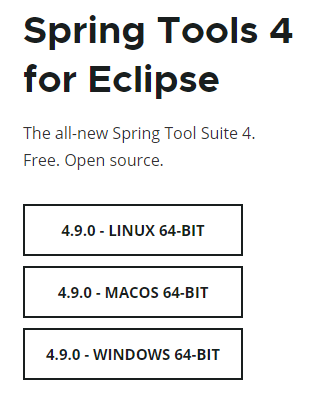
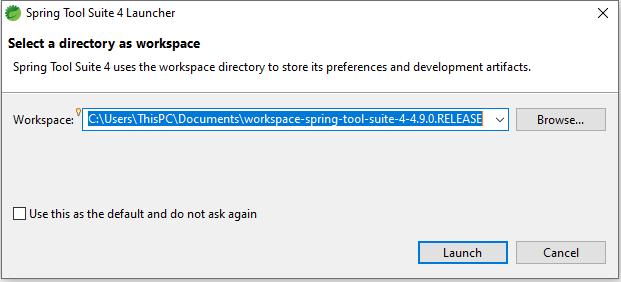
**Download and install the STS IDE (Spring Tool Suite)**

Spring Tool Suite is an IDE to develop spring applications.

1. Download Spring Tool Suite from <https://spring.io/tools3/sts/all>. Click on the platform which you are using



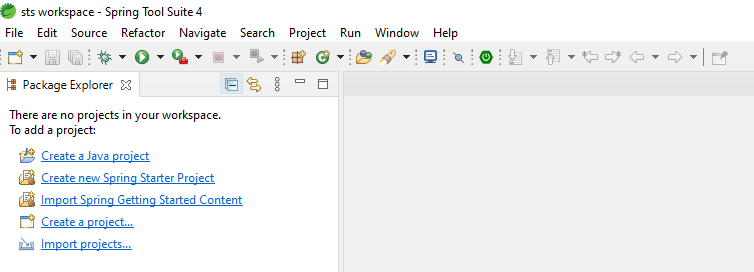
1. Select the windows 64 bit option
2. Go to download folder location, then double click on that file or Right click on file-> unzip it.
3. Go to sts-4.9.0.RELEASE folder.
4. Double click on SpringToolSuite4.exe file.
5. Spring too suite launcher box apprears on screen as



1. Set the workspace location as default or if you want you can change it.

Note- The best way is always create the workspace on any drive except c drive. Due to some system issue or virus you need to immediately format the c drive.

1. Click on launch button and it takes some time.
2. STS looks like



**Spring Boot-**

What is the Spring Boot?

Spring Boot is a spring framework module which provides RAD (Rapid ApplicationDevelopment) feature to the spring framework.

**Features of spring boot**

1. **Avoiding heavy configuration of** **XML** which is present in spring.
2. **Embedded Tomcat-server**
3. **Deployment is very easy**
4. **Lazy initialization** of the Spring applications. After enabling the lazy initialization, beans are created as per the requirement rather than during the application startup.
5. **Facility to creates profiles based on different environment** such as

Dev, Test, Prod etc.

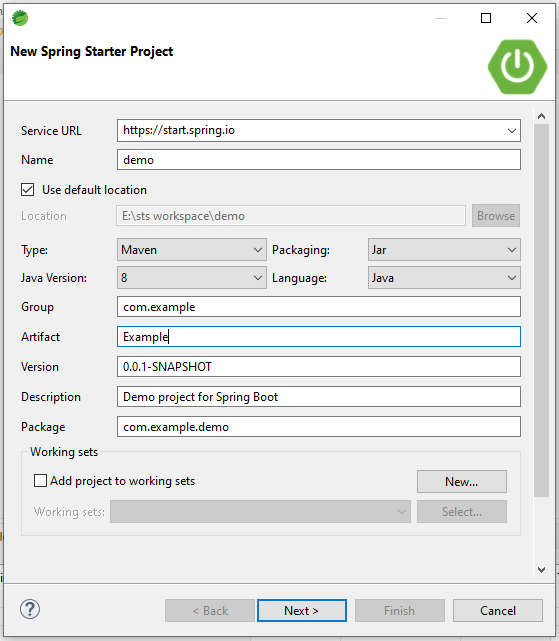
1. **Properties File –** it is used to set the properties in application.properties file such as server.port = 8082
2. **Security** which provides different authentication for end point (URL)

**Pre-requisites**

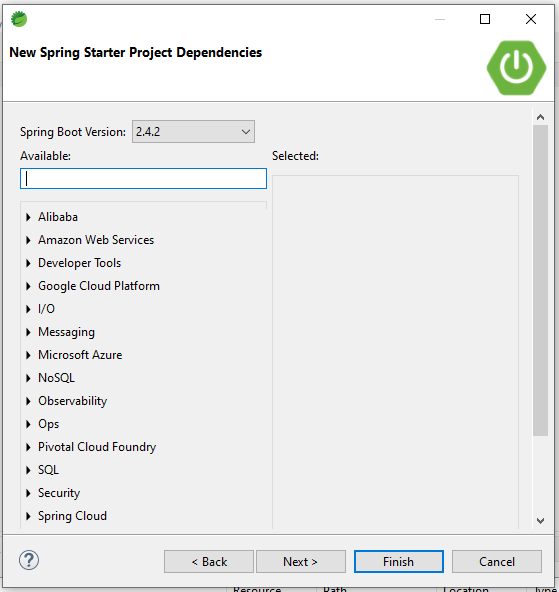
* Java 1.8
* Maven 3.0+
* An IDE (Spring Tool Suite) is recommended.

**How to create the Spring Boot Maven Project-**

File->New-> Spring Starter Project-> dialog appears as



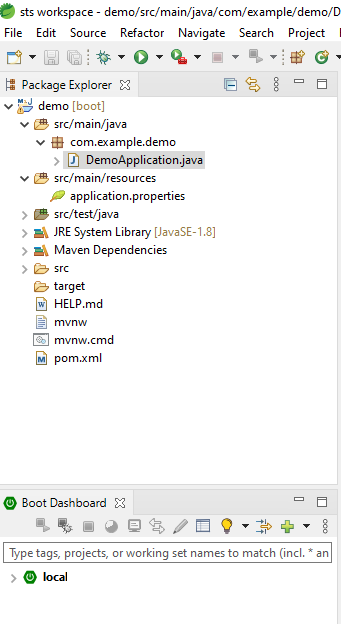
Enter the group id and artifact id and click on Next button.



If you want, you can select the dependencies otherwise click on finish button.

Then

Project structure looks like as



Note- @SpringBootApplication this annotation is equivalent to @Configuration, @ComponentScan, @EnableAutoConfiguration

**Spring Boot Dependency Purpose**

**Spring Boot Starter Actuator-**

It is used to monitor and manage your application.

**Spring Boot Starter Security-**

It is used for spring security.

**Spring Boot Starter web-**

It is used to write a rest endpoints.

**Spring Boot Starter Test-**

It is used for writing test cases.

**Spring Boot Starter-**

It is starter dependency which is used for all the spring-based application.

**Spring Boot Annotation-**

**@SpringBootApplication**

This annotation indicates that class should have the main method to run the Spring Boot application.

**@ComponentScan**

This annotation scans all the beans (POJO or Model Class) and package declarations that is base package.

**@EnableAutoConfiguration**

This annotation scans sub packages and all the classes which are available in spring boot application.

**Rest API Annotation**

**@RestController**

This annotation is used for developing a RESTful web service

**@RequestMapping**

There are no of rest Controller class contains several handler methods to handle different HTTP requests but how does Spring map a particular request to a particular handler method.

**@PathVariable**

This annotation is used to retrieve data from the URL and identify the method arguments.

**@RequestBody**

This annotation can convert inbound HTTP data into Java objects passed into the controller's handler method.

**@ResponseBody**

This annotation which binds a method return value to the web response body.

**@ResponseStatus**

This annotation can be used to override the HTTP response code for a response. You can use this annotation for error handling while developing a web application or RESTful web service.

**Rest API-**

Why?

Suppose I have one online website called as fresherworlds.com in which I want to show the tour packages of different countries then what options I have?

There is only one option call the API (Application Programming Interface) of tour packages into my site.

That’s why Rest API’s comes into picture.

Or

If your application supports third party application in that case we can use REST API.

What is REST?

The term REST stands for **RE**presentational **S**tate **T**ransfer. It is an architectural style that defines a set of rules in order to create Web Services.

In a client-server communication, REST suggests to create an object of the data requested by the client and send the values of the object in response to the user.

There are four types of method as

* Get-It is used to read resource
* Post-It is used to create new resource.
* Put-It is generally used to update resource
* Delete-it is used to delete source

Pre-requisites-

1. Create the spring boot project.
2. Pom.xml
3. Postman tool

<?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 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.4.2</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>Example</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>demo</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Example-

Student.java

**package** com.spring.model;

**public** **class** Student {

**private** **Long** id;

**private** String name;

**private** String city;

**public** **Long** getId() {

**return** id;

}

**public** **void** setId(**Long** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

@Override

**public** String toString() {

**return** "Employee [id=" + id + ", name=" + name + ", city=" + city + "]";

}

}

SpringBootDemoApplication.java

package com.spring;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringBootDemoApplication {

public static void main(String[] arg) {

SpringApplication.run(SpringBootDemoApplication.class, arg);

}

}

StudentController.java

**package** com.spring.controller;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.spring.model.Student;

@RestController

@RequestMapping("/student")

**public** **class** StudentController {

@GetMapping("/getdata")

**public** String getTest() {

**return** "student data";

}

@GetMapping("/getlist")

**public** List<String> getStudentData() {

List<String> list = **new** ArrayList<String>();

list.add("ram");

list.add("10");

list.add("pune");

**return** list;

}

@GetMapping("/getStudent")

**public** Student getStudentDetails() {

Student student= **new** Student();

**return** student;

}

}

Open the postman tool and enter the URL as

<http://localhost:8080/student/getdata>

and press enter button

will see response as

student data