# What is Maven?

**It's an open-source automation tool which is used for building java projects and create deployable binaries.Maven uses project object model (POM) to build java projects and create binaries.**

**In this class, We’ll try to explain the above statement with examples so that it becomes quite clear what maven is, and how to use it. First, we’ll see how we can manage a java project without any build tool, and then learn about maven and see how it makes the life of a java developer easy!**

# Building the old way

**Let’s consider we have to build a web application. We might have several dependencies such as web mvc, junits, etc. We need to manage all the dependencies, build WAR files, deploy, etc. All these steps take a lot of time.**

**Let us consider an example:We have two files: App.java and AppTest.java along with 2 jar files for writing JUnits.**

https://miro.medium.com/max/1120/1*j_mMmnRUCemVDbU4z-OYuA.png

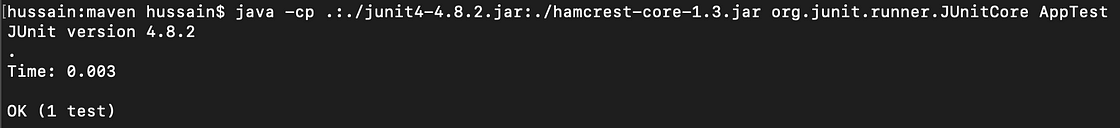
**We first need to compile App.java -> App.class**

https://miro.medium.com/max/1120/1*ecoTskNcm0-CmLWxgKk3kA.png

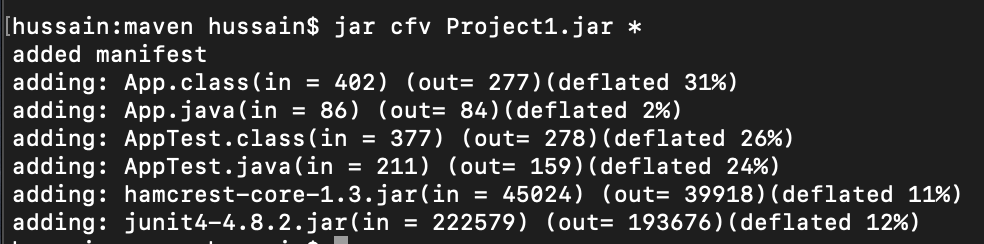
**Then we need to compile AppTest.java → AppTest.class**

https://miro.medium.com/max/1120/1*tfuxI-vBnyAZ5Qv3_bG6kw.png

**Then we need to run the test cases and check if no tests are failing.**



**Then combine all classes to jar file:**



**As you can see, the steps are quite tedious and difficult to follow along, and if you have a huge project with a lot of dependencies to manage it becomes almost impossible. Developers used ANT before to solve and automate the above process, but for this, we had to write the long scripts. Maven came in as a solution for this.**

# Using Maven as a build tool

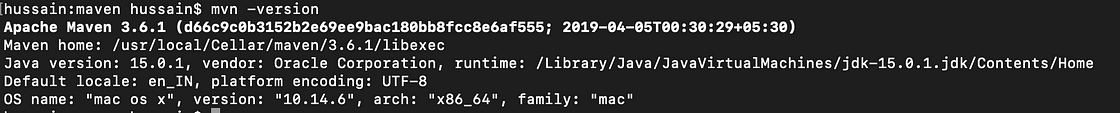
**Maven is an open-source project management build tool developed by Apache. It began as an attempt to simplify the build processes in the Jakarta Turbine project.**

**Why use maven?**

1. **It is a complete project management tool.**
2. **It makes the build process very easy (No need to write long scripts).**
3. **It provides a uniform build system. It has a standard directory structure which is followed.**
4. **It follows Convention over Configuration.**
5. **It has a remote maven repo which has all the dependencies in one place.**
6. **Can be used with other programming languages too, just not java.**

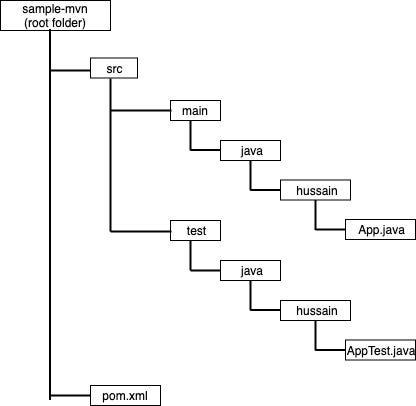
**Let us now see how maven helps us with the above project which we created. We will create a similar project, but now we’ll use Maven as the build tool.**

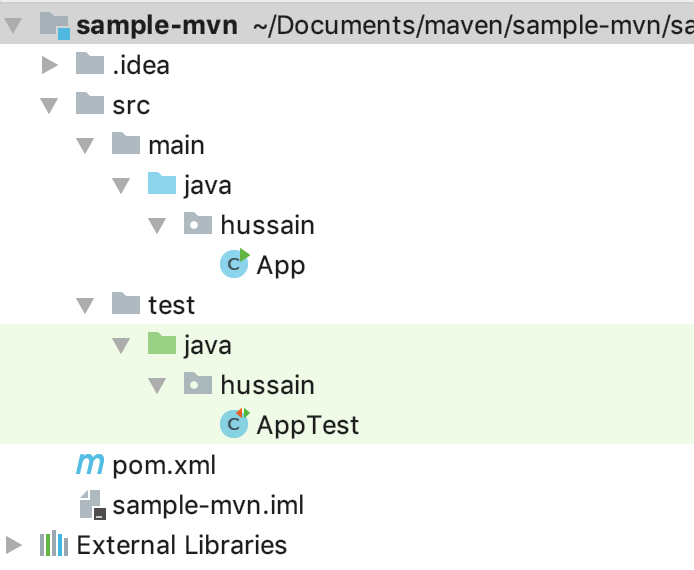
**For that first check if you have maven installed. If not please** [**install**](http://maven.apache.org/install.html) **it.**



**We will now create the same project using maven:**

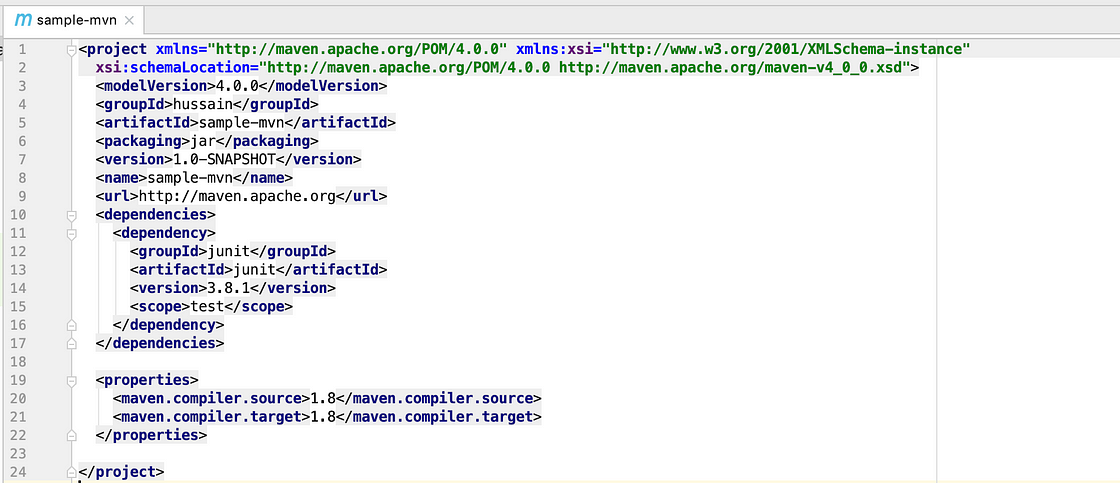
mvn archetype:generate -DgroupId=hussain -DartifactId=sample-mvn -DarchetypeArtifactId=maven-archetype-quickstart





**As Maven follows convention over configuration. Therefore it defines some defaults (can be overridden):1. Src files should be in src/main/java2. Test files should be in src/test/java3. pom.xml should be in the root folder**

**When you run any maven command, the first thing it looks for is pom.xml. We will learn more about this later.**



# Maven build lifecycle

**Maven build lifecycle is a set of phases that is executed when a maven build is run. There are three build lifecycles: default, clean, and site. The default lifecycle comprises of the following phases:**

**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 to be packaged or deployed**

**package — take the compiled code and package it in its distributable format, such as a JAR.**

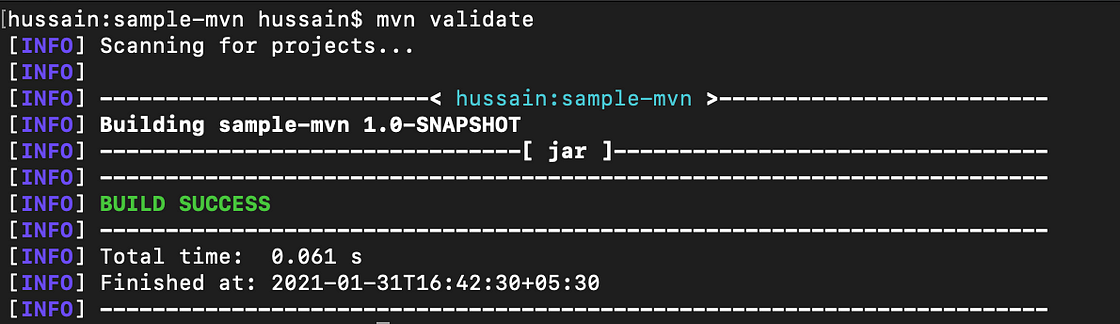
**verify — run any checks on results of integration tests to ensure quality criteria are met**

**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 are executed sequentially to complete the default lifecycle. If any specified phase is executed, all the phases before it are executed first. Let’s run a few commands to make it more clear:**

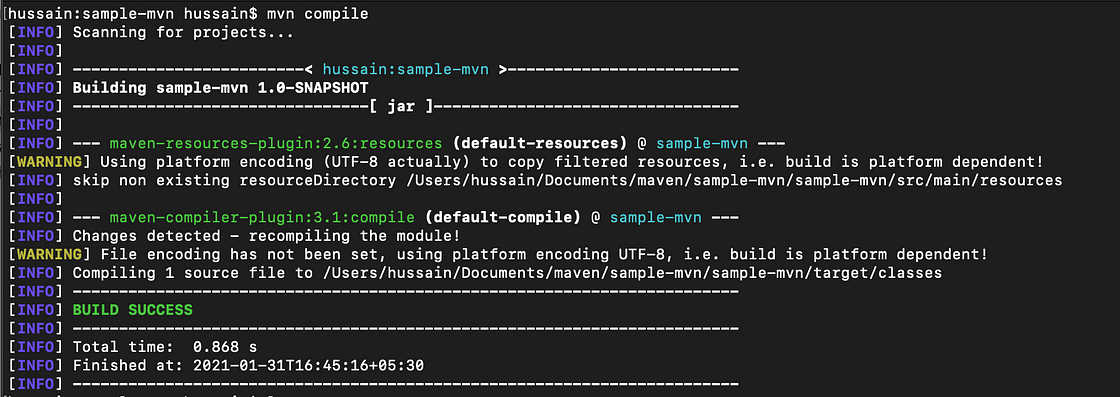
mvn validate

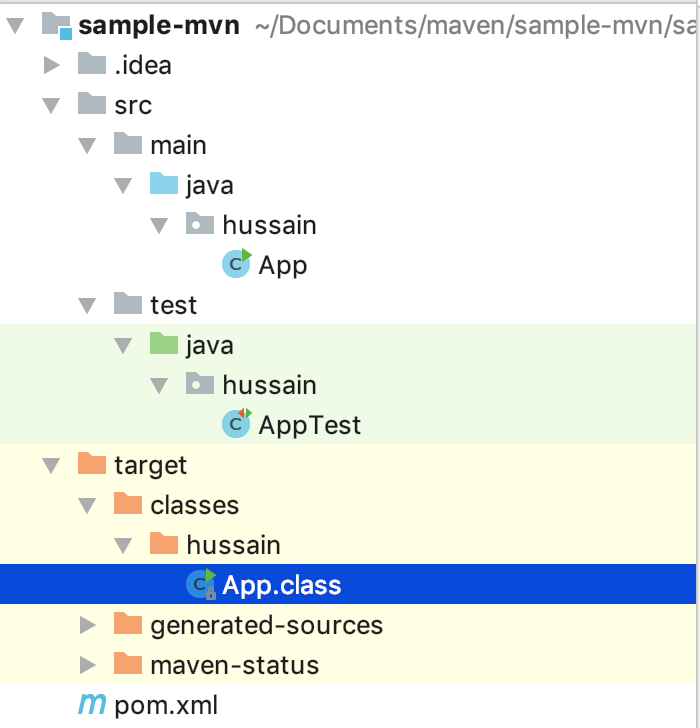


**Right now we do not have any compiled files. To compile src code, try:**

mvn compile

**You will see that a new target folder is created with src files compiled in it.**

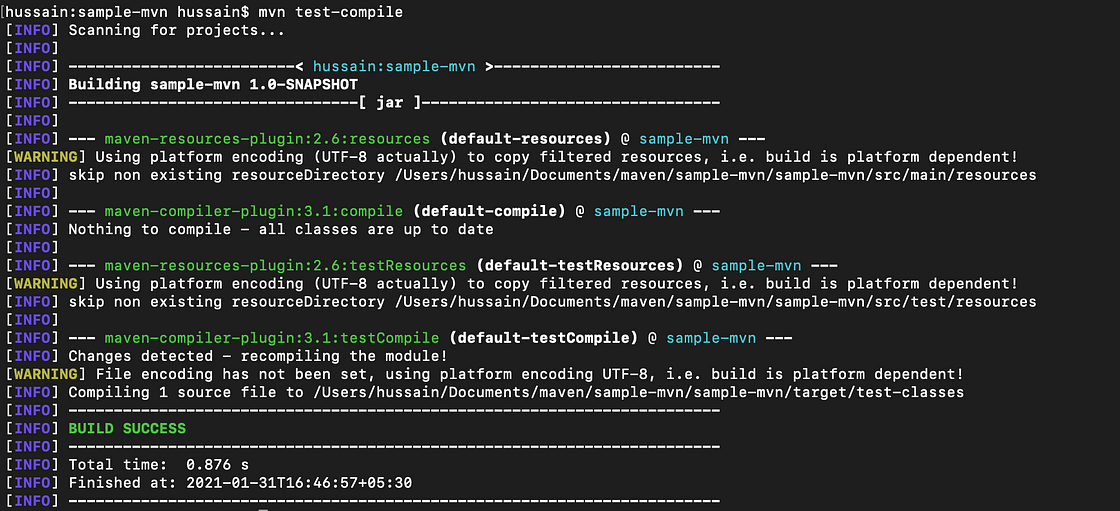


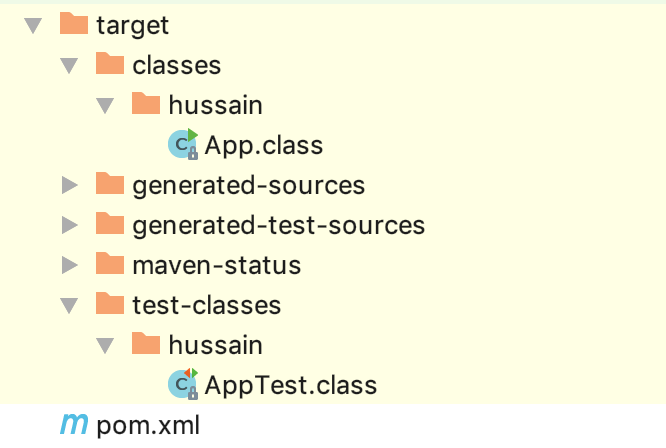


**To compile test classes too, try:**

**mvn test-compile**

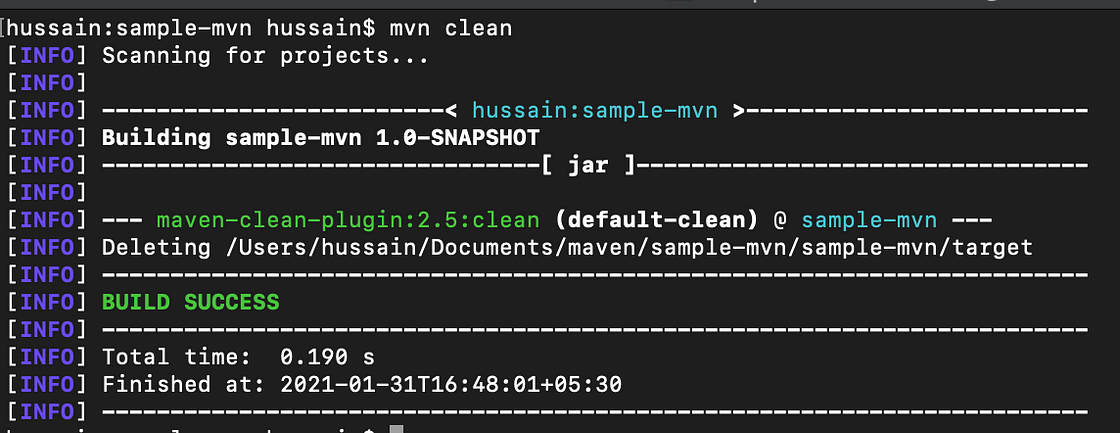
**This command compiles src files as well as test files:**





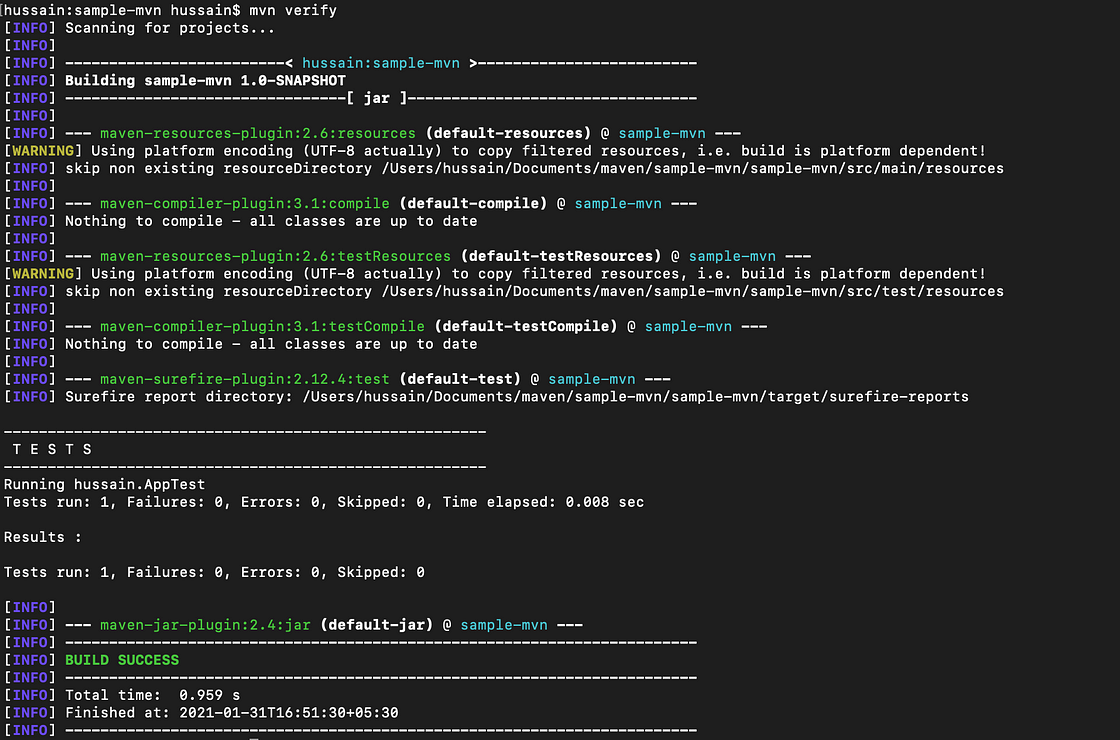
**To delete the compiled classes, try:**

mvn clean



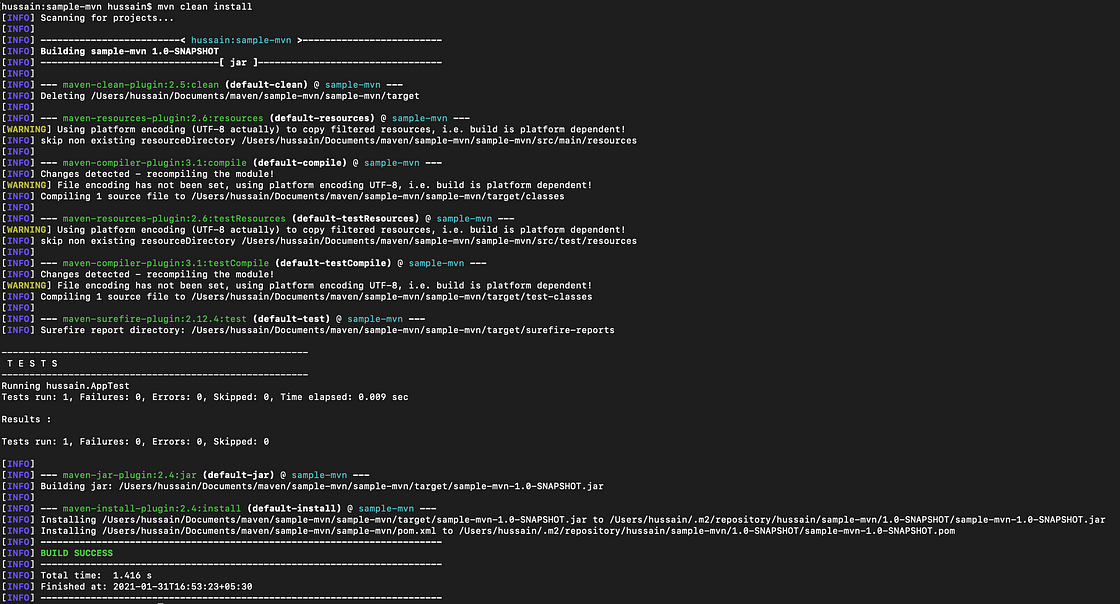
**This will delete the target folder altogether.**

mvn verify



**It compiles src and test files, runs the test, and verifies that quality criteria are met.**

mvn install



**In the install phase, maven first validates, compiles (src and test classes), packages, verifies, and then install the package into the local repository, for use as a dependency in other projects locally.**

# POM

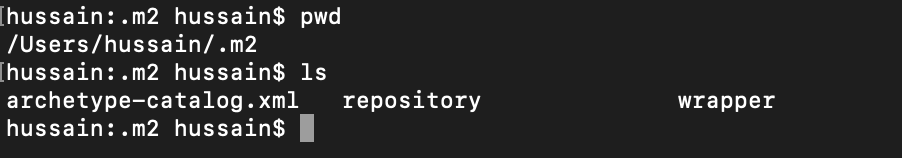
**Project object model or pom is an XML file that contains information and configuration details used by maven to build the project. When executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, then executes the goal.**

**Elements of POM:**

1. **project: root element of the XML file**
2. **modelVersion: should be set to 4.0.0**
3. **groupId: It specifies the id of the project’s group**
4. **artifactId: id of the project**
5. **version: Version of the artifact within the group**

**We have several other elements which include: packaging, dependencies, dependencyManagement, repositories, scm, modules, build, etc.**

**All the dependencies are downloaded from a remote repository into a local maven repository. This is why, while running any new project, it takes time for the project to load, as it is downloading the dependencies from remote to local.**



# Building Some Maven Assignment projects

* Project 1

# More Reading Material

<https://dev.to/saiupadhyayula/maven-complete-tutorial-for-beginners-1jek>

How to make API calls in Java?

**The REST APIs are used for fetching data from an external source. You can code the logic for calling an API in almost any programming language. In this piece, we will learn how to call an API in Java. So without any further ado, let’s jump in!**

# [Java](https://rapidapi.com/guides/make-api-call-java" \l "java)

**It is a multi-purpose object-oriented language that runs on more than 3 billion devices. You can build all types of things in Java, from web apps to mobile apps to desktop applications. You can even write games in Java. It is an open-source language that is free, fast, and powerful. The language is intended to build cross-platform applications where developers have to write code only once, and the app works on all operating systems.**

# [Making API calls in Java](https://rapidapi.com/guides/make-api-call-java" \l "making-api-calls-in-java)

**Since Java is one of the most popular languages out there, it is essential to learn how to make API calls in it so that when the need arises, you can get data from an external source in your Java app. To make things simpler, I have divided the process into a bunch of steps to make it easy for you to go through.**

## [→ STEP #1](https://rapidapi.com/guides/make-api-call-java" \l "-step-1)

**Let’s find an API that we can use. For this piece, I will use** [**Jokes API by API-Ninjas**](https://rapidapi.com/apininjas/api/jokes-by-api-ninjas?utm_source=RapidAPI.com%2Fguides&utm_medium=DevRel&utm_campaign=DevRel)**, but you can use any API you like.**

**The API I have selected is a free one, so you do not need to subscribe to it. However, you will need your RapidAPI key. So go ahead and save the value of x-rapidapi-key to use later.**

## [→ STEP #2](https://rapidapi.com/guides/make-api-call-java" \l "-step-2)

**Java needs a compiler to compile its code. Without it, your code will not run. So before we do anything, click on this** [**link**](https://www.oracle.com/java/technologies/downloads/) **and download the Java SDK for your operating system. Once it is downloaded, go ahead and install it.**

**To check everything has taken place as expected, open your terminal and run the following command**:

javac –version

**You should see a version number when you run the command. If you don’t, please restart your system.**

## [→ STEP #3](https://rapidapi.com/guides/make-api-call-java" \l "-step-3)

**The Java files have an extension of .java. So, go ahead and open your preferred code editor and create a call.java file inside it. It will give you an empty Java file.**

## [→ STEP #4](https://rapidapi.com/guides/make-api-call-java" \l "-step-4)

**We will use the HttpRequest package from the Java SDK to create an API call. So let’s import it at the top of the file.**

JAVA

import java.net.http.HttpRequest;

**To handle the response we will receive from the HttpRequest package, we have to use the HttpResponse package**.

import java.net.http.HttpResponse;

**Now to send the API call, we would need to import another package. So let’s do it.**

import java.net.http.HttpClient;

**Let’s import the Java IOException package to handle exceptions in our code.**

import java.io.IOException;

## [→ STEP #5](https://rapidapi.com/guides/make-api-call-java" \l "-step-5)

**Now create a class, and inside it, write the main function. You can copy the code snippet below to do it.**

class call {

public static void main(String[] args) {

}

}

**Now let’s create an HttpRequest builder to add all the things we will need to call an API**

class call {

public static void main(String[] args) {

HttpRequest request = HttpRequest.newBuilder().uri(URI.create("https://jokes-by-api-ninjas.p.rapidapi.com/v1/jokes"))

.header("X-RapidAPI-Host", "jokes-by-api-ninjas.p.rapidapi.com")

.header("X-RapidAPI-Key", "your-rapidapi-key")

.method("GET", HttpRequest.BodyPublishers.noBody())

.build();

}

}

**You can see that we have created a builder, giving it the API endpoint by passing it as a parameter to the uri method. Afterward, we set the header values to provide details like API host and API key. You should replace the value of X-RapidAPI-Key with the value you saved earlier in step #1.**

**Now we need to create a simple HttpResponse variable to save to the response we will receive from the API.**

class call {

public static void main(String[] args) {

HttpRequest request = HttpRequest.newBuilder()

.uri(URI.create("https://jokes-by-api-ninjas.p.rapidapi.com/v1/jokes"))

.header("X-RapidAPI-Host", "jokes-by-api-ninjas.p.rapidapi.com")

.header("X-RapidAPI-Key", "your-rapidapi-key")

.method("GET", HttpRequest.BodyPublishers.noBody())

.build();

HttpResponse<String> response = null;

}

**}**

**Finally, we need to send the API call. The HttpClient package will be used here.**

class call {

public static void main(String[] args) {

HttpRequest request = HttpRequest.newBuilder()

.uri(URI.create("https://jokes-by-api-ninjas.p.rapidapi.com/v1/jokes"))

.header("X-RapidAPI-Host", "jokes-by-api-ninjas.p.rapidapi.com")

.header("X-RapidAPI-Key", "your-rapidapi-key")

.method("GET", HttpRequest.BodyPublishers.noBody())

.build();

HttpResponse<String> response = null;

try {

response = HttpClient.newHttpClient().send(request, HttpResponse.BodyHandlers.ofString());

} catch (IOException e) {

e.printStackTrace();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(response.body());

}

}

**Error handling while making API requests is essential. That’s why I have enwrapped the API request within a try/catch block because the call can fail, leading to an error. Now go ahead and run this file in the terminal. It will make an API call and print the response in the terminal.**

# [Wrap Up](https://rapidapi.com/guides/make-api-call-java" \l "wrap-up)

**This is how simple it is to call APIs in Java. I hope you have learned something from this piece, and now you can call APIs in your Java applications.**