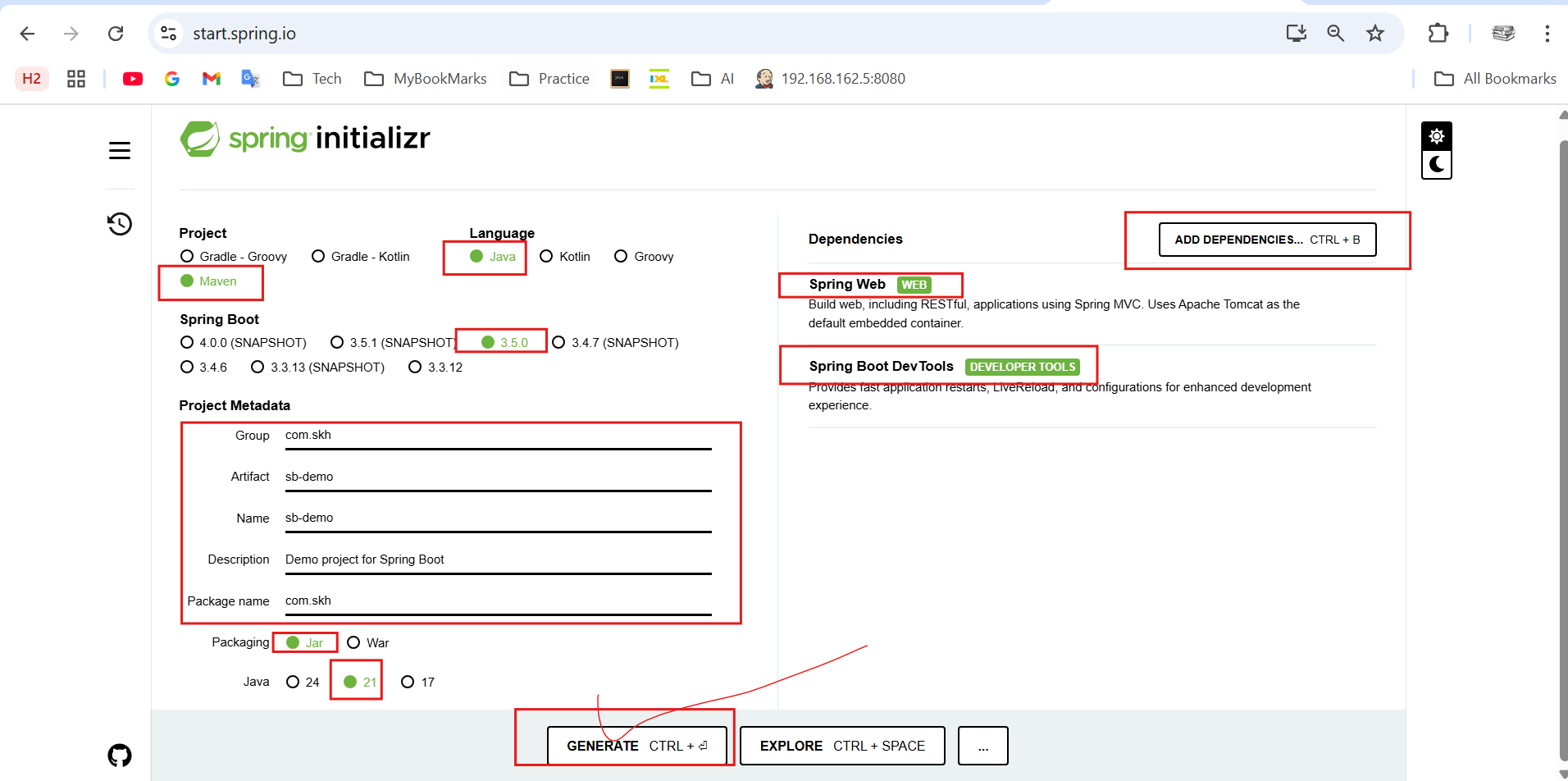
How to create Spring boot project?

Go to browser 🡪 open website called 🡪 <https://start.spring.io/> 🡪



If you don’t know java version installed in your computer.

Open command prompt and run > **java –version** it will display java version in your system.

Finally click on “**Generate**” button.

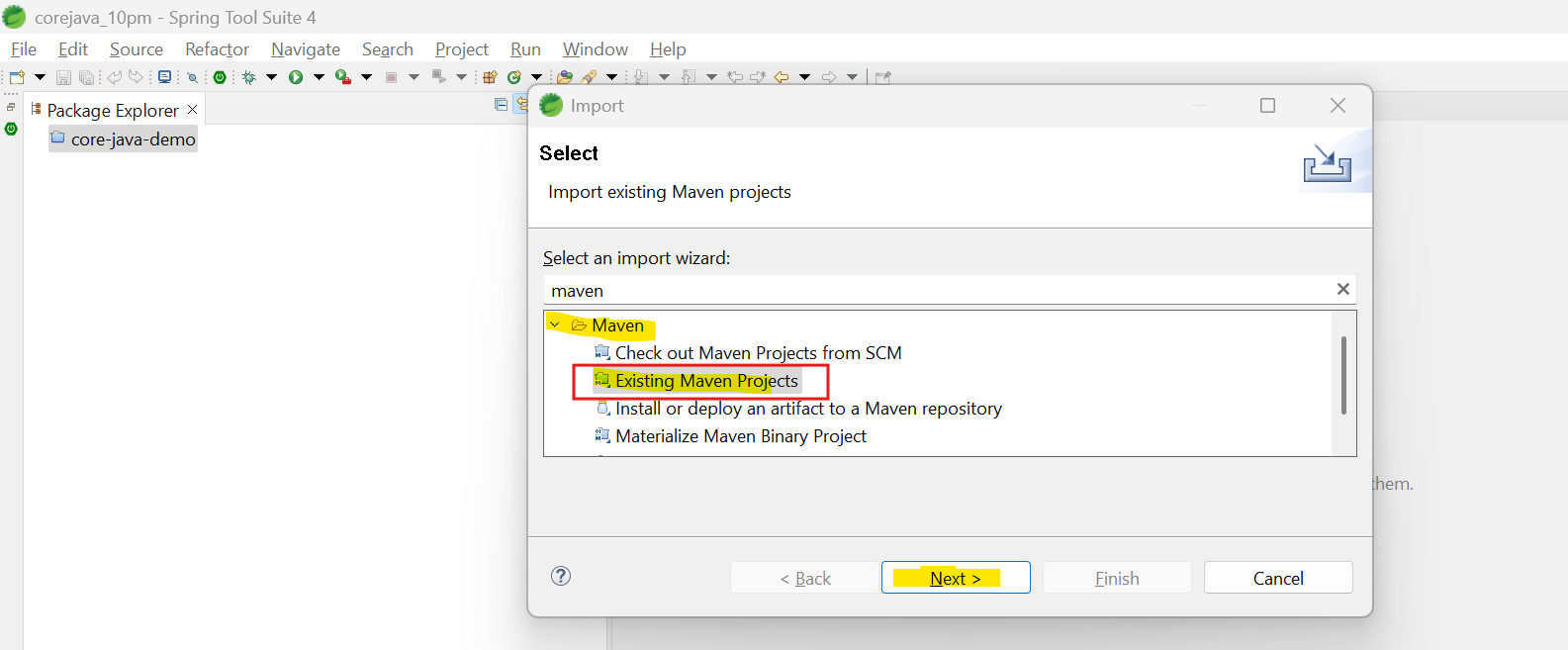
It will download the .zip file into downloads folder.

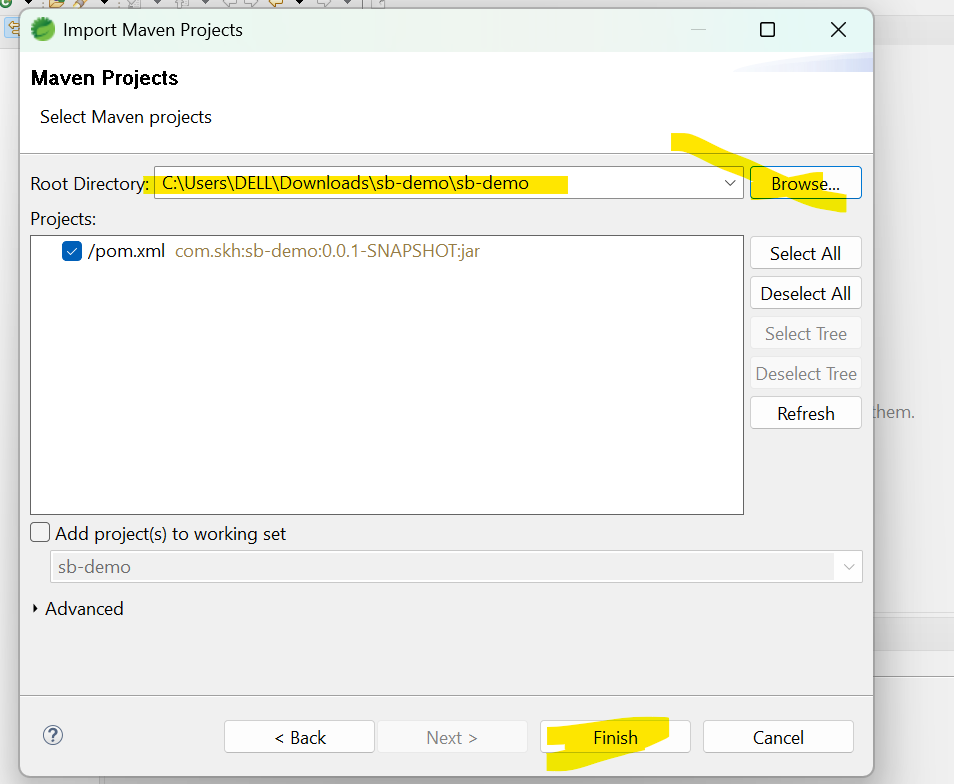
Go to Downloads folder and **extract** the .**zip** file. You will get the project folder.

Now open STS / IntelliJ and import this new project there.

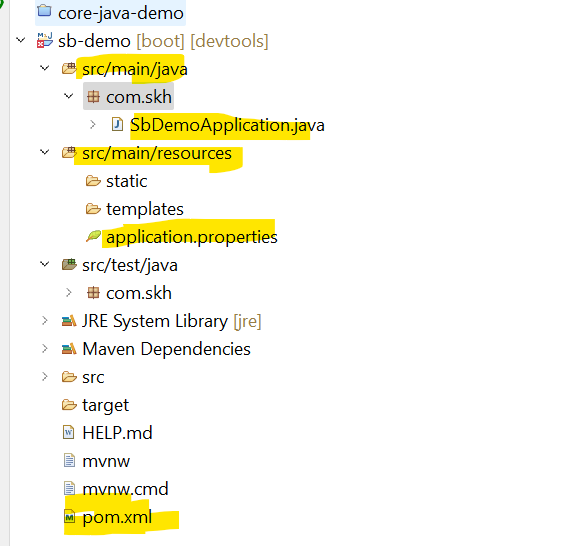
How to import new project into STS?

Open STS 🡪 File 🡪 Import 🡪 maven 🡪 Existing maven project 🡪 Next 🡪 Browser your project from downloads folder 🡪 click on Finish button.





After clicking project wait for few minutes. It will be downloaded from server.



**Project 1:**

Changes required to do:

1.**create a class** and add **@RestController** annotation on class. Eg: DemoController.java.

2.change name of **application.properties** file name into **application.yaml** file.[right click -> refractor-> rename]

3.[**OPTIONAL**]In the application.yaml file add server port number, number can be anything you interested. Like.. 9000,

9001, 9002…….etc.

4.create methods in a class you created, and add **Request method type annotations**….with **path** endpoint.

**@GetMapping(path=”/endpoint”), @PostMapping(path=”/endpoint”), @PutMapping, @DeleteMapping**..

5.start the server:

1st approach: right click on project 🡪 Run As 🡪 Spring boot App 🡪 server will start, you can see on console.

2nd approach: Open main class 🡪 right click on class 🡪 run as 🡪 java application.

6.Access the controller class method endpoint using **IP address** and **port number** and **method end point**.

<http://localhost:9000/fetchName>

here ->**http** is server **PROTOCOL**.

->**localhost** is **IP Address** of the server.

-> **9000** is **port number** of running server.

-> **fetchName** is controller method **endpoint**.

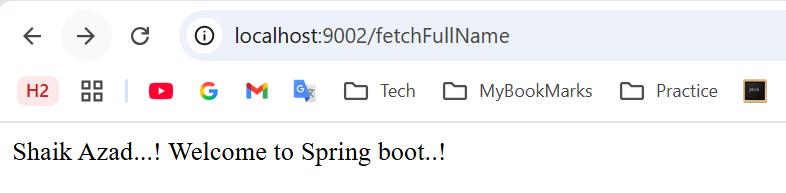
Go to Browser and use above URL.

**DemoController.java application.yaml**

|  |  |
| --- | --- |
| **package** com.skh;  **import** org.springframework.web.bind.annotation.GetMapping;  **import** org.springframework.web.bind.annotation.RestController;  **@RestController**  **public** **class** DemoController {    **@GetMapping(path = "/fetchName")**  **public** void m1() {  System.***out***.println("Hi this is Kamal..!");  }    @GetMapping(path = "/fetchNumber")  **public** **int** m2() {  **return** 123;  }    @GetMapping(path = "/fetchFullName")  **public** **String** m3() {  **return** "Shaik Azad...! Welcome to Spring boot..!";  }  } | server:  port: 9001 |

In application.yaml/yml 🡪 always maintain spaces properly.

If we don’t add port number in application.yaml file. Tomcat server by default takes 8080 as default port number.



**How many ways we can send the request to the server?**

There are so many ways are there.

1. Using browser URL
2. HTML
3. Javacript
4. **Postman – real time tools**
5. **jMeter – real time tools**

etc……

**How to send data from UI / browser to controller method?**

1. **Using path variable:** 🡪 URL**/inputValue 🡺**

[http://localhost:9002/fetchFullName/**John**](http://localhost:9002/fetchFullName/John)**....**

@GetMapping(path = "/fetchFullName**/{fName}")**

**@PathVariable** annotation in method parameter level.

|  |
| --- |
| @GetMapping(path = "/fetchFullName/{fName}")  **public** String m3(@PathVariable String fName) {  **return** String.*format*("Hi %s welcome to Spring boot", fName);  } |

1. **Using query String**. 🡪 URL**?key=value&key=value&key=valueY&key=value&….**

[**http://localhost:9002/sendQueryParams?name=kamal&city=hyd&pin=534456**](http://localhost:9002/sendQueryParams?name=kamal&city=hyd&pin=534456)

**@RequestParam**

Here : http – protocol

localhost – IP Address

9002 – PORT number.

sendQueryParams – Endpoint

?name=kamal&city=hyd&pin=534456 – Query String.

<http://localhost:9002/sendQueryParams?name=Arafth&city=mvg&pin=343434>

|  |
| --- |
| @GetMapping(path = "/sendQueryParams")  public String m4**(@RequestParam** String name, @RequestParam String city,  @RequestParam String pin) {  return String.*format*("My details %s - %s - %s ",name, city, pin);  } |

1. By submitting form.

|  |
| --- |
| @GetMapping(path = "/percentage**/{tmarks}/{hMarks}/{emarks}**") public Integer calculatePercentage(  **@PathVariable** Integer tmarks,  **@PathVariable** Integer hMarks,  **@PathVariable** Integer emarks) {  Integer percentage = ((tmarks + hMarks + emarks) \* 100) / 300;  return percentage; } |

We have seen problem with Path vriables. If we want o send more values we will send n path variables only but it will confuse us.

<http://localhost:9001/percentage/67/75/88/56/ssdfsdf/454543/sfsdfsdf/2242343>

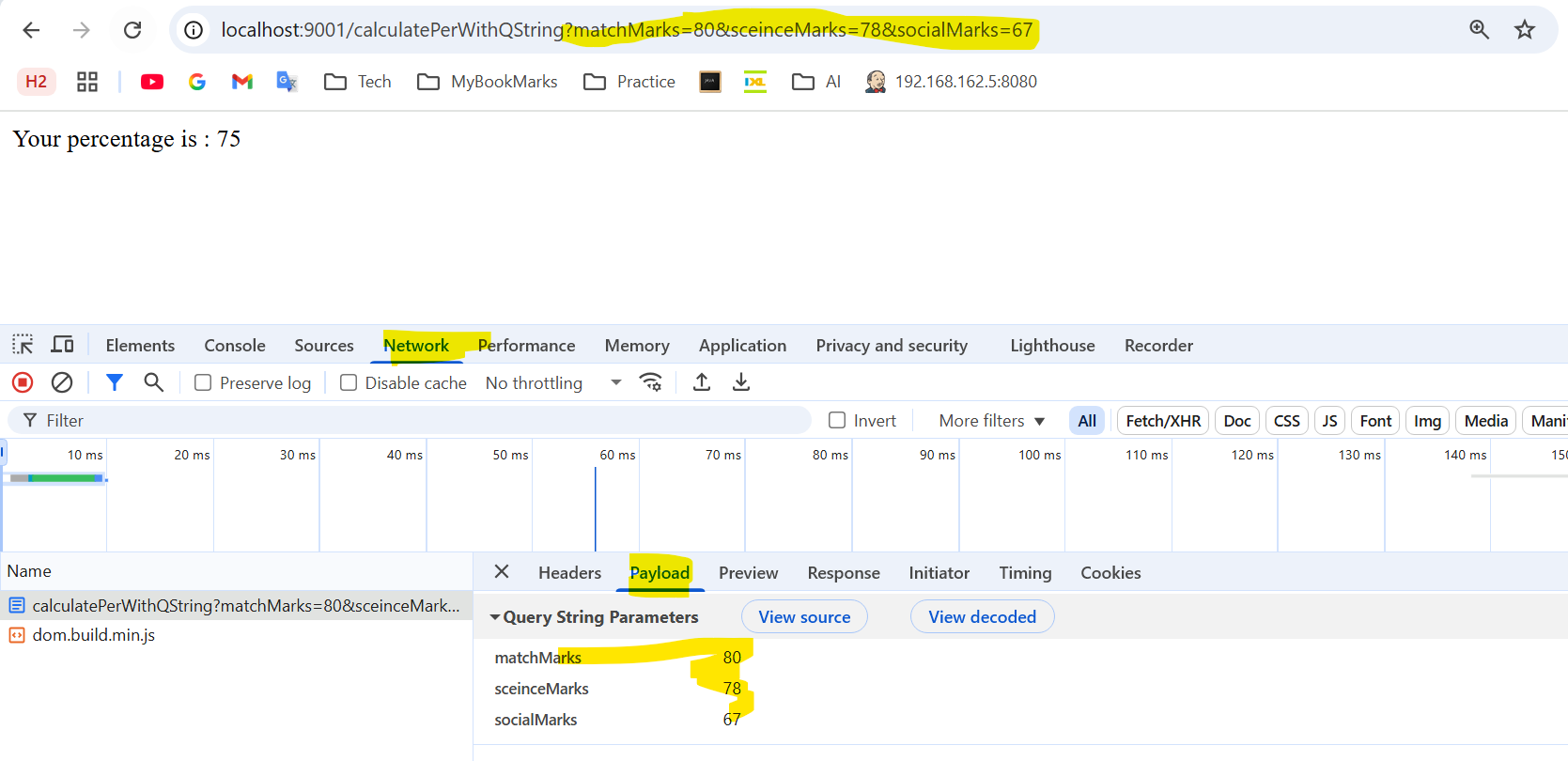
In this above URL, we don’t know which values is what for?

To overcome this we can use of “Query String” approach.

Payload means? 🡪 what ever the data we are sending to server is called as Payload.

<http://localhost:9001/calculatePerWithQString?matchMarks=80&sceinceMarks=78&socialMarks=67>

|  |
| --- |
| @GetMapping(path= "/calculatePerWithQString") public String calculatePerWithQString(  **@RequestParam** Integer matchMarks,  **@RequestParam** Integer sceinceMarks,  **@RequestParam** Integer socialMarks) {  Integer percentage = ((matchMarks + sceinceMarks + socialMarks) \* 100) / 300;  return "Your percentage is : "+percentage; } |



**When we send request, by default browser always sends “GET” request only.**

Particularly path variable and Query string approach always sends GET request only.