Microservices Communication in Spring Boot

Step 1: 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 2: 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 3: 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: 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 5: Creating a Response 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 6: Setting the port number for the project in the application properties

```
server.port=8081
spring.application.name=RestApiOne
```

Step 7: Executing the project as ‘Spring Boot App’

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

Step 8: Creating another project with the name ‘RestApiTwo’

- 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 9: 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 10: 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 11: 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 12: 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 13: Setting the port number for the project in application properties

```
server.port=8082
```

```
spring.application.name=RestApiTwo
```