**Spring Boot**

**Pre-requisites**

* Java Technology
* Spring Framework & their annotations

**Software requirements**

* Java 17 or later
* Eclipse IDE for Enterprise developers
* VS Code
* Permission to download the maven libraries

Spring Framework

Framework:

It helps developers to create complex applications in a simpler way, framework takes care of providing lot of inbuilt features which every application needs

* Database Access
* Design Patterns
* Exception Handling
* Type Conversion

Spring Framework

It is a Java framework used to develop various types of applications

1. Standalone application
2. Web application
3. Enterprise application
4. Cloud applications

Spring framework provides lot of design patterns which are useful to easily develop java applications

1. Dependency Injection
2. Singleton
3. Prototype
4. Proxy and many more

Dependency Injection: It is a process of supplying an object to another object to make your application loosely coupled

Spring Boot:

It helps you to quickly develop spring applications by providing all the necessary tools & libraries for the type of project you create

ex: For web applications:-

* Project Structure
* Server
* Front Controller
* Build tool ( Gradle or Maven )

First web service program

Software required

* Java 17 or later
* Eclipse IDE / any other Java IDE’s

If you don’t see Maven dependencies after opening the project from eclipse follow these steps

1. Eclipse -> Preferences -> Maven -> Global Checksum -> Warn
2. Project -> Properties -> Project Facets -> Click on configure link
3. Project -> Run As -> Maven Install (wait for downloading process to show success)
4. Project -> Maven -> Update Project -> Force Update -> Ok or Update

Now you must see Maven Dependencies

Spring Framework

There are many modules in Spring Framework to use your project

1. Spring Context: It provides all the core features required for spring like design patterns, dependency injection
2. Spring Web MVC: It provides the features to develop web applications like Front Controller and annotations to create controllers and other web related annotations
3. Spring Boot: It helps to automate the spring configurations for you, so that manual setup can be avoided

Spring Boot:

It is one of the module in spring framework that can automate the configurations for the application based on the type of project you need, it does by providing starter projects, these starter projects will take care of automating the configurations.

Below are widely used starter projects

1. Spring Web Starter: It will auto-configure your application such that you get all the features and setup required to create web applications like embedded tomcat server, design patterns, front-controller configurations
2. Spring Data JPA Starter: It will auto-configure your application to connect to the database & provide connection pool for your application by reading the application.properties file for datasource information
3. Spring Actuator: It will provide endpoints such that you can monitor your application like health, status, memory

Spring Web Starter

You can create webservices using Spring Web Starter

What is a webservice

Webservices are online services which are made available for different applications to exchange the data either in XML or JSON or Text or CSV formats, there are two types of webservices

1. SOAP (Simple Object Access Protocol) – old / legacy webservice it exchanges the data only in XML format
2. ReSTful (Representational State Transfer) – new webservice it exchanges the data in JSON/XML/Text/CSV

Examples

ICICI ATM Machine can debit amount from any banking service

Principles of RESTful Webservice

1. Resource URL: These are the webservices which must have some URL’s so that client programs can access, every webservice must have an URL
2. HTTP protocol: A common protocol both client & webservice can use, so that they will be knowing what operations they are doing, because HTTP provides common methods for client & webservice which are:-
   1. GET – Read/Fetch
   2. POST – Create/Store
   3. PUT – Update/Modify
   4. DELETE – Delete/Remove

Spring Framework provides a library called Spring Web MVC to develop REST webservcies, if you are using Spring Boot then you must use Spring Boot Web Starter that will have Spring Web MVC library & also environment required to develop web application is automated by spring boot, there are annotations which are used to develop REST API’s, below are some of the annotations

1. @RestController
2. @RequestMapping
3. @GetMapping, @PostMapping, @PutMapping, @DeleteMapping
4. @PathVariables
5. @RequestBody

All these annotations are part of Spring Web MVC or Spring Boot Web starter

@RestController: We must write this on top of the class that will have webservices, spring boot creates object of this class that can take the requests from the clients  
@RestController  
class ProfileController { }

@RequestMapping: We must use this to map the request URL regardless of the HTTP method client wants to use

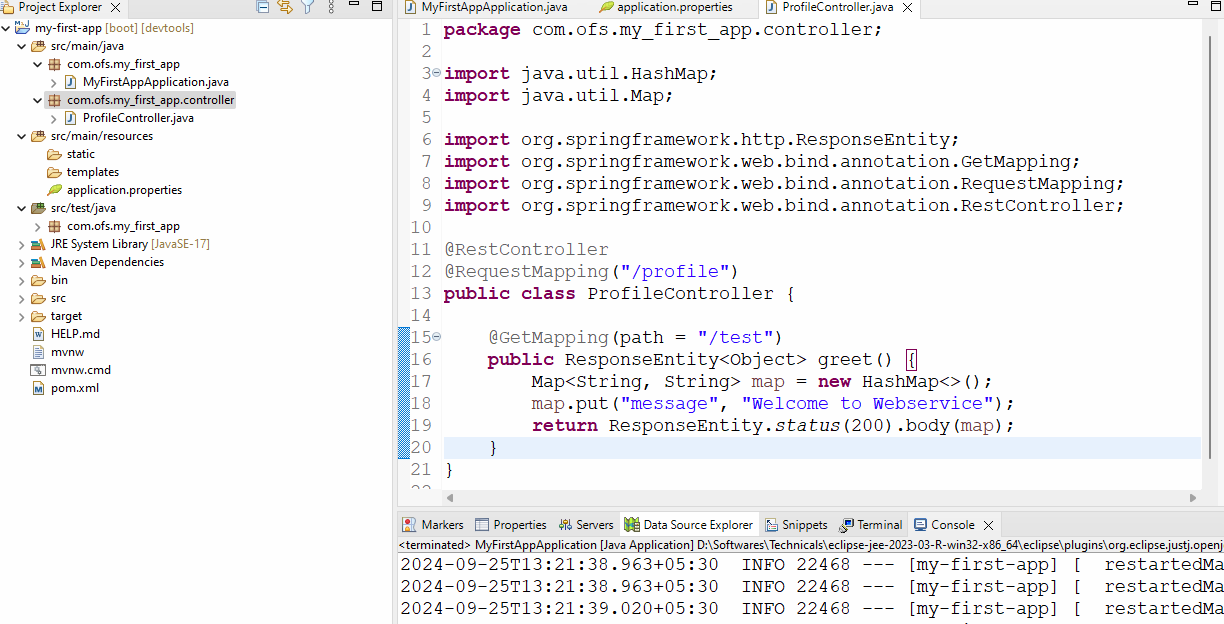
@RestController  
@RequestMapping(“/profile”)  
class ProfileController { }

@GetMapping, @PostMapping, @PutMapping, @DeleteMapping: These are written on top of the methods that will have logics like request processing & response generating logics

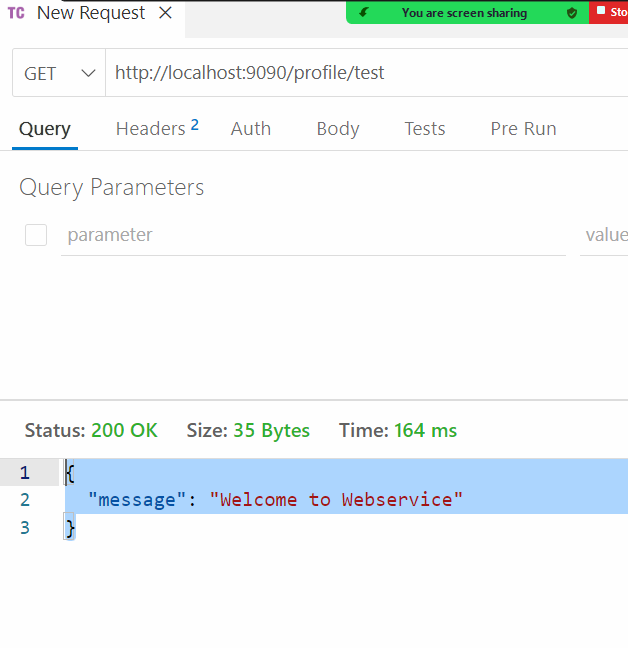
@RestController  
@RequestMapping(“/profile”)  
class ProfileController {   
 @GetMapping(path = “/greet”)  
 public ResponseEntity hello() {   
 return ResponseEntity.status(200).body(“Welcome User”);  
 }  
   
 @GetMapping(path = “/test”)  
 public ResponseEntity hello2() {   
 Map<String, String> map = new HashMap<>();   
 map.put(“key”, “value”); map.put(“key”, “value”)  
 return ResponseEntity.status(201).body(map);  
 }  
}

Note: All the classes must be inside the same package where main class is present or in the subpackage of the main class, because spring boot auto-scans the classes from the main class package to create the objects of the classes

First Webservice program



Output:



Can we create multiple webservices using the same URL but different HTTP methods?

Answer: Yes

Activity:

Create multiple webservice that uses the same URL but different HTTP methods, test them through postman or thunder client

Passing data to the webservice

Clients can pass the data either through URL or Body of the request, to read these data Spring provides two methods

1. @PathVariable: Reads the data from the URL which are called as path parameters
2. @RequestBody: Reads the data from the Body of the request, it lets client to pass complex data structure like JSON/XML and initializes the Java object based on the properties of the complex data

@GetMapping(“/find/{id}”)  
public ResposneEntity fetchProfileById(@PathVariable int id) { …. some logics … }

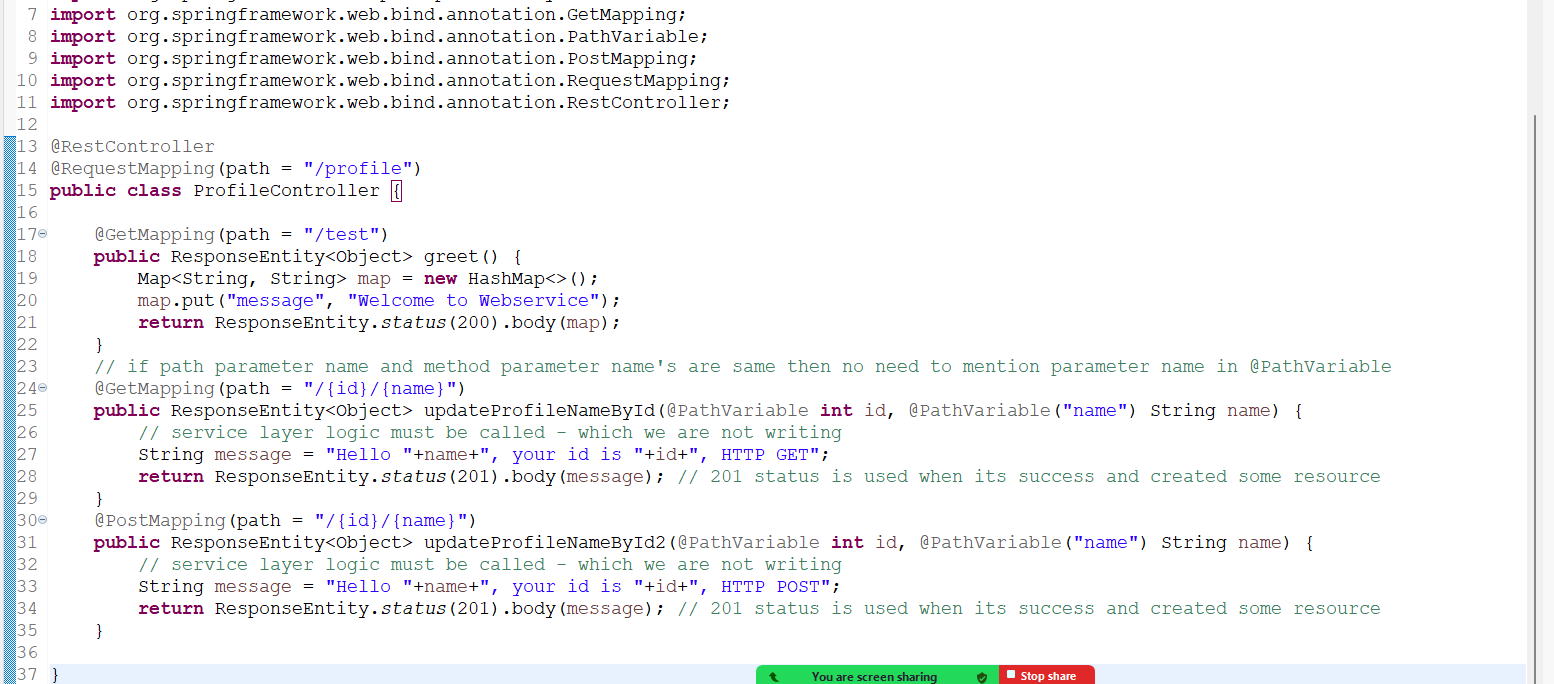
The above webservice matches with /find/anyValue i.e., /find/1, find/2, find/3 and so on

@GetMapping(“/find/{x}”)  
public ResposneEntity fetchProfileById(@PathVariable(“x”) int id) { …. some logics … }

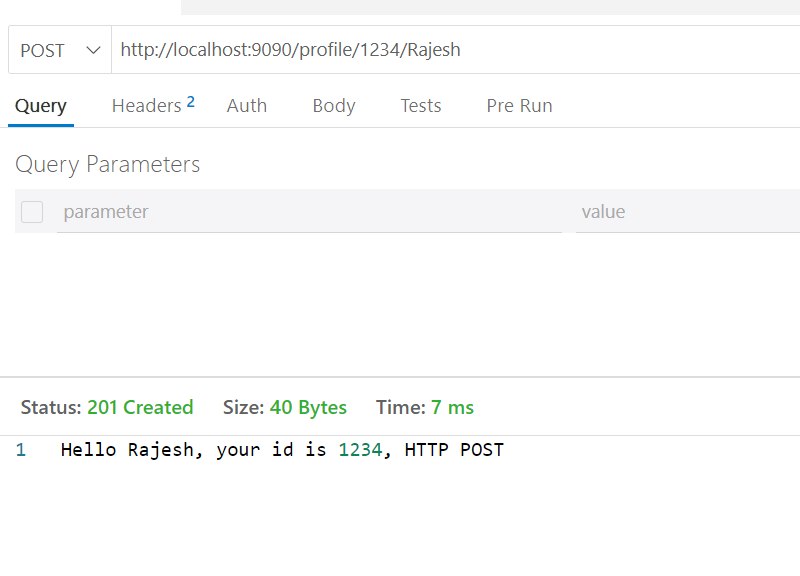
The above webservice matches with /find/anyValue i.e., /find/1, find/2, find/3 and so on

@GetMapping(“/find/{x}/{y}”)  
public ResposneEntity fetchProfileById( @PathVariable(“x”) int id,   
 @PathVariable(“y”) String name) {   
…. some logics … }

The above webservice matches with /find/anyValue i.e., /find/1/Alex, find/2/Brad, find/3/Charles and so on



Output:



@ReqeustBody:

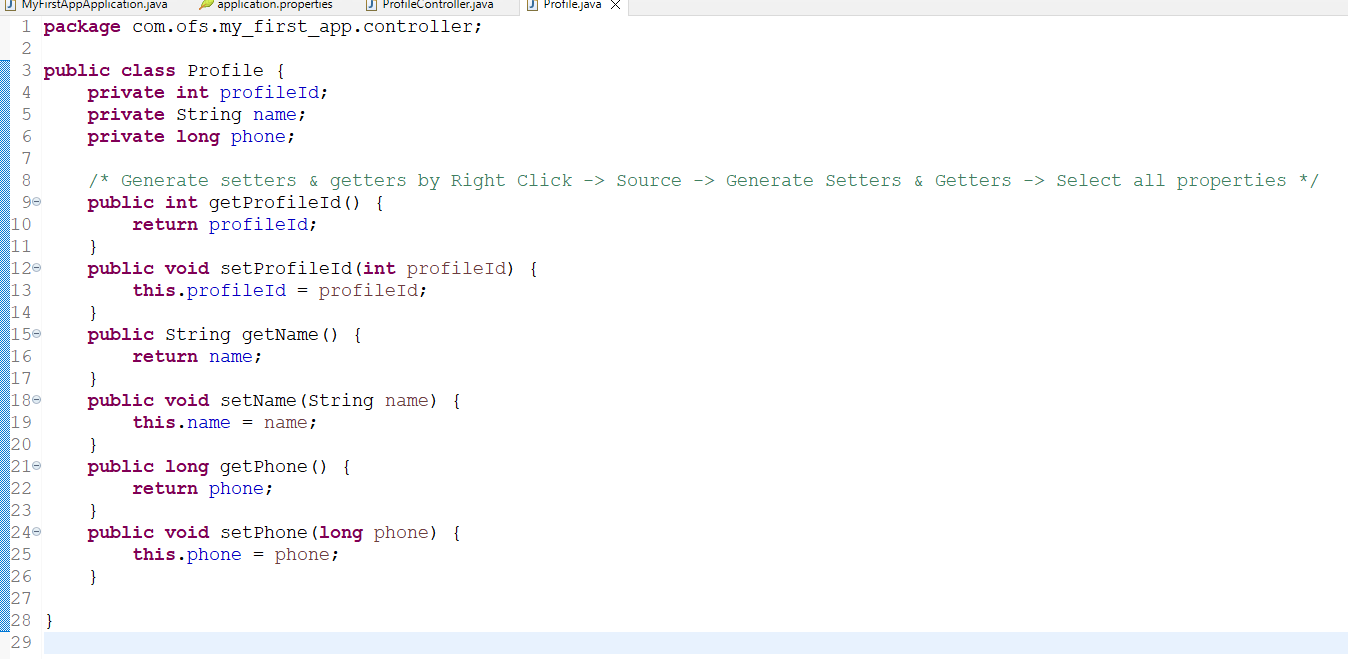
When the client sends a complex data in JSON / XML in a request body you can use @RequestBody that takes care of initializing a java object based on the properties present in the JSON / XML, since client sends a complex data and web service reads the complex data from the body, it is treated as webservice is consuming the data, hence webservice must use consumes = (MediaType.APPLICATION\_JSON\_VALUE) in POST or PUT or DELETE mapping, we can’t send body in the GET request

@PostMapping(path = “/store”, consumes = MediaType.APPLICATION\_JSON\_VALUE)  
public ResponeEntity save(@RequestBody Profile obj) { … }

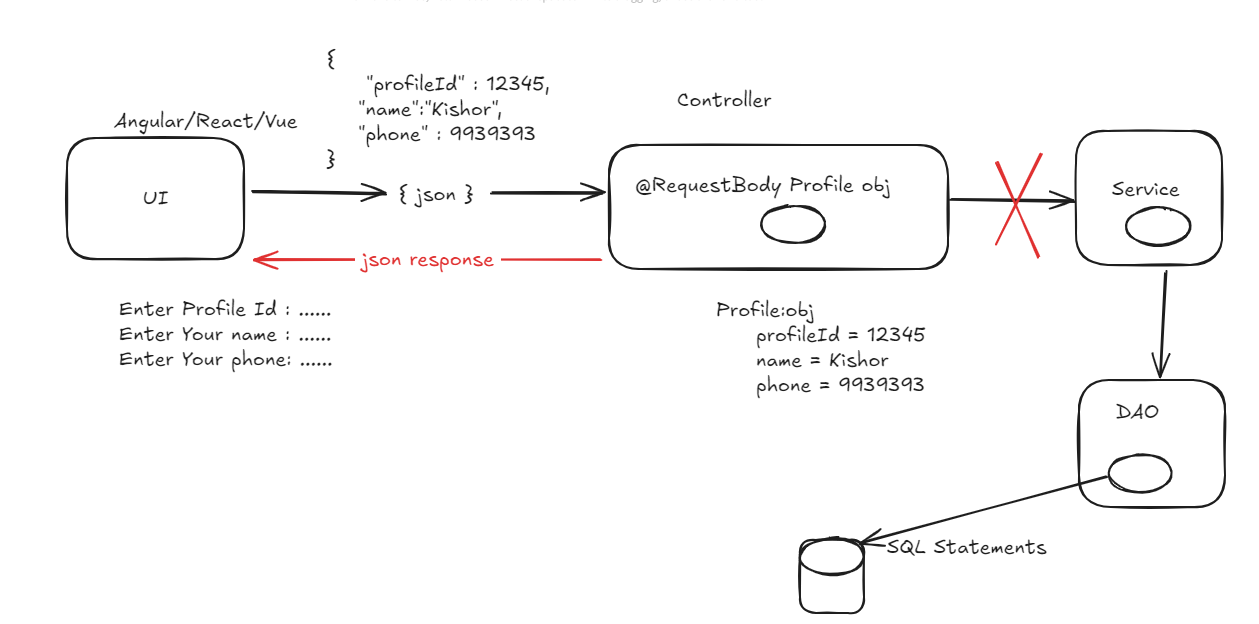
class Profile { profileId, name, phone }

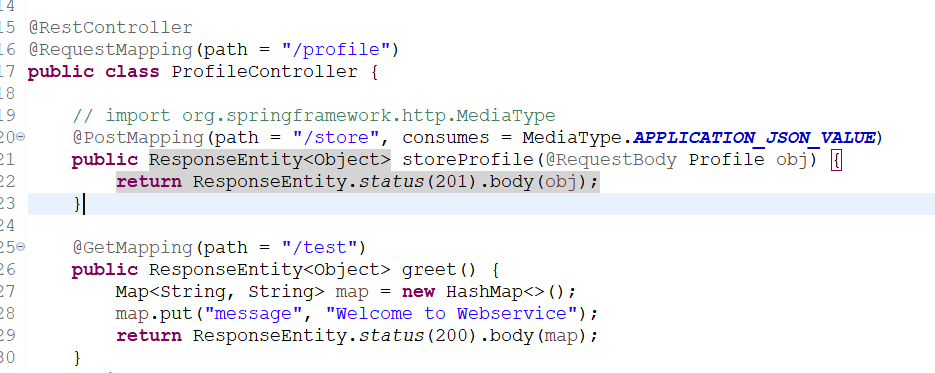
JSON data must have properties like profileId, name, phone to initialize the Profile object in the webservice in case the property name is different we must use some annotations to specify which json property matches to java property  
  
{  
 “profileId” : 123465, “name”: “Alex”, “phone” : 99392939   
}  
UI  
 profile id : Input  
 name : Input  
 phone : Input

Profile.java

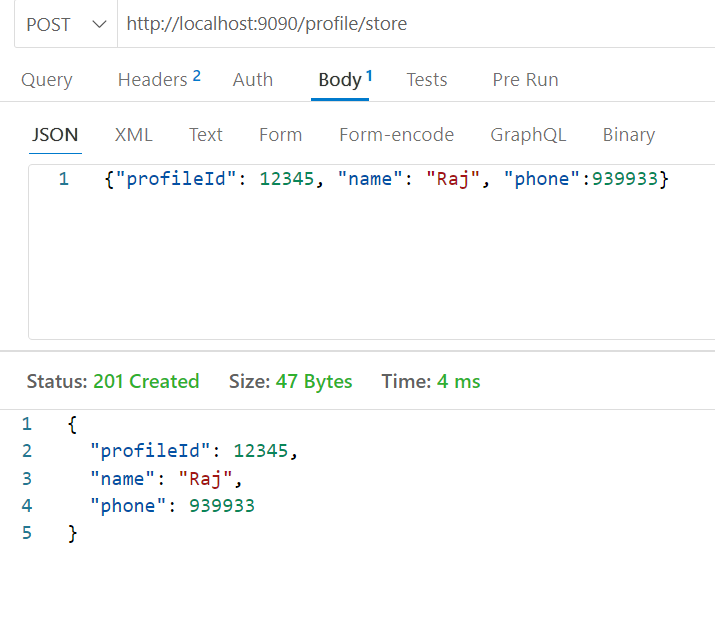


Usually controller needs to call the Service, but we are not using a service layer as of now, instead we will generate the response without accessing the service layer





Output:



MVC architecture

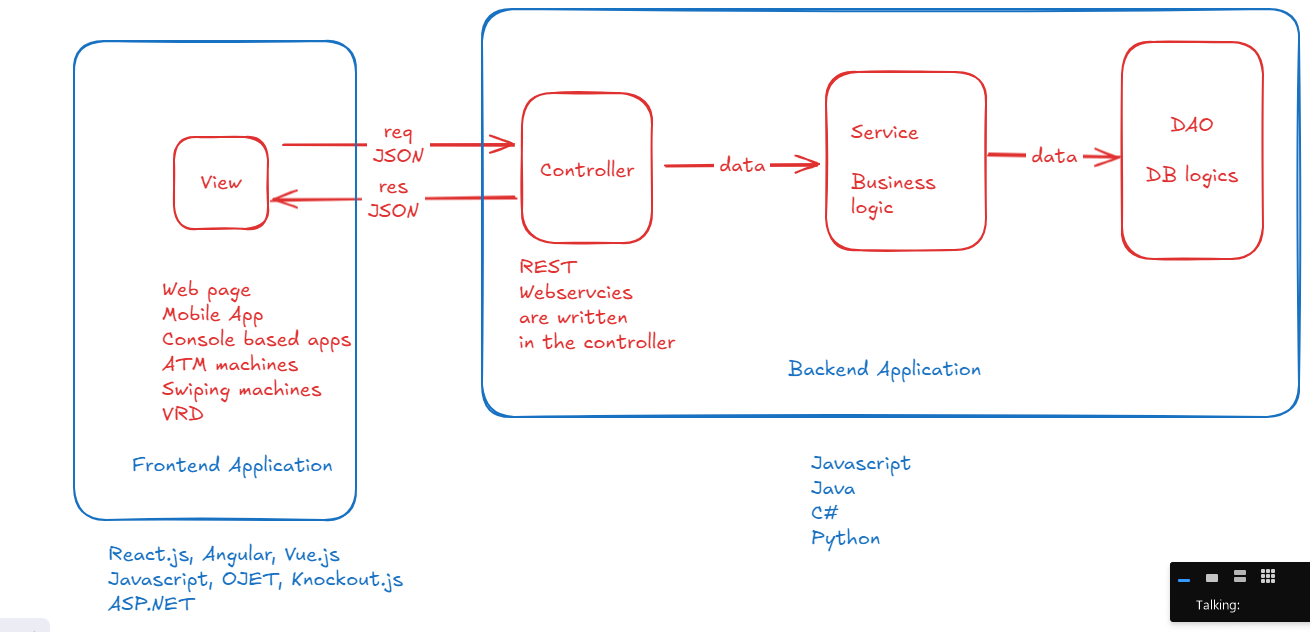
It separates the logics of the application into Model, View & Controller

View: Presentation logic -> Send Request to the application & Gets Response from the application

Controller: Request processing logic -> Access Service/Business logic -> Generate the response to the View

Model: It will have Business logic & DB logic, it is divided into two parts

1. Service: Business logic
2. DAO: DB logic



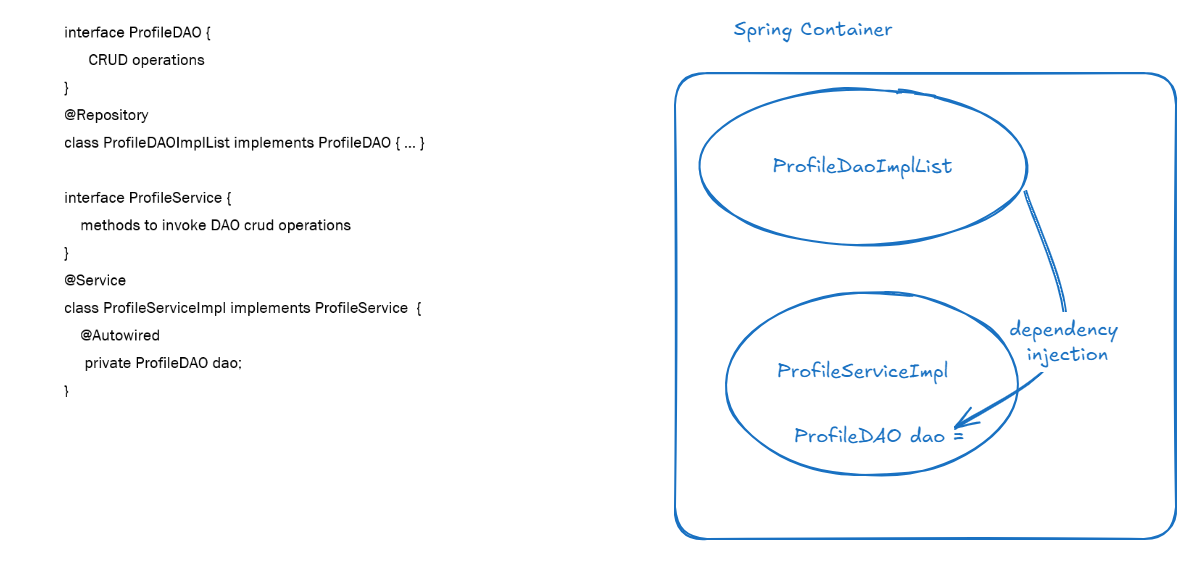
Controller depends on the service layer

Service layer depends on the dao layer

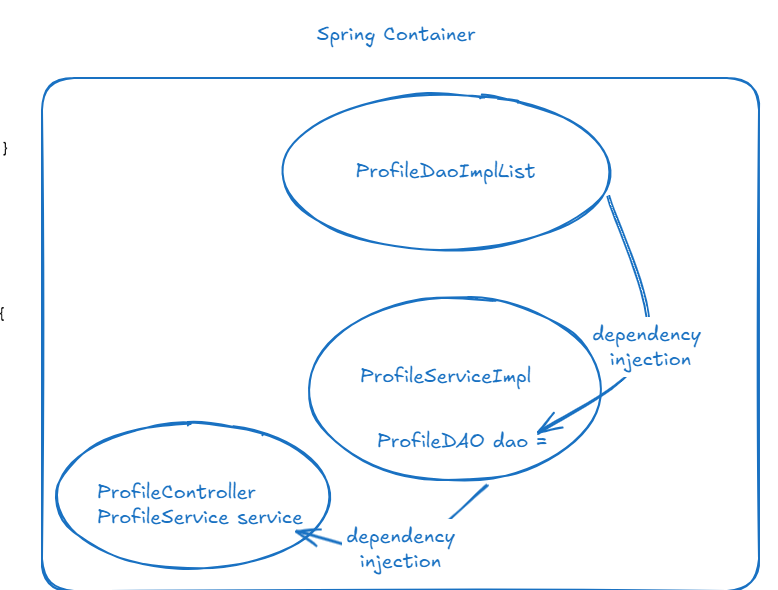
Things to do

1. Creating a Service layer to have some business logics
2. Creating a DAO layer to maintain the data in some temporary memory using collections (instead of DB)
3. Controller depends on the service
4. Service depends on the DAO
5. Spring Framework creates an object of Service & Injects to the controller, & also creates an object of DAO & injects to the service

interface ProfileDAO {   
 CRUD operations  
}  
@Repository  
class ProfileDAOImplList implements ProfileDAO { … }  
  
interface ProfileService {   
 methods to invoke DAO crud operations  
}  
@Service  
class ProfileServiceImpl implements ProfileService {   
 @Autowired  
 private ProfileDAO dao;  
}



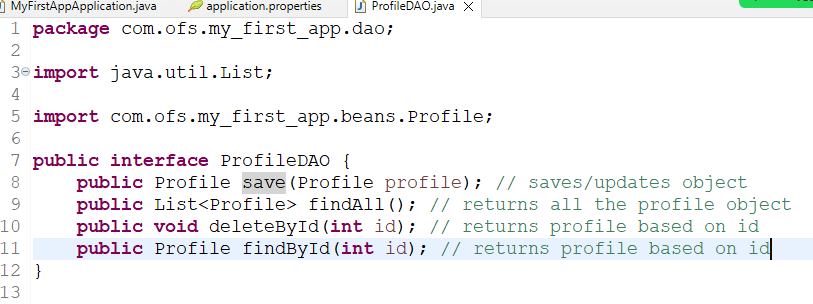
@RestController  
@ReeustMapping  
class ProfileController {  
 @Autowired  
 private ProfileService service; // ProfileServiceImpl is injected  
}



List of interfaces & classes we must create

1. ProfileDAO interface
2. ProfileService interface
3. ProfileDAOListImpl that implements ProfileDAO
4. ProfileServiceImpl that implements ProfileService

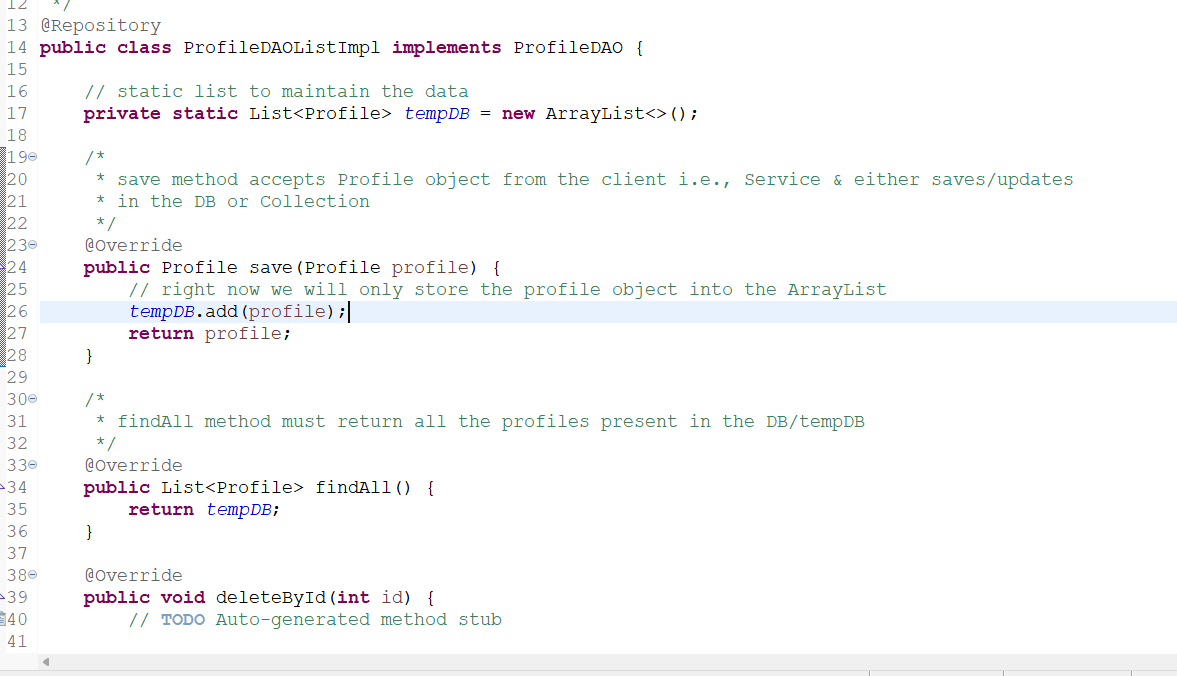
ProfileDao.java



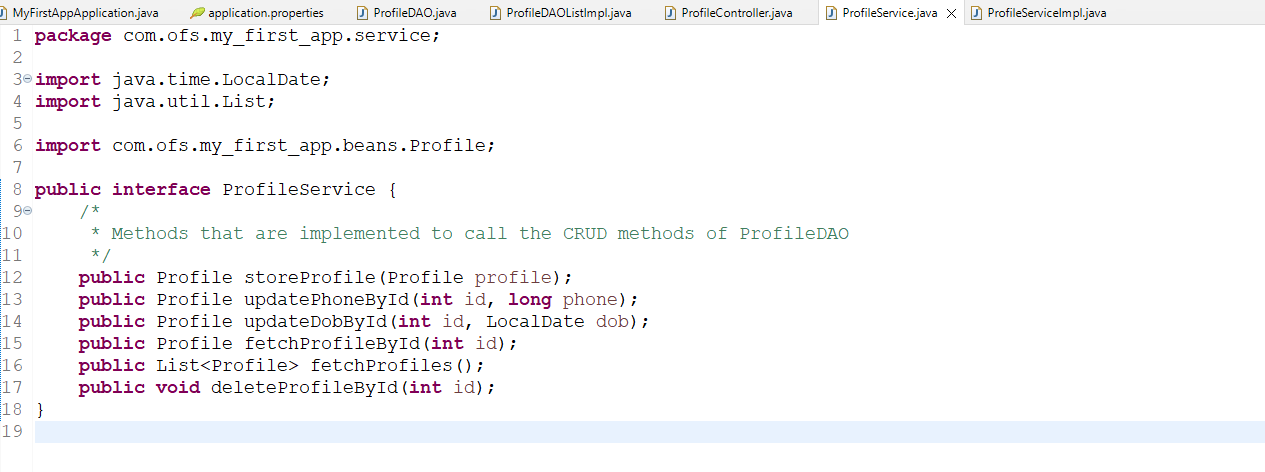
ProfileDAOListImpl.java

Note: @Repository helps spring to create its object in the spring container so that it can be injected to their dependencies

Similar to @Repository there are other annotations in Spring Boot that helps spring to create object in the container which are @RestController, @Service, @Component and etc.



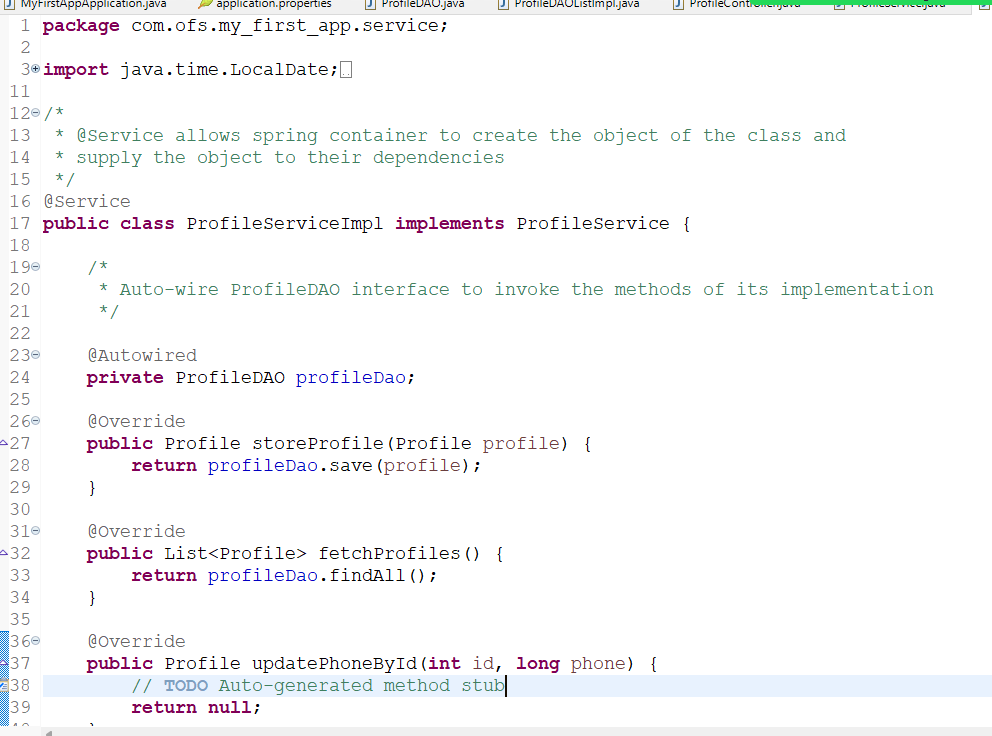
ProfileService.java



ProfileServiceImpl.java

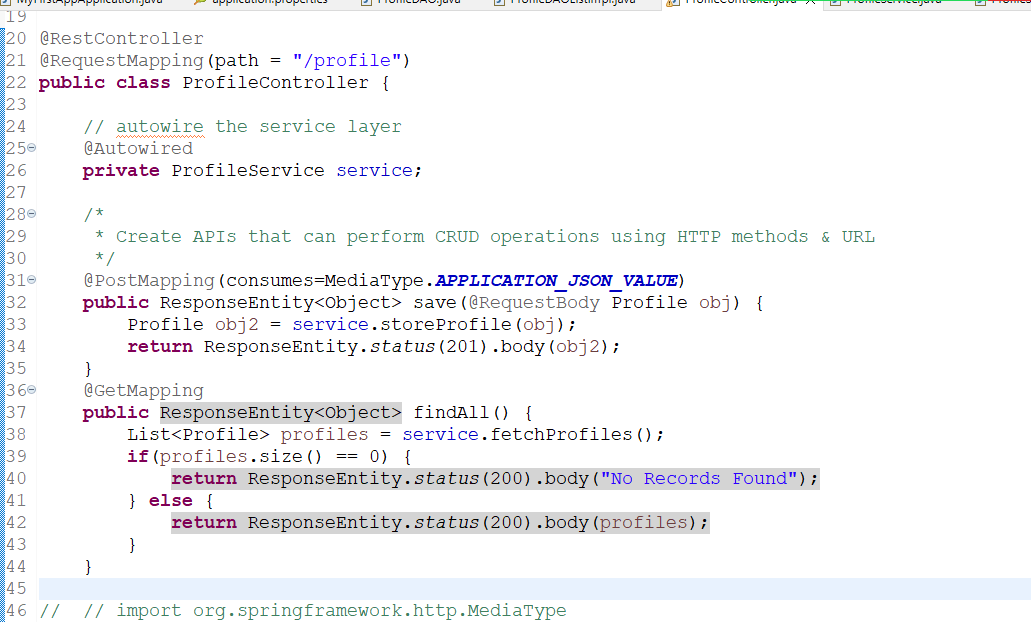
Note: @Service helps spring to create its object in the spring container so that it can be injected to their dependencies

Service layer depends on DAO layer, hence you must use @Autowired on DAO interface



Now Controller must access Service layer methods

Use @Autowired in the Controller to inject the Service layer object and create methods that invokes the service layer methods with appropriate HTTP mapping.

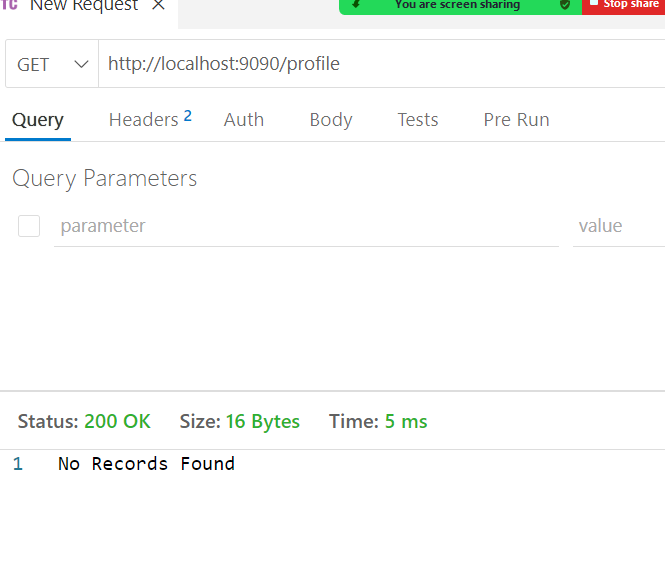


Note: DOB is of LocalDate type and Profile class has dob property, in JSON we must pass dob in ISO format i.e., yyyy-MM-dd

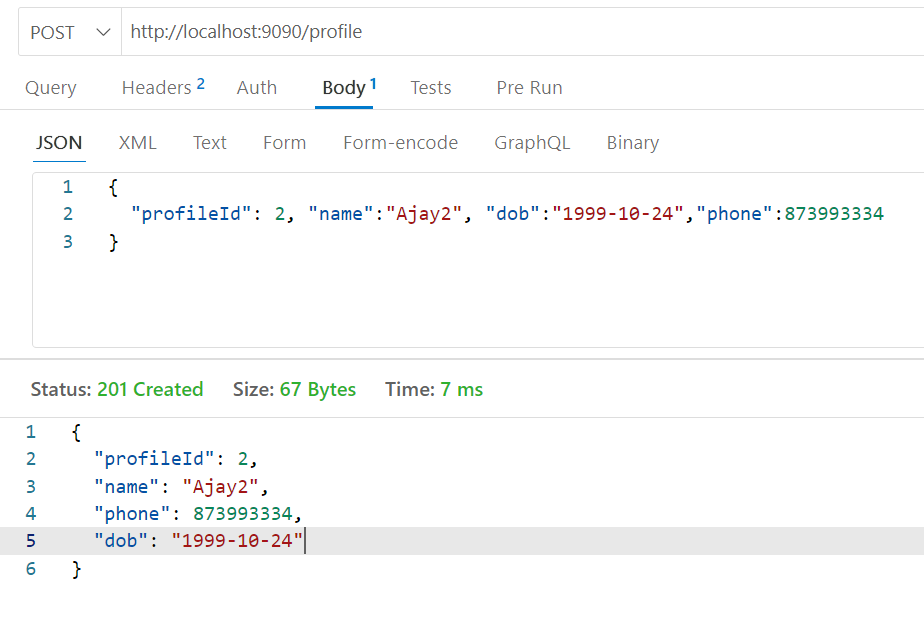
{  
 “profileId”:1, “name”:”Alex”, “phone”:9399393, “dob”: “2000-09-15”   
}

Send two or more requests to POST & also send GET requests as well and observe the output

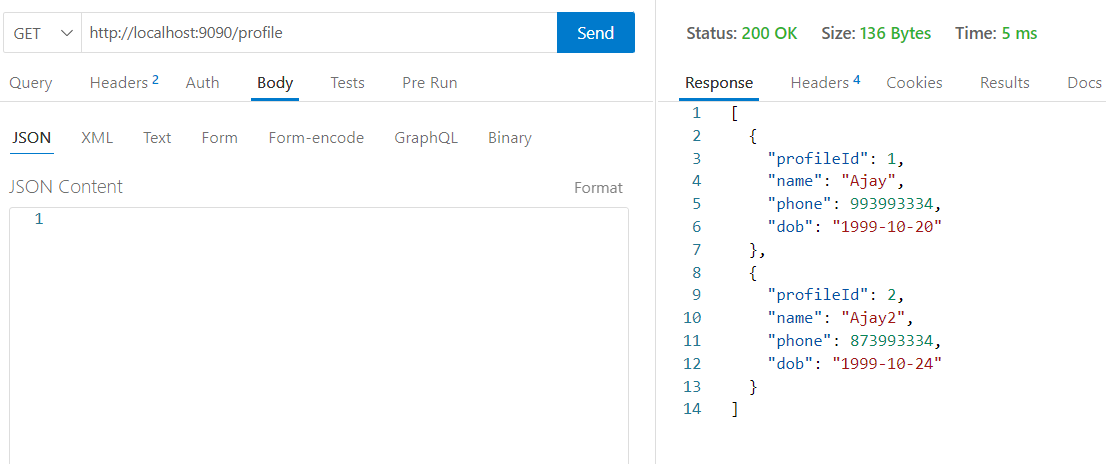
Output when List is empty



Adding some data to the list



Output for GET



Note: Once the application reloads all the data will be lost because we are not storing in Database, we are storing in Collection

Activity: Implement all the methods of ProfileDAO where it can perform the CRUD operations on the ArrayList, Implement the ProfileDAO methods as per below instructions

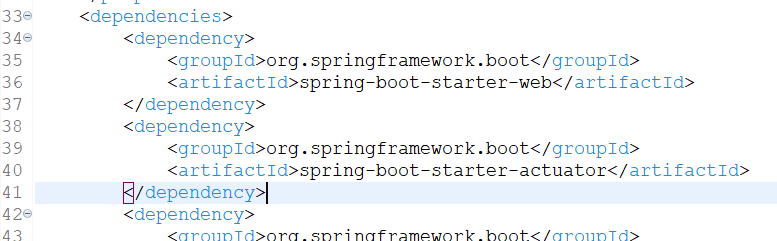
1. the save() method of ProfileDAO is just adding the profile object, it must be modified to add the profile object if the id is not present already or update the entire profile object if the id is already present
2. the deleteById() method of ProfileDAO must delete the profile object that matches to the profile id present in the ArrayList
3. the findById() method of ProfileDAO must return the profile if id is found in the ArrayList else return null
4. Modify the ProfileService interface that throws ProfileNotFoundException(create a custom exception) in the following methods
   * updatePhoneById, updateDobById, fetchProfileById, deleteProfileById
   * i.e., public void deleteProfileById() throws ProfileNotFoundException;
5. Modify the ProfileServiceImpl so that if profile id is not found it must throw ProfileNotFoundException to its caller
   * i.e, caller(ProfileController) will use try & catch to handle the ProfileNotFoundException while invoking the above methods of the ProfileService
6. ProfileController must invoke all the methods of ProfileService to perform CRUD operations & generate either success or error Response in JSON format for all the methods that throw exceptions i.e., any method throws a ProfileNotFoundException

ex: { “message”: “Profile with an id 25 not found” } : this if profile id is not found

ex: { “profileId”:1, “name”:”Raj”, “dob”:”2000-10-25”, phone:8888}: this if profile is found

Actuator:

It is mainly used to monitor the applications, it provides end points to check the application health and other application related features



All the application related details Operations team can view using application-path/actuator end point

