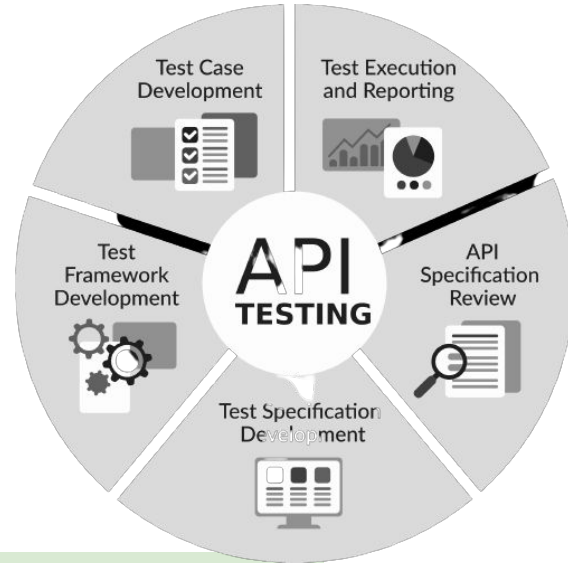


# Maven Mastery



Pramod Dutta  
Lead SDET.



## Maven Overview for API Automation

# Agenda

- Why Maven ?
- Learning Maven Project
- Maven Lifecycle & Phases , Plugins
- Creating Maven Project and Undersing it
- Important Interview QnA on Maven



# Why use MAVEN?

- Maven is chiefly used for Java-based projects, helping to download dependencies, which refers to the libraries or JAR files.
- 
- Maven is a powerful project management tool that is based on POM (project object model).

401  
Response

<https://stackoverflow.com/questions/3589562/why-maven-what-are-the-benefits>



# Creating a Maven Project and TestNG Dependency



# What is Maven?

- Maven is a build automation tool used primarily for Java projects.
- The Maven project is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project.
- It is used for projects build, dependency and documentation



# Install Maven

- If you are using IntelliJ, Maven is installed by Default.
- Else <https://maven.apache.org/download.cgi>
- Verify the version of the Maven
  - `mvn --version`

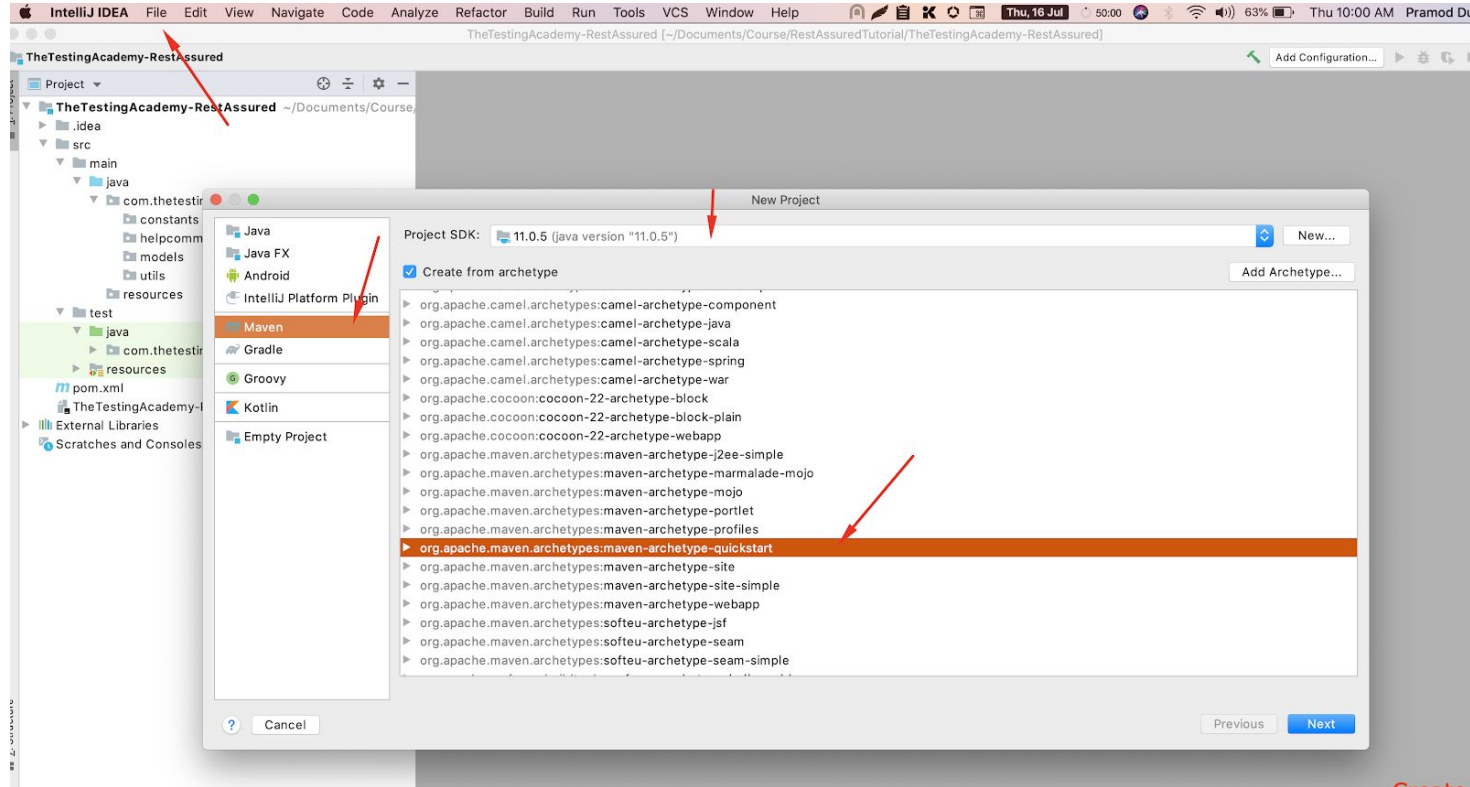


# Create a Maven Project using CMD

- mvn archetype:generate
  - DgroupId=com.mycompany.app
  - DartifactId=my-app
  - DarchetypeArtifactId=maven-archetype-quickstart
  - DarchetypeVersion=1.4
  - DinteractiveMode=false



# Creating a Maven using IntelliJ







# Creating a Maven using IntelliJ

The screenshot displays the IntelliJ IDEA interface with a Maven project named 'MavenDemo' created using an archetype. The Project view on the left shows the directory structure: MavenDemo > src > main > java > com > thetestingacademy > ra. The pom.xml file is open in the editor, showing the Maven project configuration. The Run console at the bottom shows the successful execution of the Maven archetype plugin.

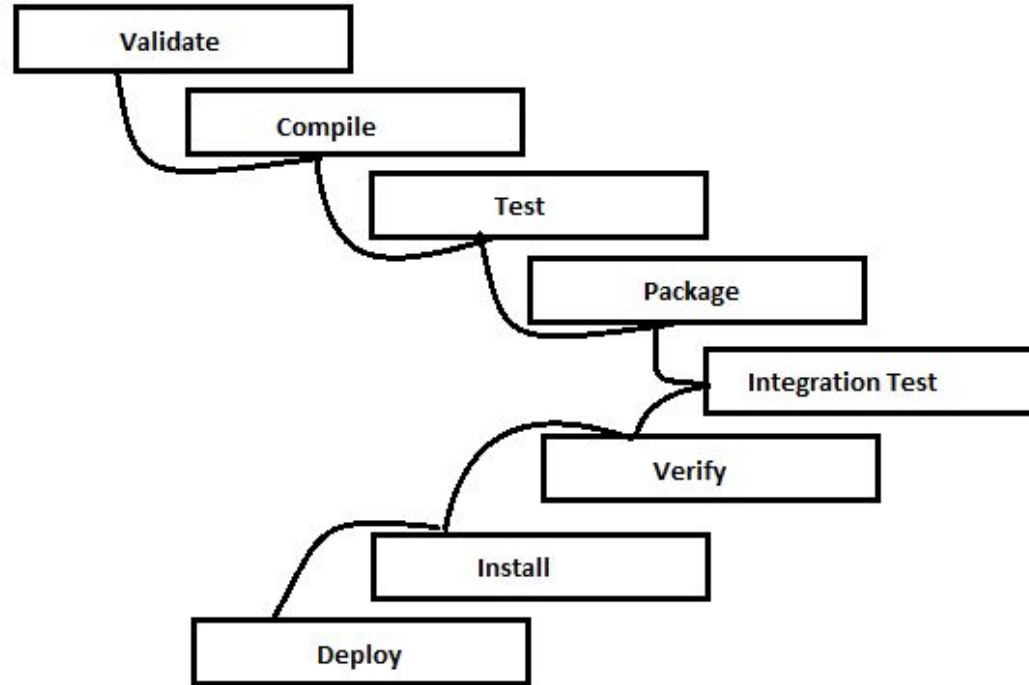
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.thetestingacademy.ra</groupId>
<artifactId>MavenDemo</artifactId>
<version>1.0-SNAPSHOT</version>
<name>MavenDemo</name>
<!-- FIXME change it to the project's website -->
<url>http://www.example.com</url>
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<maven.compiler.source>1.7</maven.compiler.source>
<maven.compiler.target>1.7</maven.compiler.target>
</properties>
```

Run: [org.apache.maven.plugins:maven-archetype-plugin:RELEASE:10 s 610 ms]

```
[INFO] Parameter: artifactId, value: MavenDemo
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Project created from Archetype in dir: /private/var/folders/62/g90p0jff91s
[INFO] -----
```



# Maven Build Cycle





# Maven Build Cycle

## Build Lifecycle Basics

Maven is based around the central concept of a build lifecycle. What this means is that the process for building and distributing a particular artifact (project) is clearly defined.

For the person building a project, this means that it is only necessary to learn a small set of commands to build any Maven project, and the [POM](#) will ensure they get the results they desired.

There are three built-in build lifecycles: default, clean and site. The `default` lifecycle handles your project deployment, the `clean` lifecycle handles project cleaning, while the `site` lifecycle handles the creation of your project's site documentation.

### A Build Lifecycle is Made Up of Phases

Each of these build lifecycles is defined by a different list of build phases, wherein a build phase represents a stage in the lifecycle.

For example, the default lifecycle comprises of the following phases (for a complete list of the lifecycle phases, refer to the [Lifecycle Reference](#)):

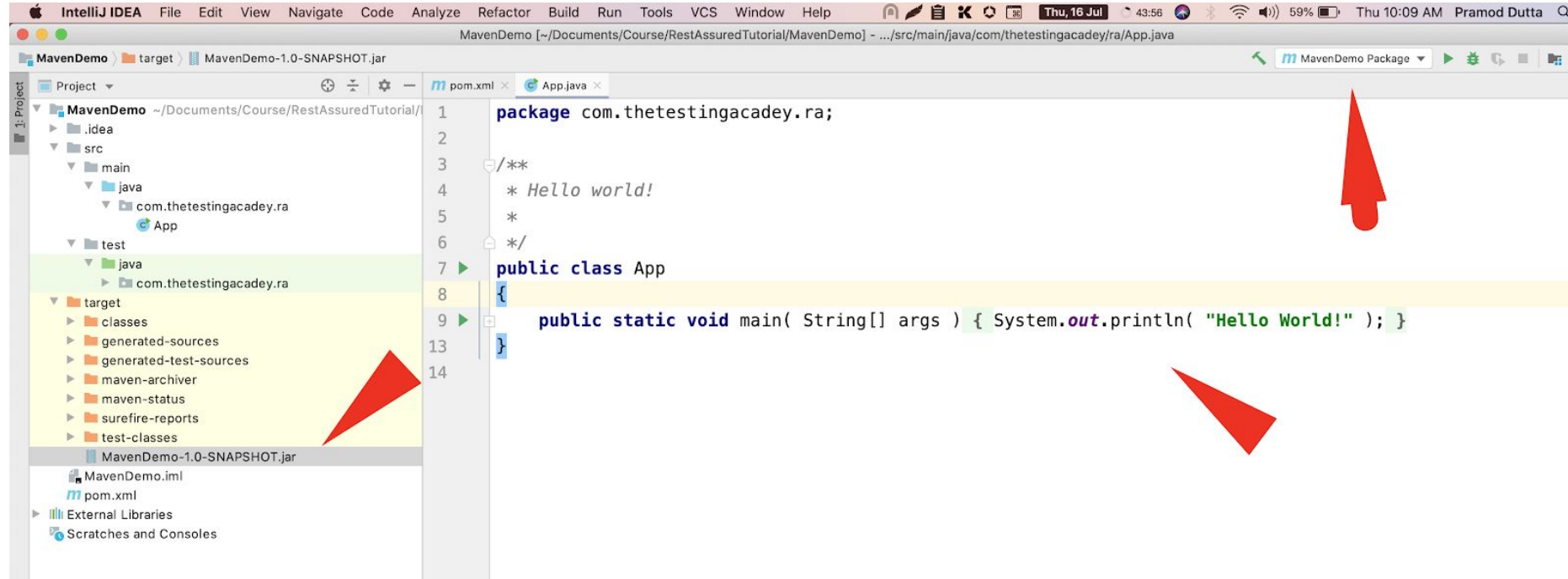
- `validate` - validate the project is correct and all necessary information is available
- `compile` - compile the source code of the project
- `test` - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- `package` - take the compiled code and package it in its distributable format, such as a JAR.
- `verify` - run any checks on results of integration tests to ensure quality criteria are met
- `install` - install the package into the local repository, for use as a dependency in other projects locally
- `deploy` - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

These lifecycle phases (plus the other lifecycle phases not shown here) are executed sequentially to complete the `default` lifecycle. Given the lifecycle phases above, this means that when the default lifecycle is used, Maven will first validate the project, then will try to compile the sources, run those against the tests, package the binaries (e.g. jar), run integration tests against that package, verify the integration tests, install the verified package to the local repository, then deploy the installed package to a remote repository.

[\[1\]](#)



# Maven Package and Run



The screenshot shows the IntelliJ IDEA interface with the MavenDemo project. The Project tool window on the left displays the file structure, with the `target` directory expanded. The `MavenDemo-1.0-SNAPSHOT.jar` file is highlighted in the `target` directory. The Run tool window on the right shows the Maven package and run process, with the `MavenDemo Package` dropdown menu selected. The `App.java` file is open in the editor, showing the following code:

```
1 package com.thetestingacademy.ra;
2
3 /**
4  * Hello world!
5  *
6  */
7 public class App
8 {
9     public static void main( String[] args ) { System.out.println( "Hello World!" ); }
10 }
11
12
13
14
```

Two red arrows point to the `MavenDemo Package` dropdown menu and the `App.java` file in the editor.



# Running Your JAR -

```
java -cp target/MavenDemo-1.0-SNAPSHOT.jar com.thetestingacademy.ra.App
```



# Two other Maven lifecycles of note beyond the default list above. They are

- clean: cleans up artifacts created by prior builds
- site: generates site documentation for this project
-



# Add TestNG in Maven

- <https://testng.org/doc/maven.html>

The dependency in your project should look like

```
<dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>6.8</version>
  <scope>test</scope>
</dependency>
```



# Assignments

- Create a Project in Maven.
- Add Rest Assured, TestNG
- Add Apache POI also.
- Build Project

401  
Response





# What You are going to Learn?

1. Install Java JDK.
2. IntelliJ
3. Create a Project.



# Setting up Environment variables in Mac

<https://www.java.com/en/download/help/path.xml>

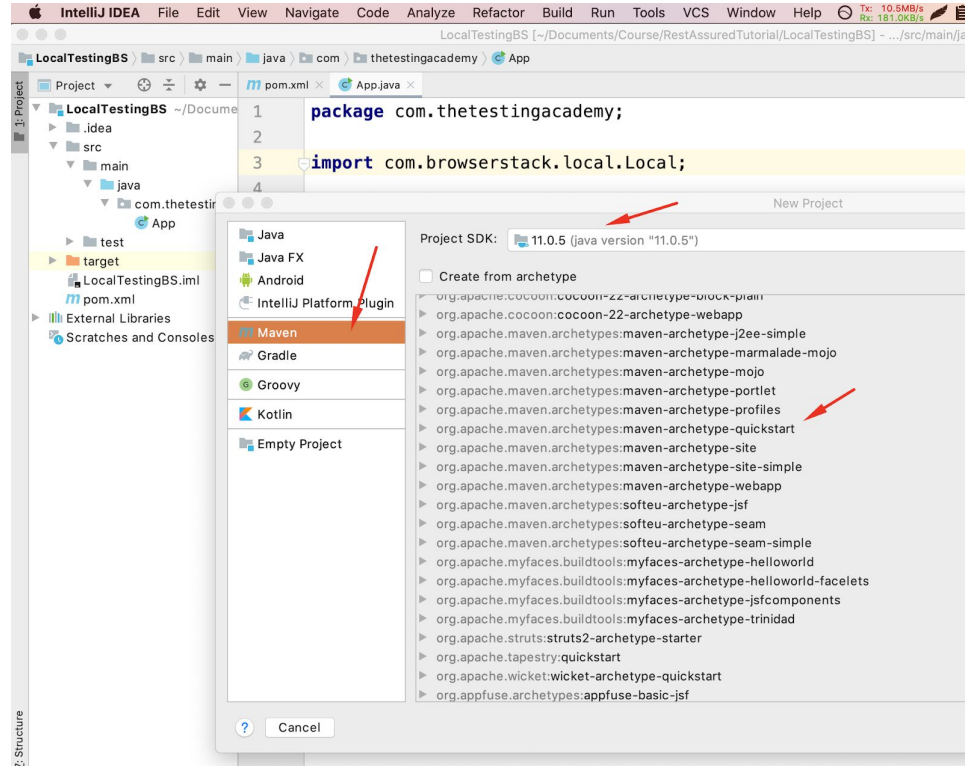


# Setting up Environment variables in Windows 10

<https://stackoverflow.com/questions/32241179/setting-up-environmental-variables-in-windows-10-to-use-java-and-javac>



# Creating a Project





# Add TestNG

## Specifying your pom.xml

The dependency in your project should look like the following:

```
<dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>6.8</version>
  <scope>test</scope>
</dependency>
```

## Sample Report

A sample surefire report with TestNG can be found [here](https://testng.org/doc/maven.html).

<https://testng.org/doc/maven.html>



# Add RestAssured

Maven

Gradle

SBT

Ivy

Grape

Leiningen

Buildr

```
<!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured -->
<dependency>
  <groupId>io.rest-assured</groupId>
  <artifactId>rest-assured</artifactId>
  <version>4.3.1</version>
  <scope>test</scope>
</dependency>
```

☒ Include comment with link to declaration

<https://mvnrepository.com/artifact/io.rest-assured/rest-assured/4.3.1>



# Add Json Schema Validator

Used by

27 artifacts

Maven

Gradle

SBT

Ivy

Grape

Leiningen

Buildr

```
<!-- https://mvnrepository.com/artifact/io.rest-assured/json-schema-validator -->
<dependency>
  <groupId>io.rest-assured</groupId>
  <artifactId>json-schema-validator</artifactId>
  <version>4.3.1</version>
</dependency>
```

☒ Include comment with link to declaration

<https://mvnrepository.com/artifact/io.rest-assured/json-schema-validator/4.3.1>



# Add JackSon Java Bind

Used By	15,728 artifacts
---------	------------------

Maven Gradle SBT Ivy Grape Leiningen Buildr

```
<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.11.1</version>
</dependency>
```

☒ Include comment with link to declaration

<https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind/2.11.1>





# Add JAXB

**JAXB API » 2.4.0-b180830...**  
JAXB API

License	CDDL 1.1
Categories	XML Processing   Java Specifications
Date	(Aug 30, 2018)
Files	jar (125 KB) View All
Repositories	Central   JCenter
Used By	3,827 artifacts

Maven | Gradle | SBT | Ivy | Grape | Leiningen | Buildr

```
<!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
<dependency>
  <groupId>javax.xml.bind</groupId>
  <artifactId>jaxb-api</artifactId>
  <version>2.4.0-b180830.0359</version>
</dependency>
```

☒ Include comment with link to declaration

<https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api/2.4.0-b180830.0359>

**Fin.**