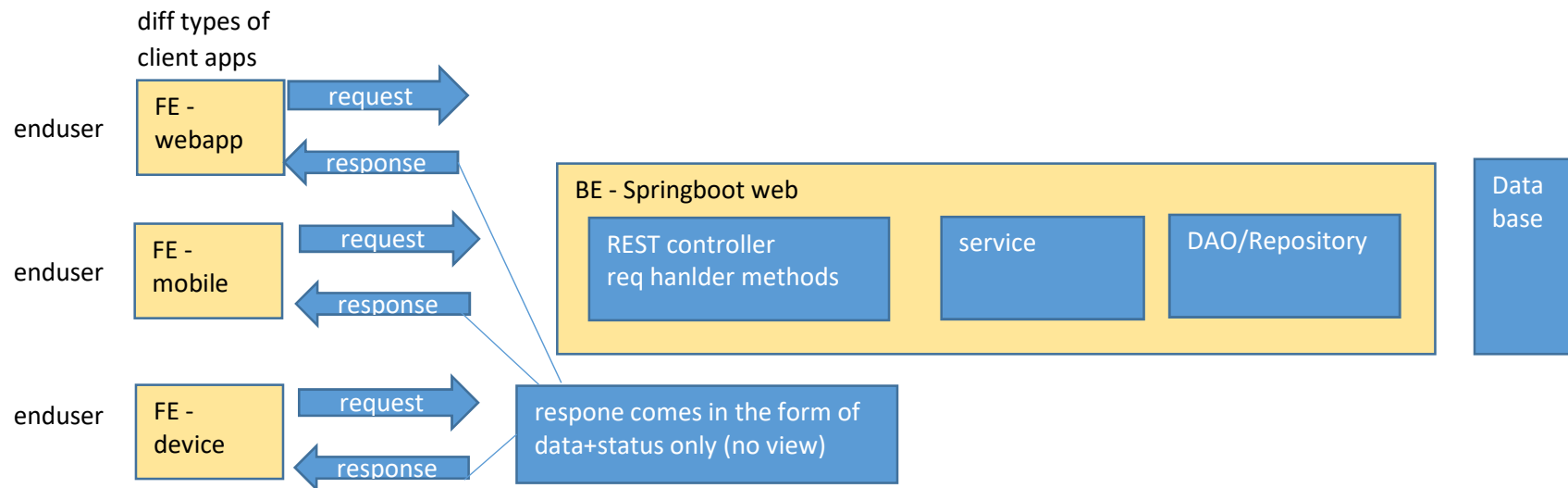
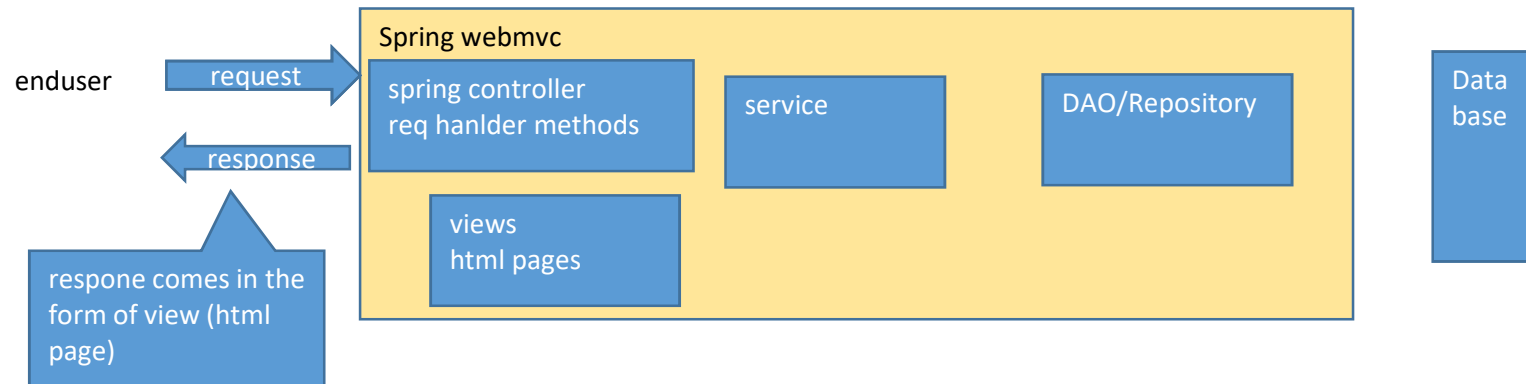
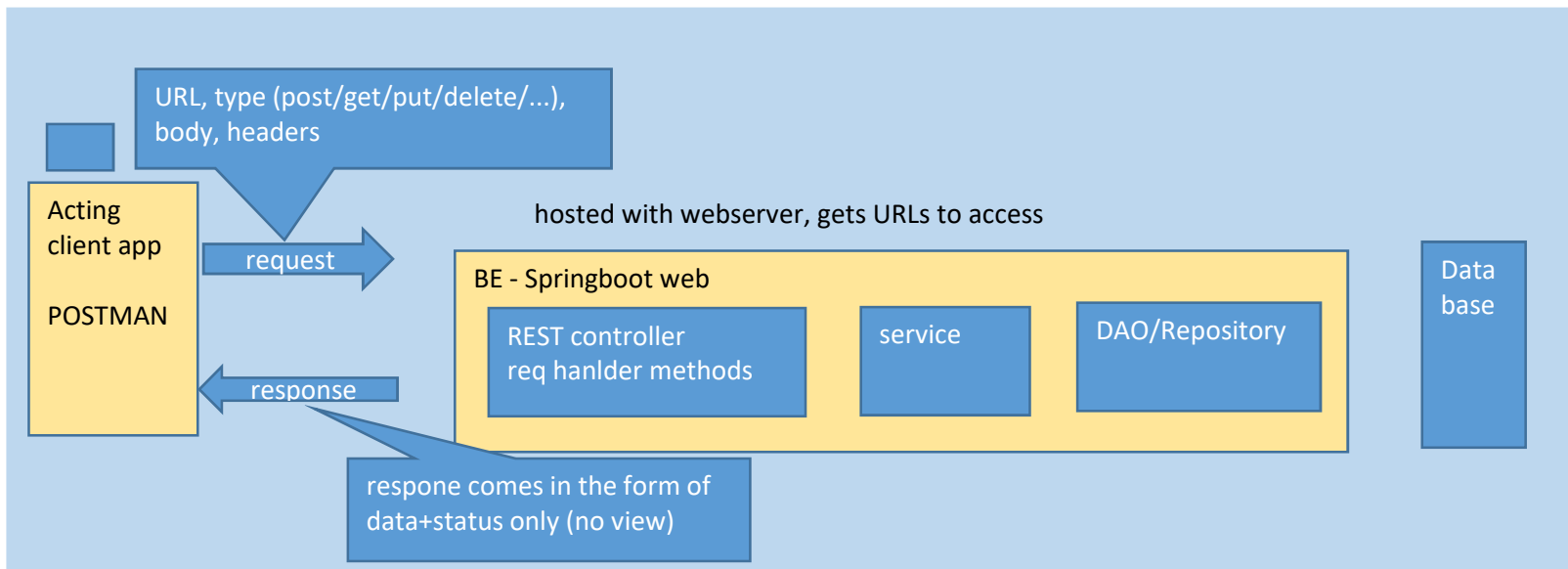


Spring MVC

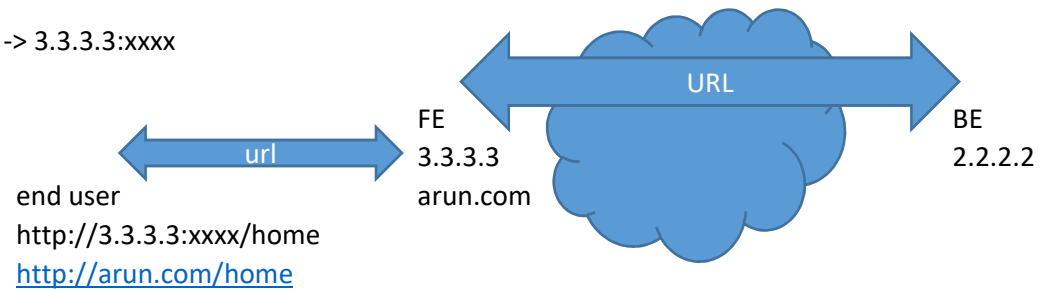
End User <-> UI <-> Controller <-> Service <-> DAO <-> DB





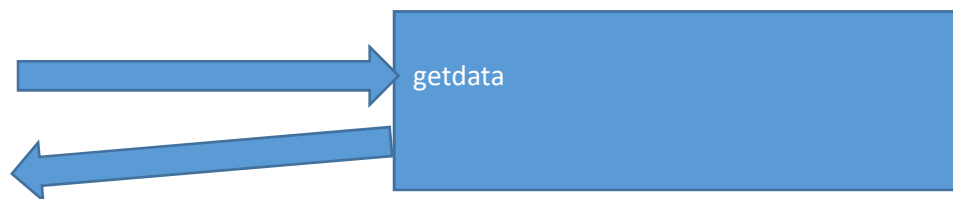
dns

arun.com -> 3.3.3.3:xxxx



REST Controller

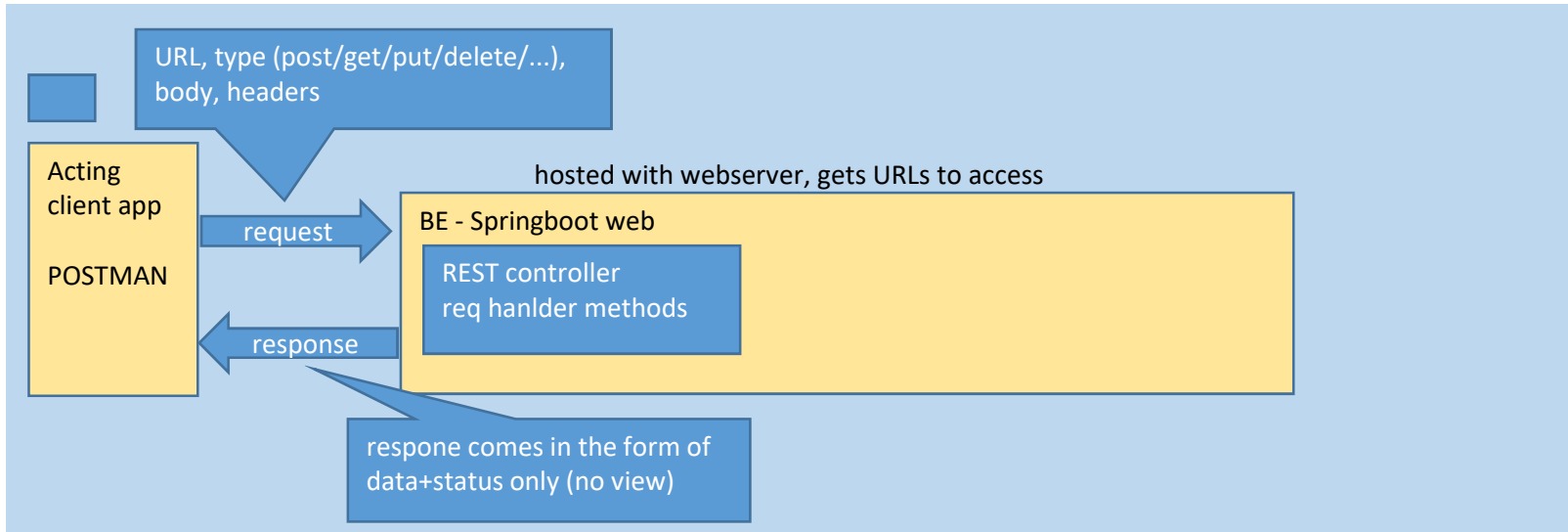
Holds request handler methods



Example 1

Creating simple springboot web application

Implementing request handling with REST controller



Step 1 Create new springboot application using spring initializer
add dependency 'web'

The screenshot shows the Spring Initializr web application in a browser. The URL is start.spring.io. The page has a sidebar with a hamburger menu icon. The main content area is divided into sections: Project, Language, Spring Boot, Project Metadata, and Dependencies. In the Project section, Maven is selected. In the Language section, Java is selected. In the Spring Boot section, 2.7.6 is selected. In the Project Metadata section, the Group is com.stackroute, Artifact is webapp1, Name is webapp1, Description is Demo project for Spring Boot, and Package name is com.stackroute.webapp1. In the Packaging section, Jar is selected. In the Dependencies section, Spring Web is selected, and there is a button to add dependencies. A green banner at the bottom says "You are screen sharing" and a red button says "Stop Share".

start.spring.io

spring initializr

Project

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ Java ☐ Kotlin ☐ Groovy

☒ Maven

Spring Boot

☐ 3.0.1 (SNAPSHOT) ☐ 3.0.0 ☐ 2.7.7 (SNAPSHOT) ☒ 2.7.6

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 19 ☒ 17 ☐ 11 ☐ 8

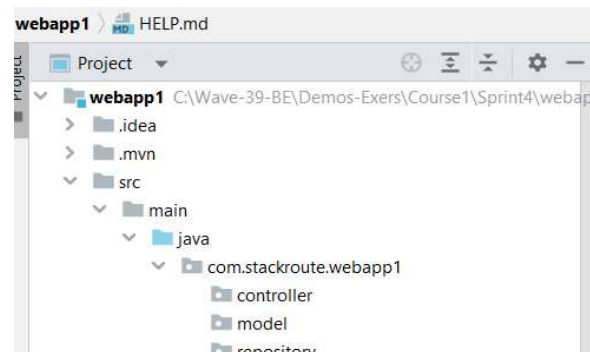
Dependencies ADD DEPENDENCIES... CTRL + B

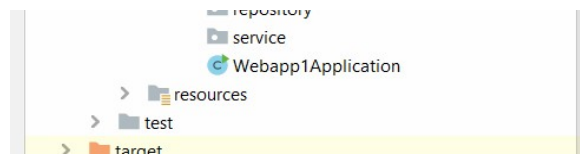
Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

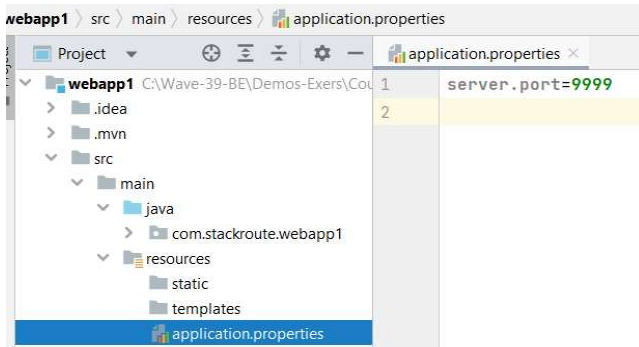
You are screen sharing Stop Share

download, extract, copy to workspace, load in intelliJ
check settings if required
create required packages



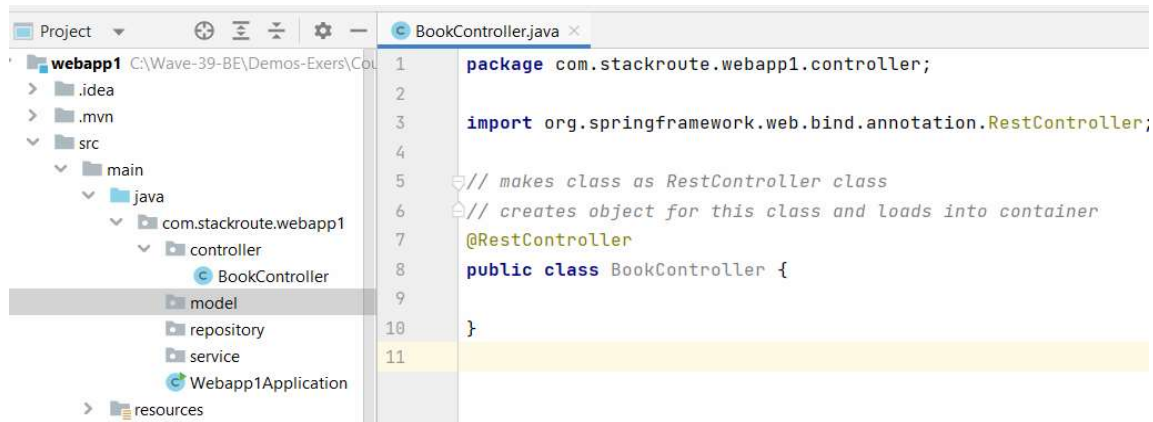


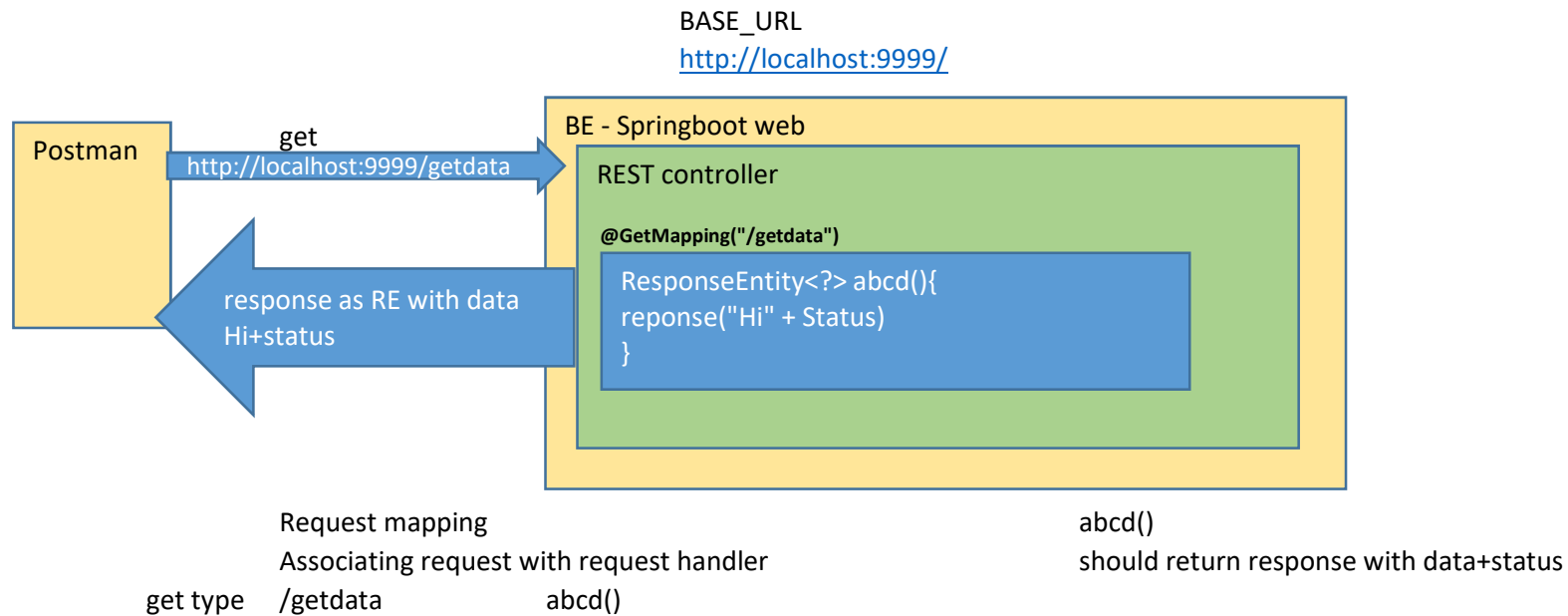
change port number in application.properties



Step 2

Create controller class





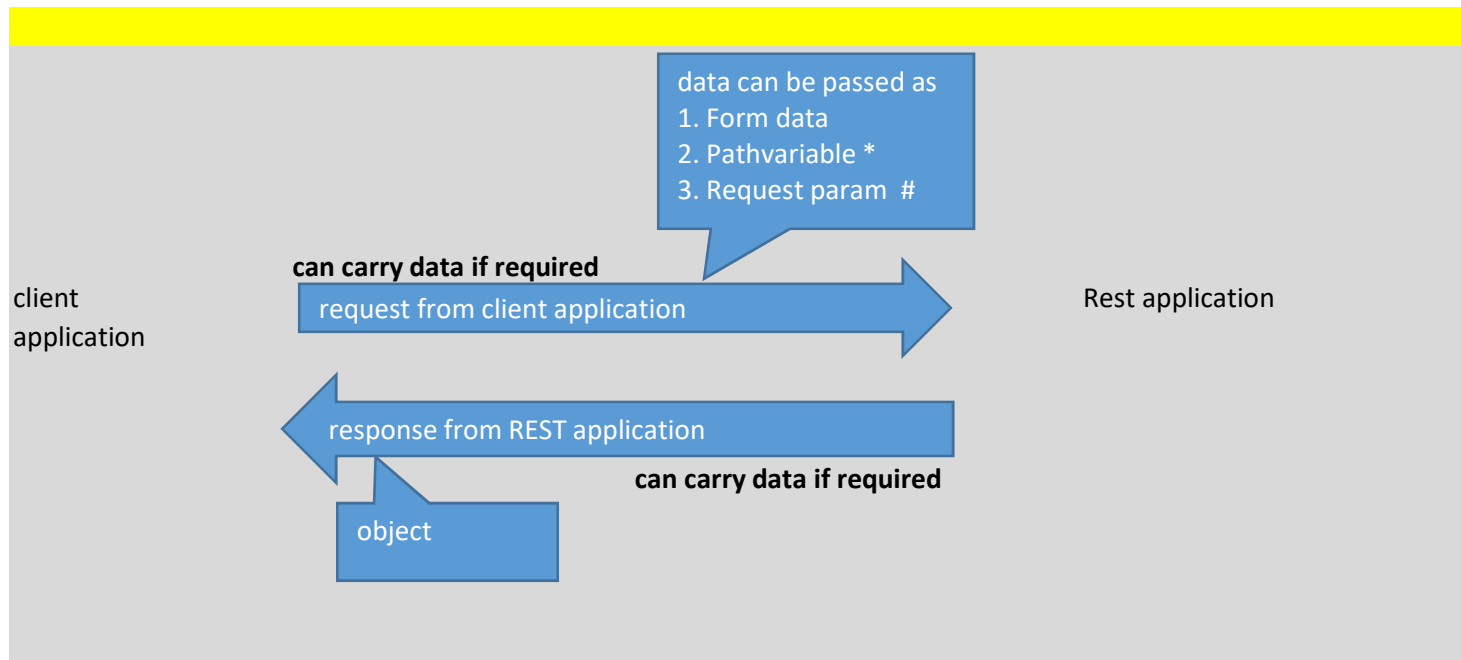
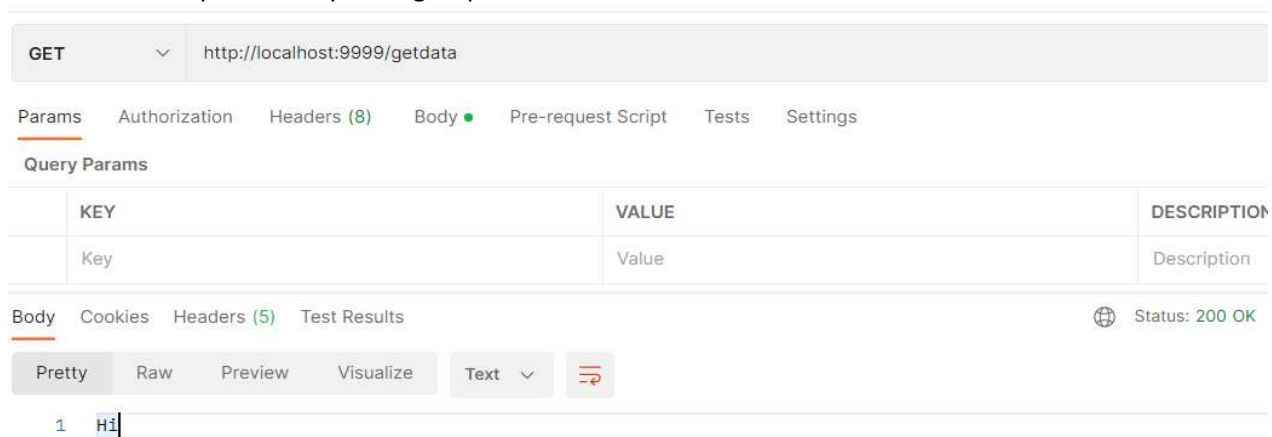
Step 3 Define request handler method in controller

```

8 // makes class as RestController class
9 // creates object for this class and loads into container
10 @RestController
11 public class BookController {
12
13     @GetMapping("/getdata")
14     public ResponseEntity<?> abcd(){
15         return new ResponseEntity<>( body: "Hi", HttpStatus.OK); // data+status
16     }
17
18 }

```

run application
check in postman by raising request



Pathvariable

Data can be passed as part of URL
using '/' as separator

BASE_URL <http://localhost:9999>
BASE_URL/request <http://localhost:9999/calculate>
BASE_URL/request/PV1/PV2 <http://localhost:9999/calculate/20/3>

defining request handler method in controller to receive data which comes along with request

```
23  /* GET
24  http://localhost:9999/getcalc/20/3
25  */
26  @GetMapping("/getcalc/{value1}/{value2}")
27  public ResponseEntity<?> calculate(@PathVariable int value1, @PathVariable int value2){
28      int result=value1+value2;
29      return new ResponseEntity<>(result,HttpStatus.OK);
30  }
```

GET <http://localhost:9999/getcalc/15/17>

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

1 32

getting book objects as response

```
33  /* GET
34  http://localhost:9999/getbooks
35  */
36  @GetMapping("/getbooks")
37  public ResponseEntity<?> getBooks(){
38      Book books[ ] = {
39          new Book( name: "Tags", subject: "HTML", author: "Ravi", price: 2345),
40          new Book( name: "Rest", subject: "Springboot", author: "MC", price: 3454)
41      };
42      return new ResponseEntity<>(books,HttpStatus.OK);
43  }
```

GET http://localhost:9999/getbooks

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK TI

Pretty Raw Preview Visualize JSON

```
1  {
2      {
3          "name": "Tags",
4          "subject": "HTML",
5          "author": "Ravi",
6          "price": 2345
7      },
8      {
9          "name": "Rest",
10         "subject": "Springboot",
11         "author": "MC",
12         "price": 3454
13     }
14 }
```

sending book data as form data

POST `http://localhost:9999/addbook`
carry book object in JSON format

Request handler method has to receive as java object

```
43  /*
44      POST
45      http://localhost:9999/addBook
46  */
47  @PostMapping("/addBook")
48  public ResponseEntity<?> addBook(@RequestBody Book book){
49      System.out.println(book);
50      return new ResponseEntity<>( body: "Data received",HttpStatus.OK);
51  }
```

POST `http://localhost:9999/addBook`

Params Authorization Headers (8) Body Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL ☒ JSON

```
1  {
2    "name": "Let Us C",
3    "subject": "C Language",
4    "author": "BG Swamy",
5    "price": 345
6  }
```

Body Cookies Headers (5) Test Results

Status: 200 OK

Pretty Raw Preview Visualize

Text

1 Data received