**Assisted Practice: 4.2 Microservices Communication in Spring Boot**

This section will guide you to:

* Build Microservices Communication with Spring Boot

This lab has eighteen sub-sections, namely:

4.2.1 Opening Spring Tool Suite and creating a new project using Spring Initializer

4.2.2 Selecting the required project dependencies

4.2.3 Creating an Entity Class

4.2.4 Creating a Repository Class

4.2.5 Creating a Service Class

4.2.6 Creating a Controller Class

4.2.7 Creating a Response Class

4.2.8 Setting the port number for the project in the application properties

4.2.9 Executing the project as ‘Spring Boot App’

4.2.10 Creating another project

4.2.11 Creating an Entity Class

4.2.12 Creating a Repository Class

4.2.13 Creating a Service Class

4.2.14 Creating a Controller Class

4.2.15 Setting the port number for the project in application properties

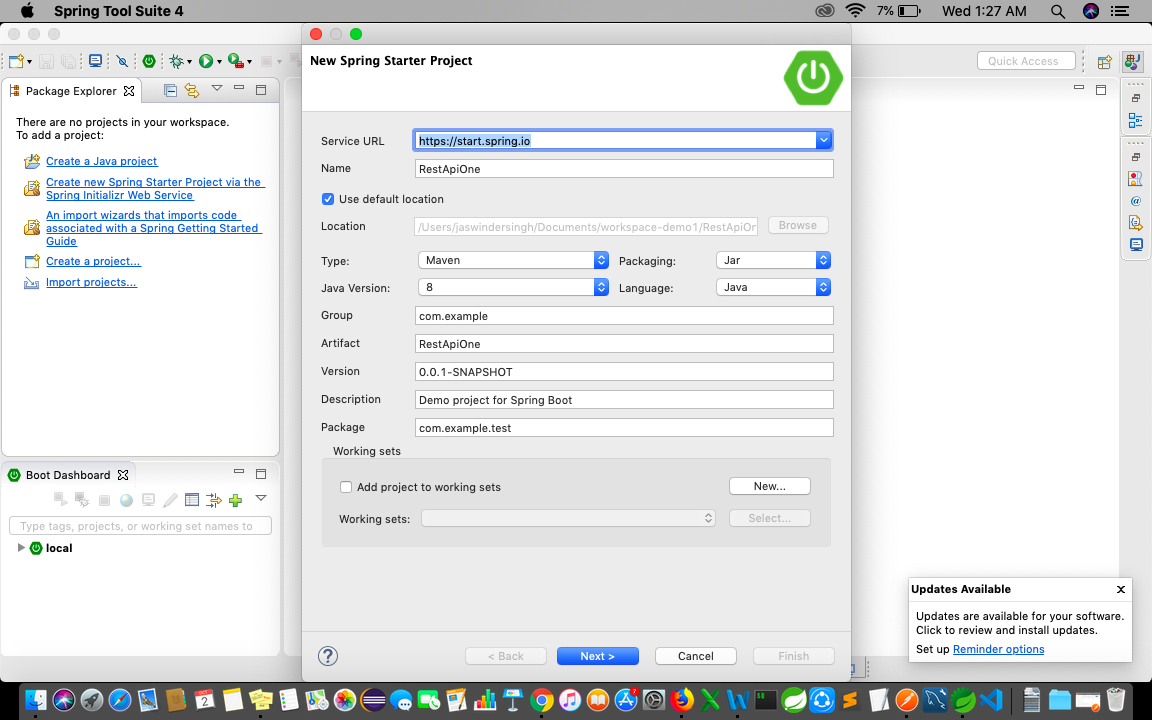
4.2.16 Executing the project ‘Spring Boot App’

4.2.17 Making a GET request for ‘RestApiOne’ using Postman. It will be fetching the data from the existing running service ‘RestApiTwo’ and will be showing the communication of two microservices.

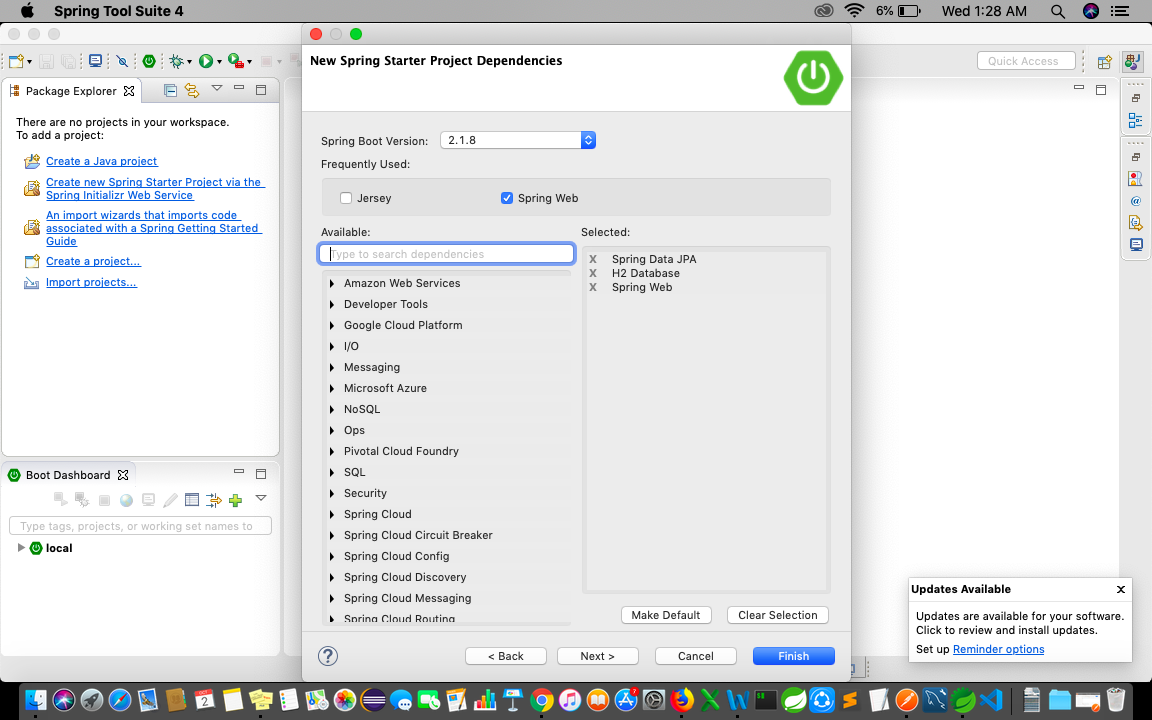
4.2.18 Pushing the code to your GitHub repositories

**Step 4.2.1:** Opening Spring Tool Suite and creating a new project using Spring Initializer

* Fill in the required fields and call it ‘RestApiOne’.

****

**Step 4.2.2:** Selecting the required project dependencies Spring Web, H2 Database, and Spring Data JPA. Now, click on Finish.

****

* It will automatically create the main class.

package com.example.test;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class RestApiOneApplication {

public static void main(String[] args) {

SpringApplication.run(RestApiOneApplication.class, args);

}

}

**Step 4.2.3:** Creating an Entity Class

package com.example.test;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class PersonEntity {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@Column(name = "id", updatable = false, nullable = false)

private Integer personId;

@Column

private String name;

@Column

private Integer age;

public PersonEntity() {

super();

}

public PersonEntity(Integer personId, String name, Integer age) {

super();

this.personId = personId;

this.name = name;

this.age = age;

}

public Integer getPersonId() {

return personId;

}

public void setPersonId(Integer personId) {

this.personId = personId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Integer getAge() {

return age;

}

public void setAge(Integer age) {

this.age = age;

}

}

**Step 4.2.4:** Creating a Repository Class

package com.example.test;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface PersonRepository extends JpaRepository<PersonEntity, Integer> {

}

**Step 4.2.5:** Creating a Service Class

package com.example.test;

import java.util.HashMap;

import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.web.client.RestTemplate;

@Service

public class PersonService {

@Autowired

PersonRepository personRepository;

RestTemplate restTemplate = new RestTemplate();

public PersonResonse getPerson(int personId){

final String uri = "http://localhost:8082/webapitwo/hobby/{personId}";

Map<String, Integer> params = new HashMap<String, Integer>();

params.put("personId", personId);

String result = restTemplate.getForObject(uri, String.class, params);

PersonEntity pe=personRepository.findById(personId).get();

PersonResonse pr=new PersonResonse();

pr.setPersonId(pe.getPersonId());

pr.setName(pe.getName());

pr.setAge(pe.getAge());

pr.setHobby(result);

return pr;

}

public void addPerson(PersonEntity pe){

personRepository.save(pe);

}

}

**Step 4.2.6:** Creating a Controller Class

package com.example.test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestBody;

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

import org.springframework.web.bind.annotation.RequestMethod;

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

@RestController

@RequestMapping(path = "/webapione")

public class PersonControlller {

@Autowired

PersonService personService;

@RequestMapping("/person/{personId}")

public PersonResonse getPerson(@PathVariable int personId){

return personService.getPerson(personId);

}

@RequestMapping(method=RequestMethod.POST, value="/person")

public void addPerson(@RequestBody PersonEntity pe ) {

personService.addPerson(pe);

}

}

**Step 4.2.7**: Creating aResponse Class

package com.example.test;

public class PersonResonse {

private Integer personId;

private String name;

private Integer age;

private String hobby;

public Integer getPersonId() {

return personId;

}

public void setPersonId(Integer personId) {

this.personId = personId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Integer getAge() {

return age;

}

public void setAge(Integer age) {

this.age = age;

}

public String getHobby() {

return hobby;

}

public void setHobby(String result) {

this.hobby = result;

}

}

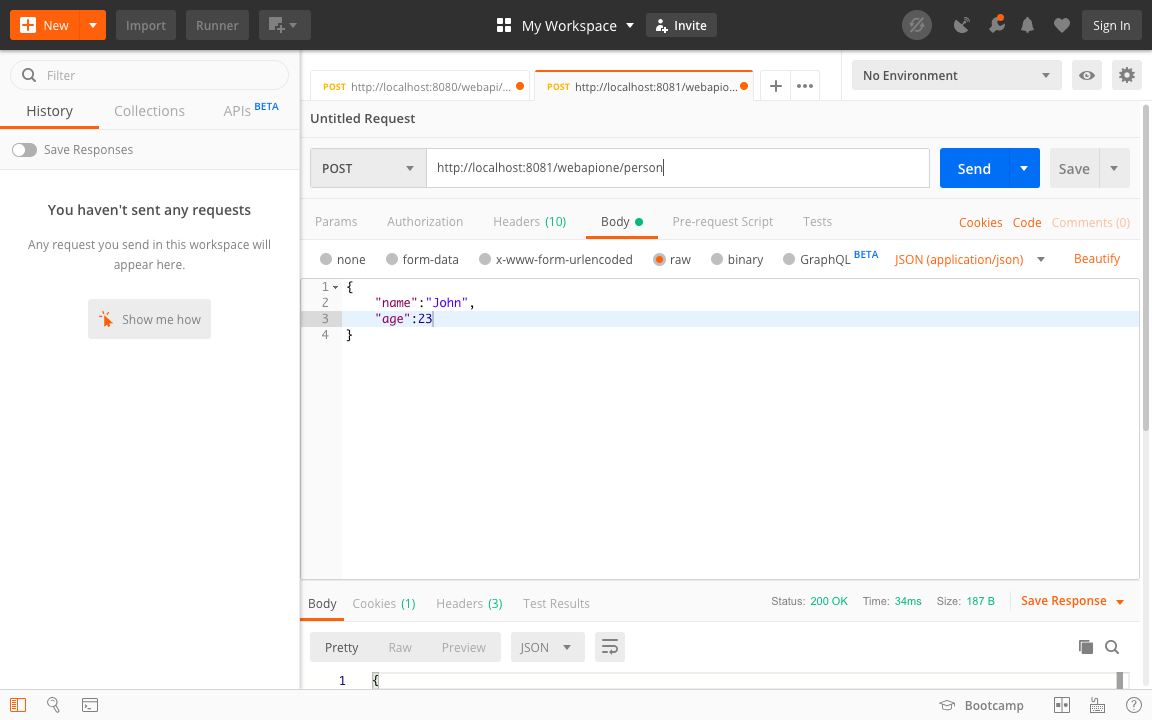
**Step 4.2.8**: Setting the port number for the project in the application properties

server.port=8081

spring.application.name=RestApiOne

**Step 4.2.9:** Executing the project as ‘Spring Boot App’

* It will run on port:8081 and make a POST request using POSTMAN

****

**Step 4.2.10:** Creating another project with the name ‘RestApiTwo’ as discussed in step 4.21

* It will automatically create the main method:

package com.example.test;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class RestApiTwoApplication {

public static void main(String[] args) {

SpringApplication.run(RestApiTwoApplication.class, args);

}

}

**Step 4.2.11:** Creating an Entity Class

**package com.example.test;**

**import javax.persistence.Column;**

**import javax.persistence.Entity;**

**import javax.persistence.GeneratedValue;**

**import javax.persistence.GenerationType;**

**import javax.persistence.Id;**

**@Entity**

**public class HobbyEntity {**

**@Id**

**@GeneratedValue(strategy = GenerationType.AUTO)**

**@Column(name = "id", updatable = false, nullable = false)**

**private Integer id;**

**@Column**

**private Integer personId;**

**@Column**

**private String name;**

**public HobbyEntity() {**

**super();**

**}**

**public HobbyEntity(Integer personId, String name) {**

**super();**

**this.personId = personId;**

**this.name = name;**

**}**

**public Integer getPersonId() {**

**return personId;**

**}**

**public void setPersonId(Integer personId) {**

**this.personId = personId;**

**}**

**public String getName() {**

**return name;**

**}**

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

**this.name = name;**

**}**

**}**

**Step 4.2.12:** Creating a Repository Class

package com.example.test;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.stereotype.Repository;

@Repository

public interface HobbyRepository extends JpaRepository<HobbyEntity, Integer> {

@Query("SELECT h.name FROM HobbyEntity h WHERE h.personId=:personId")

public String findByPersonId(Integer personId);

}

**Step 4.2.13:** Creating a Service Class

package com.example.test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class HobbyService {

@Autowired

HobbyRepository hobbyRepository;

public String findByPersonId(int personid){

return hobbyRepository.findByPersonId(personid);

}

public void addHobby(HobbyEntity he){

hobbyRepository.save(he);

}

}

**Step 4.2.14**: Creating a Controller Class

package com.example.test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestBody;

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

import org.springframework.web.bind.annotation.RequestMethod;

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

@RestController

@RequestMapping(path = "/webapitwo")

public class HobbyController {

@Autowired

HobbyService hobbyService;

@RequestMapping("/hobby/{personid}")

public String findByPersonId(@PathVariable int personid){

return hobbyService.findByPersonId(personid);

}

@RequestMapping(method=RequestMethod.POST, value="/hobby")

public void addHobby(@RequestBody HobbyEntity he ) {

hobbyService.addHobby(he);

}

}

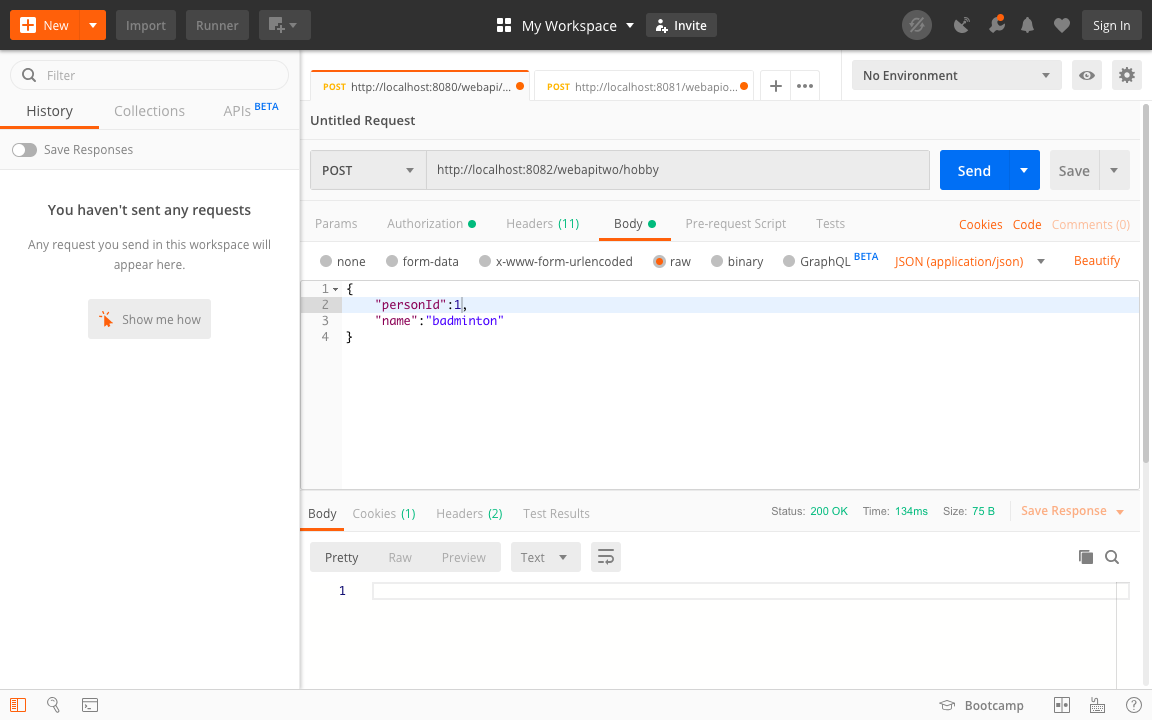
**Step 4.2.15**: Setting the port number for the project in application properties

server.port=8082

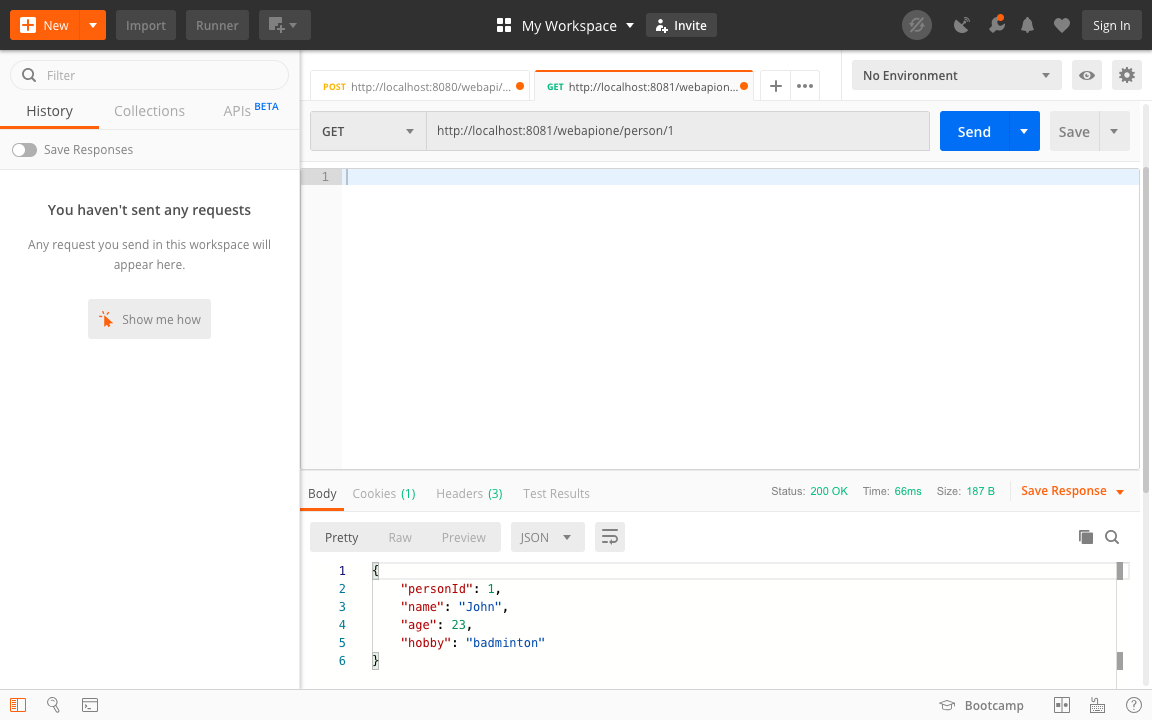
spring.application.name=RestApiTwo

**Step 4.2.16**: Executing the project ‘Spring Boot App’

* It will run on port:8082 and making a POST request using Postman

****

**Step 4.2.17:** Making a GET request of ‘RestApiOne’ using Postman, it will be fetching the data from the existing running service ‘RestApiTwo’ and will be showing the communication of two microservices

****

**Step 4..2.18:** Pushing the code to your GitHub repositories

* Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

* Initialize your repository using the following command:

**git init**

* Add all the files to your git repository using the following command:

**git add .**

* Commit the changes using the following command:

**git commit . -m “Changes have been committed.”**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**