a high-level overview of Maven the tool. This document’s audience can potentially consist of developers well versed in the use of Maven or developers involved in a first-time migration to Maven.

The ***Apache Maven Getting Started Guide*** web site is pointed to by this URL:

<http://maven.apache.org/guides/getting-started/>

This an informative source of all things Maven

### What is Maven

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

The following are excerpts from the ***Philosophy of Maven*** section of the ***Apache Maven Getting Started Guide*** web site:

*Maven is generally considered by many to be a build tool. Many people who come to Maven initially are familiar with Ant so it’s a natural association but Maven is not just a build tool, and not just a replacement for Ant. Maven is an entirely different creature from Ant. Ant is simply a toolbox whereas Maven is about the application of patterns in order to achieve an infrastructure which displays the characteristics of visibility, reusability, maintainability, and comprehensibility.*

*Maven was born of the very practical desire to make several projects at Apache work in the same way. So that developers could freely move between these projects, knowing clearly how they all worked by understanding how one of them worked. If a developer spent time understanding how one project built it was intended that they would not have to go through this process again when they moved on to the next project.*

Summarizing the above excerpts, Maven is a build tool that emphasizes the concept of reusable patterns that direct developers toward a build philosophy that is consistent across all Maven projects. Where ANT was a free form XML based tools with some guidelines, there were no consistent implementation across projects. Maven, on the other hand is built on the premise that the build infrastructure “plumbing” and philosophies will be consistent across applications.

### Very High-Level Maven Overview

As has been stated at the beginning of this document, this is a high-level overview of Maven. So, instead of detailing the process and procedures of building a project with Maven, this document will focus on file artifacts that are needed and facilitate the Maven build process. The artifacts that will be discussed briefly are:

* .m2 Repository directory
* Pom.xml
* Settings.xml
* Maven Archetypes

Again, for a full understanding of the Maven product refer to the ***Apache Maven Getting Started Guide*** web site

#### .m2 Repository Directory

The maven local repository is a local folder that is used to store a developers’ s project’s dependencies (plugin jars and other files which are downloaded by Maven). When you build a Maven project, all dependency files will be stored in your Maven local repository. For example, when the ***mvn clean install***

command line interface command is executed

The contents of .m2 are directory are:

* A ***settings.xml*** file that contains global settings for all Maven executions.
* A folder called ***repository*** that holds all the local copies of various maven artifacts, either caches of artifacts pulled down from remote repositories, such as Maven Central, or artifacts built by a developer’s local maven builds.

The .m2 directory is typically located at *USER\_HOME*/.m2

#### Pom.xml

The pom.xml contains the Project Object Model (POM) for this project. The POM is the basic unit of work in Maven. This is important to remember because Maven is inherently project-centric in that everything revolves around the notion of a project. In short, the POM contains every important piece of information about a project and is essentially one-stop-shopping for finding anything related to the project.

##### Dependency Resolution

The cornerstone of the POM is its dependency list. Most projects depend upon others to build and run correctly. Maven downloads and links the dependencies for a project compilation and other goals that require them. As a bonus, Maven brings in the dependencies of those dependencies (transitive dependencies), allowing the project list to focus solely on the dependencies the project requires.

By default, pom.xml is at root path of the project but can be overridden by the Maven command line -f flag (mvn … -f file<arg>).

A detailed description of the pom.xml is pointed to by this URL:

<http://maven.apache.org/pom.html>

The following figure diagram shows the elements defined in a pom.xml

Figure 1 pom.xml

<?xml version="1.0" encoding="UTF-8"?>

project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<!-- The Basics -->

<groupId>...</groupId>

<artifactId>...</artifactId>

<version>...</version>

<packaging>...</packaging>

<dependencies>...</dependencies>

<parent>...</parent>

<dependencyManagement>...</dependencyManagement>

<modules>...</modules>

<properties>...</properties>

<!-- Build Settings -->

<build>...</build>

<reporting>...</reporting>

<!-- More Project Information -->

<name>...</name>

<description>...</description>

<url>...</url>

<inceptionYear>...</inceptionYear>

<licenses>...</licenses>

<organization>...</organization>

<developers>...</developers>

<contributors>...</contributors>

<!-- Environment Settings -->

<issueManagement>...</issueManagement>

<ciManagement>...</ciManagement>

<mailingLists>...</mailingLists>

<scm>...</scm>

<prerequisites>...</prerequisites>

<repositories>...</repositories>

<pluginRepositories>...</pluginRepositories>

<distributionManagement>...</distributionManagement>

<profiles>...</profiles>

</project>

#### Settings.xml

The settings element in the settings.xml file contains elements used to define values which configure Maven execution in various ways, like the pom.xml, but should not be bundled to any specific project (i.e. it is Maven specific).

The settings.xml can be specified at two levels:

1. User Level. This settings.xml file provides configuration for a single user
   * It is typically located at normally provided in *USER\_HOME*/.m2/settings.xml.
2. Global Level. This settings.xml file provides configuration for all Maven users
   * All users on a server using the same version of Maven
   * It is located at *"MVN\_HOME*/conf/settings.xml.

If both files exist, their contents are merged, with the user-level settings.xml being dominant.

A detailed description of the settings.xml is pointed to by this URL:

<https://maven.apache.org/ref/3.5.4/maven-settings/settings.html>

The following figure diagram shows the elements defined in a setings.xml. Lower level elements have been deleted for clarity.

Figure 2 setings.xml

<?xml version="1.0" encoding="UTF-8"?>

<settings xmlns="http://maven.apache.org/SETTINGS/1.1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.1.0 http://maven.apache.org/xsd/settings-1.1.0.xsd">

<localRepository>

</localRepository>

<interactiveMode>

</interactiveMode>

<usePluginRegistry>

</usePluginRegistry>

<offline>

</offline>

<proxies>

<proxy>

</proxy>

</proxies>

<servers>

<server>

</server>

</servers>

<mirrors>

<mirror>

</mirror>

</mirrors>

<profiles>

<profile>

<activation>

</activation>

<properties>

</properties>

<repositories>

<repository>

</repository>

</repositories>

<pluginRepositories>

<pluginRepository>

</pluginRepository>

</pluginRepositories>

<id/>

</profile>

</profiles>

<activeProfiles>

</activeProfiles>

<pluginGroups>

</pluginGroups>

</settings>

#### Maven Archetypes

A Maven Archetype is a Maven plugin that provides the facility the create a Maven Project Structure whose specifics are defined by the specific archetype utilized.

The archetype will create the project artifacts consisting of a java application project consisting of a standardized folder structure plus a pom.xml file.

To initiate the archetype process, execute the following Maven command line:

***mvn archetype:generate***

The ***mvn*** process will interactively prompt the user a number of question which will include for example the required archetype to use.

The following URL give a detail archetype introduction:

<https://maven.apache.org/guides/mini/guide-creating-archetypes.html>

Additionally the following URL provides a good overview of the creation of the Quick start Maven-Archetype the will create a simple java application Maven project structure including the prompted archetype question alluded to above: -

<https://www.tutorialspoint.com/maven/maven_project_templates.htm>

## Eclipse Git Plugin (Egit)

The Configuration Management Playbook document contains an overview of the GitHub product so that detailing will not be repeated in this document. This section will focus on the Eclipse Git Plugin (EGit) installation into Eclipse, although there is substantial Eclipse documentation that discusses all things EGit. The appropriate documentation links are provided below.

### EGit for Eclipse Users Web Site

A developer new to Git or distributed version control systems generally, might want to initially examine the ***EGit for Eclipse Users*** web site. The site is pointed to by this URL:

<http://wiki.eclipse.org/EGit/Git_For_Eclipse_Users>

The site is aimed at developers that are Eclipse literate but probably have been using either the integrated CVS or external SVN providers to store application source code. The content of the website is about Git, what it means to a developer as an Eclipse user, and specifically, how it affects how a developer obtains or works with projects from Eclipse.org.

### EGit Users Guide Web Site

The EGit Users Guide Web Site is the complete source of information for using Egit in an Eclipse environment. The site is pointed to by this URL:

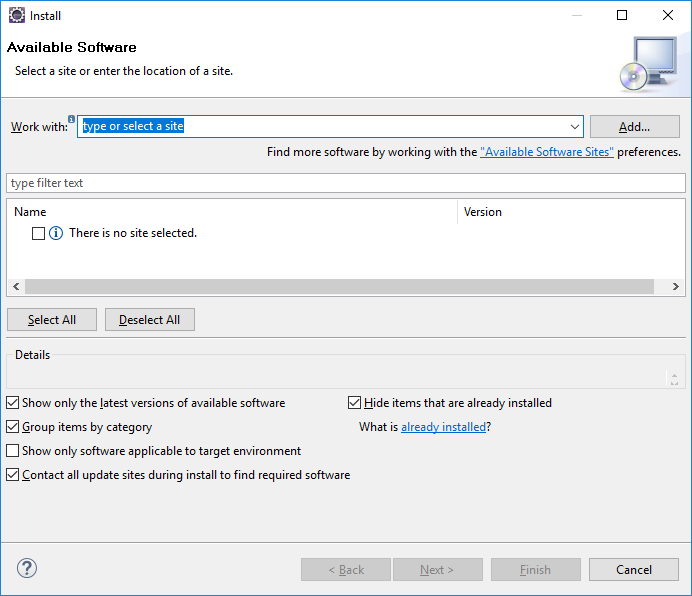
[http://wiki.eclipse.org/EGit/User\_Guide#Getting\_Started](http://wiki.eclipse.org/EGit/User_Guide%23Getting_Started)

The site contains extensive documentation spanning repository integration, code branching and merging etc. It is the go-to site for EGIT in Eclipse process and procedures.

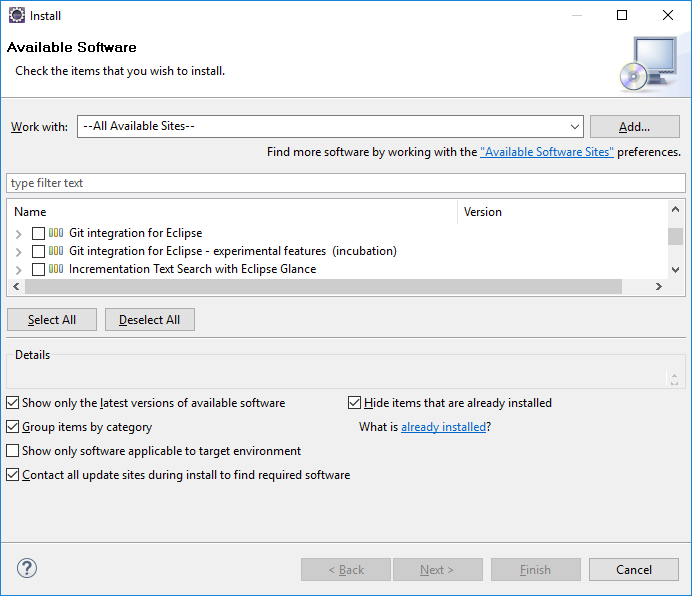
### Installing EGit into Eclipse

The version of eclipse used in this document s Mars.1 Release (4.5.1) which comes installed with EGit. The installation procedure was detailed nonetheless if for no other reason than to install any pending EGit updates.

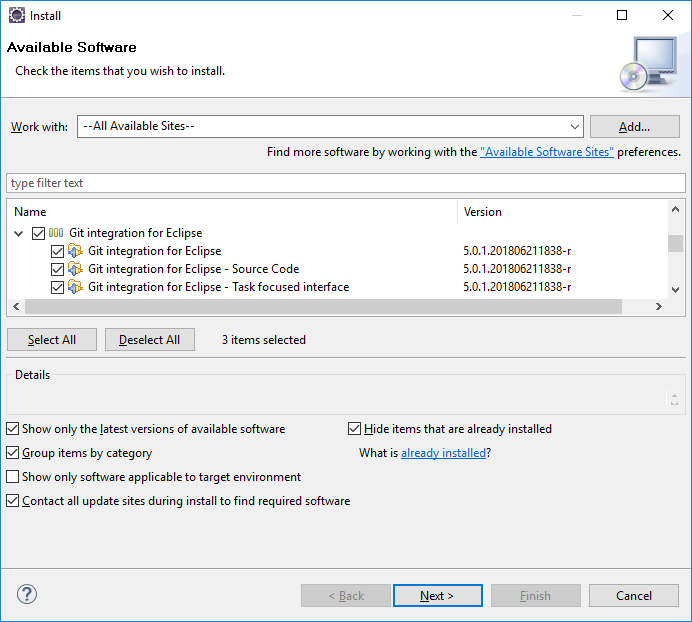
Click on ***Help*** the click on ***Install New Software***. The following screen appears:



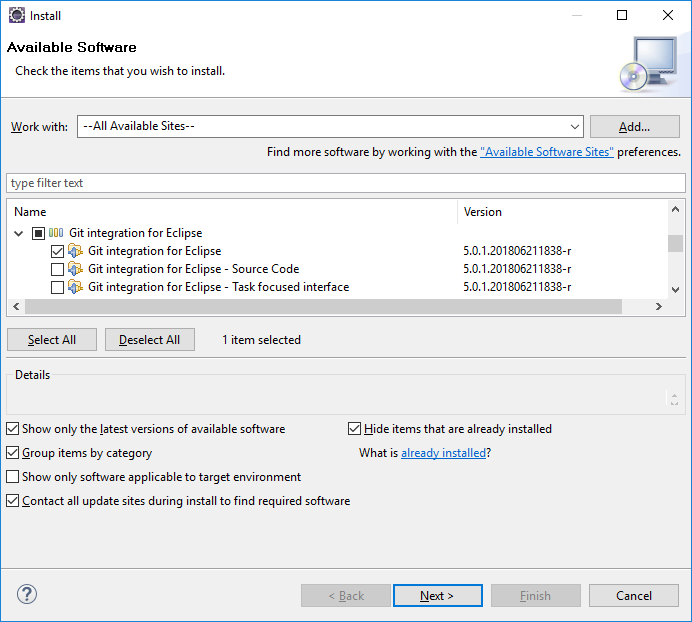
In the ***Work with*** dialog box select **--All Available Sites--**. In the ***Name*** dialog box, a list of applications will appear. Scroll down to ***Git Integration for Eclipse***.



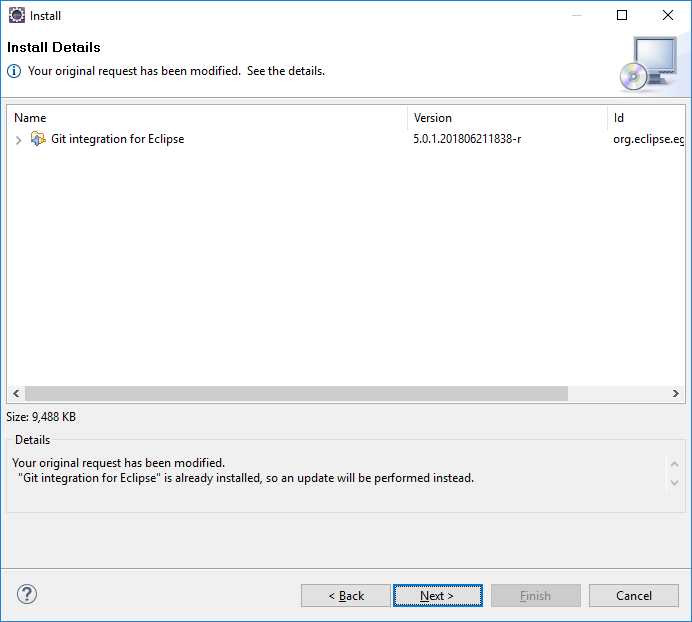
Click on the ***Git Integration for Eclipse Check Box***



Unclick all ***Check Boxes*** except ***Git Integration for Eclipse***.

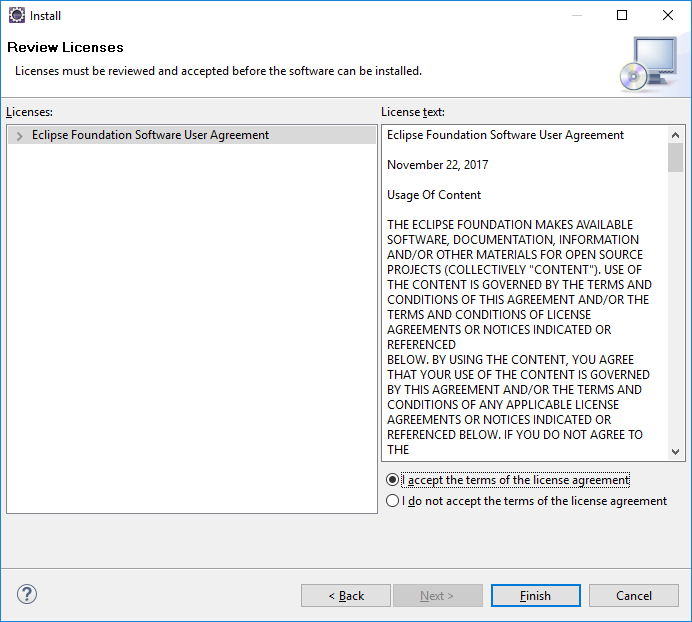


Click on the ***Next*** button. The following screen appears:

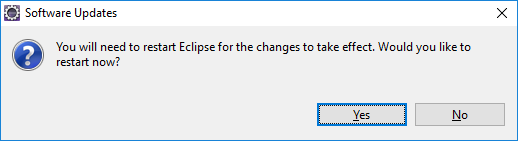


As stated previously Git was already installed on this instance of Eclipse, so updates will be performed.

Click on the **Next** button. The following screen will appear. Click on the ***I accept the terms of the license agreement radio button*** then click ***Finish***.



After installation the following dialog box will appear:



## Unit Testing Frameworks

### JUnit

JUnit is a unit testing framework for the Java programming language. The section of the document will detail the creation of a **very simple** HelloWorld JUnit test class using the Mars.1 Release (4.5.1) Eclipse IDE which come bundled with JUnit.Only the creation and running of the HelloWorld JUnit test class will be detailed in this section.

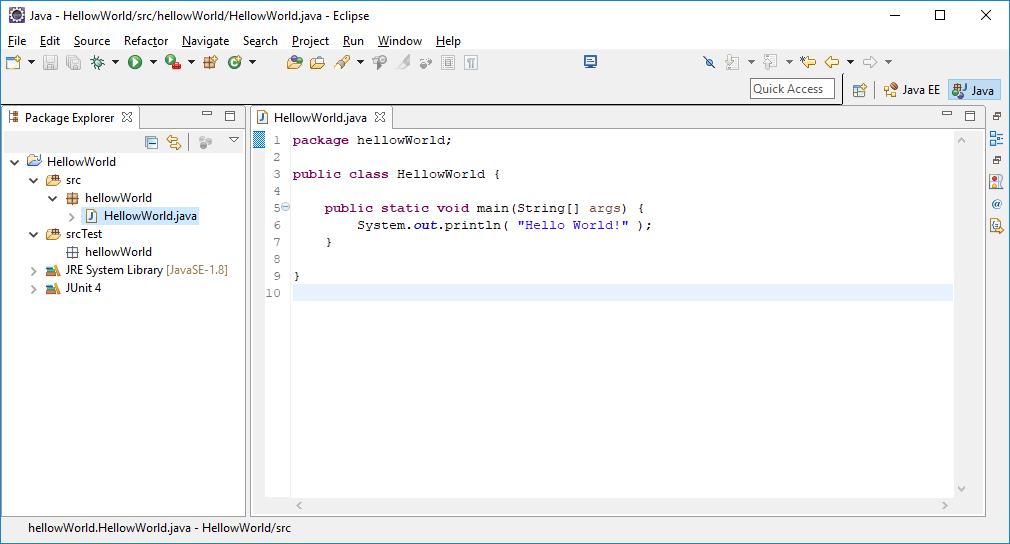
A good reference guide for the creation of more sophisticated JUnit test cases can be found in the following IBM Developer Works Tutorial and is pointed to by this UTL:

<https://www.ibm.com/developerworks/java/tutorials/j-junit4/index.html>

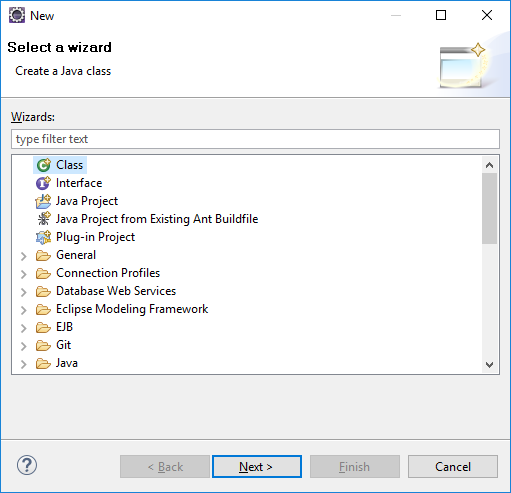
#### JUnit Test Class Creation

Using Eclipse, a HelloWorld java class will be created followed by the creation of a corresponding HelloWorldTest java class. The creation of the of the Eclipse Project, the HelloWorld java class and the srcTest folder will not be described, only the creation and processing of the HelloWorldTest java class.

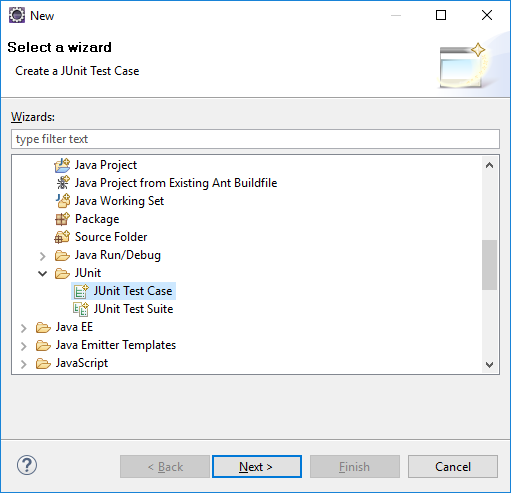
Eclipse after creation of the Eclipse Project, the HelloWorld java class and the srcTest folder:



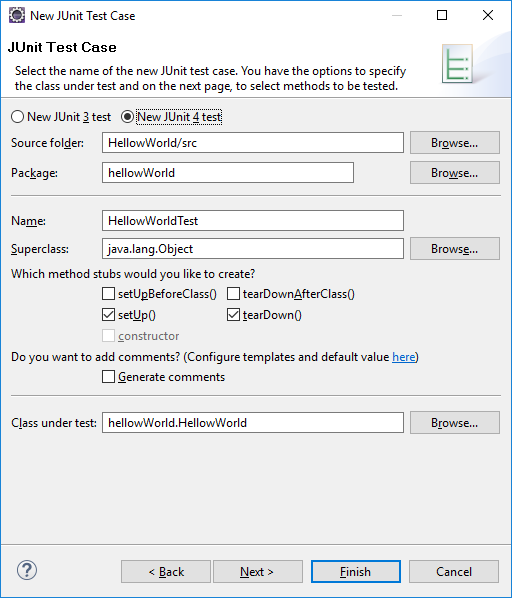
Highlight the ***hellowWorld*** class as shown above, Click on ***File***, ***New*** and ***Other*** and the following screen appears:



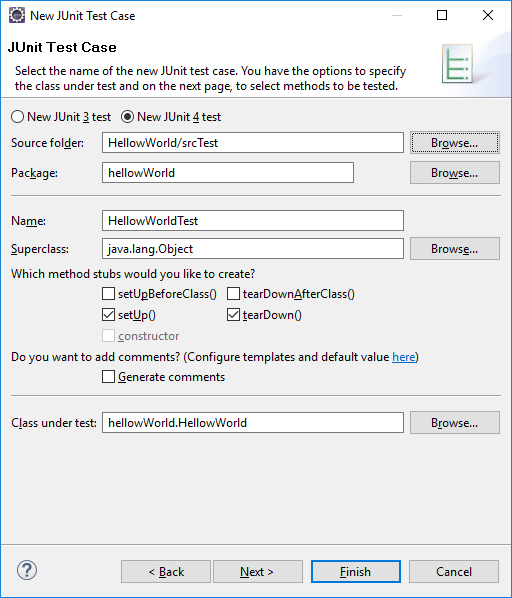
Scroll down to Java then Click on ***JUnit*** then select ***JUnit Test Case***:



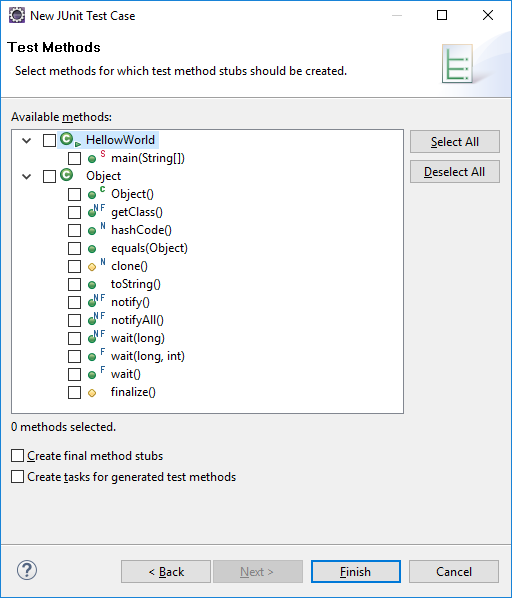
Select ***Next*** and the following default JUnit Test Case screen appears:



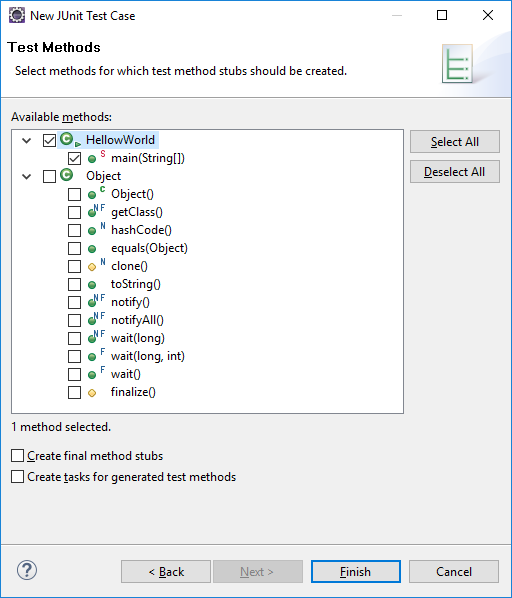
The only change is to the ***SourceFolder*** value which is changed to srcTest:



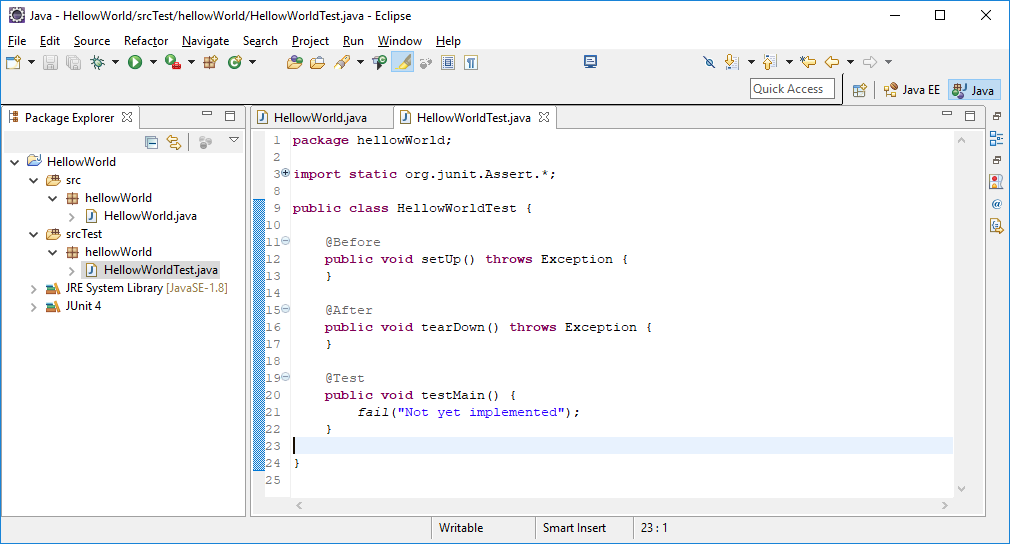
Click **Next**and the following screen appears:



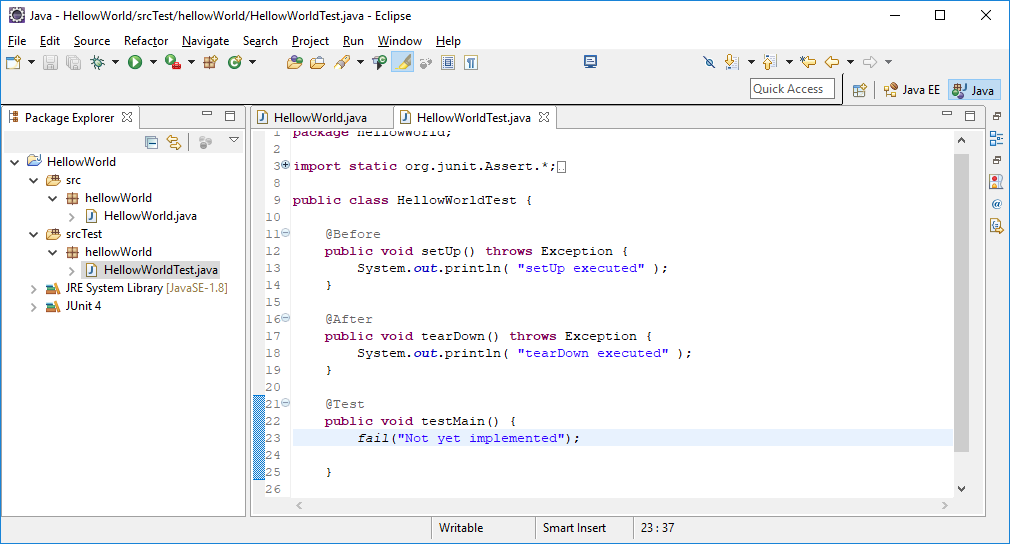
Select the ***main (String[])*** checkbox



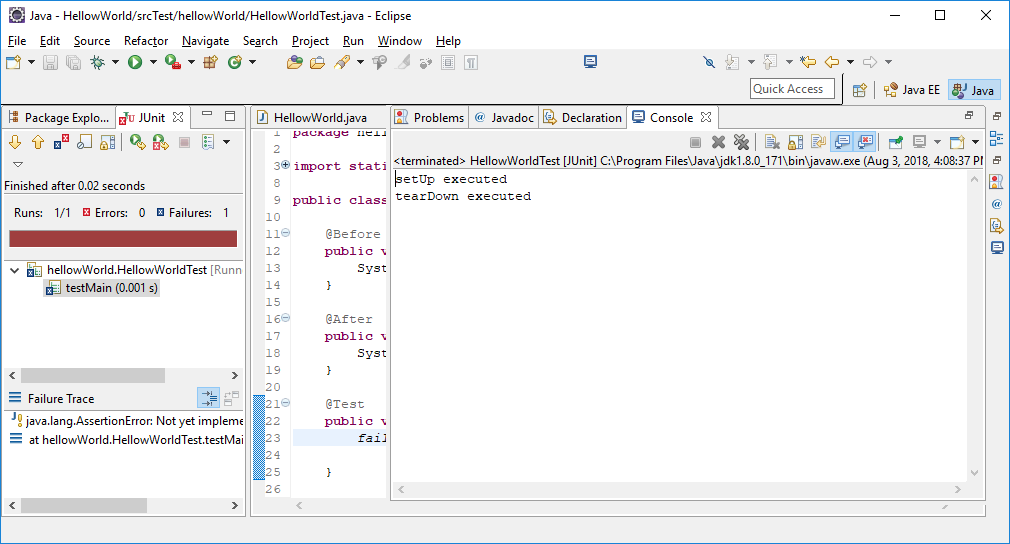
Click ***Finish*** and the HellowWorldTest class is generated



*Syetem.out.println* lines were added to the setup and tearDown methods. The testMain method has the default *org.junit.Assert* *fail* static value which will cause the HellowWorldTest to fail.



Highlight the hellowWorldTest class as shown above, Right mouse click on hellowWorldTest class then select ***Run AS*** then select ***2 JUnit Test*** then the following screen appears:



Notice that the **setUp executed** and **tearDown executed** messages were displayed and that the test failed as expected.

# MuleSoft Developer Setup

## Maven with MuleSoft

Maven is a project management utility that you can use to develop projects in MuleSoft.

Refer to section 1.1 Maven for a more a generic (non-MuleSoft) description of Maven.

The following sections are a summation of the numerous Maven web pages with MuleSoft documentation

### Maven Reference

These pages summarize reference information that is helpful for working with Maven and Mule and is pointed to by this URL:

<https://docs.mulesoft.com/mule-user-guide/v/3.9/maven-reference>

Topics include:

* Populating the Maven Repositories from the Command Line
* Deploying Mule Artifacts to a Central Maven Repository
* Referencing MuleSoft’s Enterprise Repositories
* Checking JARs Included in a Project
* Files
* Useful Commands
* External Resources

### Using Maven with MuleSoft (High Level)

These pages are a high-level Maven/MuleSoft description and is pointed to by this URL:

<https://docs.mulesoft.com/mule-user-guide/v/3.9/using-maven-with-mule>

Topics include:

* Prerequisites
* Benefits
* Options for Using Maven with Mule
* Tips

### Using Maven in Mule

These pages detail how to use Maven outside of AnyPoint Studio, working directly with Mule. The Mule plugins helps to develop Mule applications within Maven, and to control Mule instances from within a Maven environment description and is pointed to by this URL:

<https://docs.mulesoft.com/mule-user-guide/v/3.9/using-maven-in-mule-esb>

Sub Links include:

* Maven Tools for Mule
* Mule Plugin for Maven
* Configuring Maven to Work with Mule

These sub links are detailed below:

#### Maven Tools for Mule

Maven Tools for Mule is a plugin for Maven that allows you to develop Mule applications using Maven tooling. The kit provided by the plugin includes archetypes for building regular Mule applications, Mule domains, and Mule domain bundles and is pointed to by this URL:

<https://docs.mulesoft.com/mule-user-guide/v/3.9/maven-tools-for-mule-esb>

Topics include:

* Assumptions
* Adding the Plugin to Maven
* Creating a Mule Application
* Creating a Mule Domain
* Creating a Complete Mule Domain Project
* Available Parameters

#### Mule Plugin for Maven

These pages detail how the mule-maven-plugin allows the deployment of Mule applications to different kinds of servers: Standalone (both Community and Enterprise), clustered, Anypoint Runtime Manager and CloudHub. It is part of the framework for developing Mule applications with Maven. The most important capabilities provided by the plugin are running integration tests, and deploying applications to different environments and is pointed to by this URL:

<https://docs.mulesoft.com/mule-user-guide/v/3.9/mule-maven-plugin>

Topics include:

* Prerequisites
* Adding the Plugin
* A Simple Example
* Deploying to Anypoint Runtime Manager
* Deploying to CloudHub
* Using a Mule Server Instead of Downloading Mule Dependency
* Deploying to a Mule Server Using the Agent
* Running Integration Tests
* Full Example
* Deploying to a Local Mule Cluster
* Deploying Multiple Applications
* Skipping Plugin Execution
* Anypoint Runtime Manager On Premises TLS Errors
* Full List of Parameters

#### Configuring Maven to Work with Mule

These pages detail when working with Maven to build Mule applications outside of Anypoint Studio, there is a need to configure the Maven installation to work successfully with Mule. This page covers how to maintain the POM file to add or adjust dependencies and point to the correct MuleSoft repositories, and how to modify the Maven settings.xml file to include credentials and profiles for Enterprise repositories and is pointed to by this URL:

<https://docs.mulesoft.com/mule-user-guide/v/3.9/configuring-maven-to-work-with-mule-esb>

Topics include:

* Setup Overview
* Configuring Your Maven Installation for Mule

### Using Maven in Anypoint Studio

These pages describe Maven integration with Anypoint Studio, including building applications, importing Maven projects or "mavenizing" existing projects description and is pointed to by this URL:

<https://docs.mulesoft.com/anypoint-studio/v/6/using-maven-in-anypoint-studio>

Sub Links include:

* Maven Support in Anypoint Studio
* Building a Mule Application with Maven in Studio
* Enabling Maven Support for a Studio Project
* Importing a Maven Project into Studio

These sub links are detailed below:

#### Maven Support in Anypoint Studio

These pages describe how to enable or disable Maven support for your projects in Anypoint Studio and is pointed to by this URL:

<https://docs.mulesoft.com/anypoint-studio/v/6/maven-support-in-anypoint-studio>

Topics include:

* Setting Your Maven Preferences
* Disabling Maven Support in a Single Studio Project

#### Building a Mule Application with Maven in Studio

These pages walk one through creating a Mavenized Mule Application project from scratch. It covers providing the project with unit tests, maintaining its POM file, then building and deploying the project and is pointed to by this URL:

<https://docs.mulesoft.com/anypoint-studio/v/6/building-a-mule-application-with-maven-in-studio>

Topics include:

* Prerequisites
* Creating a Maven Project
* Build a Simple Project
* Adding a Unit Test
* Running the Application with Maven
* Managing Dependencies
* Troubleshooting

#### Enabling Maven Support for a Studio Project

These pages take a Mule project that was already created in Anypoint Studio (i.e. the "mavenization" of a Mule project) and add Maven functionality to it. Maven can make a project more easily shareable with other developers as it takes care of its dependencies and helps automate the project’s build process project and is pointed to by this URL:

<https://docs.mulesoft.com/anypoint-studio/v/6/enabling-maven-support-for-a-studio-project>

Topics include:

* Mavenizing an Existing Mule Project
* Running a Mavenized Project
* Running Maven Life-Cycle Commands

#### Importing a Maven Project into Studio

These pages describe a Mule project that was built using Maven outside Anypoint Studio, you can still import it into Studio and continue to work on it with the benefits of the Studio IDE and is pointed to by this URL:

<https://docs.mulesoft.com/anypoint-studio/v/6/importing-a-maven-project-into-studio>

Topics include:

* Prerequisites
* Importing a Maven Project into Studio
* Troubleshooting

## GitHub Plugin

Refer to section 2.2Eclipse Git Plugin (Egit) for a more a generic (non-MuleSoft) description of Git.

This page provides a description Using Maven with GIT and is pointed to by this URL:

<https://docs.archive.mulesoft.com/mule-user-guide/v/3.3/using-git-with-studio>

This page provides a suite of Mule example materials to help you learn more about building applications in Anypoint Studio and Mule ESB Standalone and is pointed to by this URL:

<https://github.com/mulesoft/mulesoft-docs/blob/master/mule-user-guide/v/3.5/mule-examples.adoc>

## Unit Testing Frameworks

### Munit Setup

MUnit is a Mule application testing framework that allows you to easily build automated tests for your integrations and APIs. It provides a full suite of integration and unit test capabilities and is fully integrated with Maven and Surefire for integration with your continuous deployment environment. MUnit is fully integrated with Anypoint Studio, allowing you to create, design and run MUnit tests just like you would Mule applications.

With MUnit you can:

* Create your test by writing Mule code
* Mock processors
* Spy any processor
* Verify processor calls
* Enable or ignore particular tests
* Tag tests
* Check visual coverage in Studio
* Generate coverage reports

MUnit is divided in two main sub modules:

* MUnit
* MUnit Tools

A detailed description Using Maven with MUnit is pointed to by this URL:

<https://docs.mulesoft.com/munit/v/2.1/>

#### MUnit Event Processors

MUnit has a set of dedicated Event Processors aimed to validate, assert and mock the content and status of the Mule event so you can run automatic tests against your Mule application.

* Assertion Event Processor
* Set Event Processor
* About Mock When Event Processor
* Verify Event Processor
* Spy Event Processor

A description Using Maven with MUnit Event Processors are pointed to by this URL:

<https://docs.mulesoft.com/munit/v/2.1/message-processors>