<https://github.com/eazybytes/microservices/tree/3.2.3>

Step – 1

package com.example.hello;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HelloApplication {

    public static void main(String[] args) {

        SpringApplication.run(HelloApplication.class, args);

    }

}

package com.example.hello;

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

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

@RestController

public class HelloController {

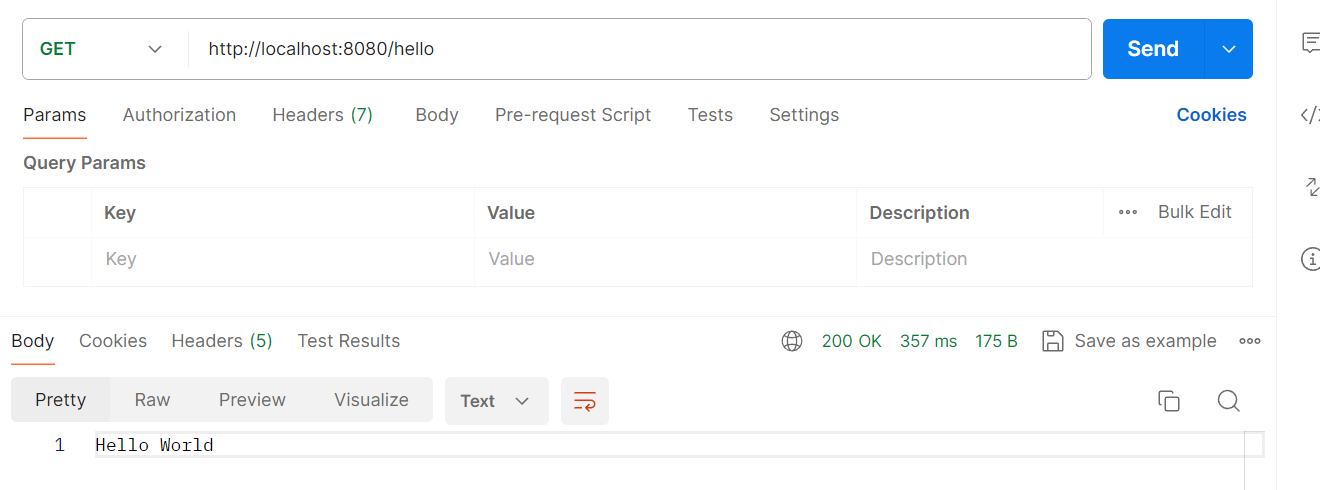
@GetMapping("/hello")

public String helloWorld() {

    return "Hello World";

}

}



Step - 2 Spring data JPA entities and repository classes to interact with the tables  
Configuring H2 DB  
Standard template for H2 database configuration for all services

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.h2.console.enabled=true

package com.eazybytes.accounts.model;

@MappedSuperclass

@Setter @Getter @ToString

public class BaseEntity {

    @Column(updatable = false)

    private LocalDateTime createdAt;

    @Column(insertable = false)

    private String createdBy;

    @Column(updatable = false)

    private LocalDateTime updatedAt;

    @Column(insertable = false)

    private String updatedBy;

}

package com.eazybytes.accounts.model;

@Entity

@Getter @Setter @ToString @AllArgsConstructor @NoArgsConstructor

public class Customer extends BaseEntity{

    @Column(name = "customer\_id")

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int customerId;

    @Column(name = "name")

    private String name;

    @Column(name = "email")

    private String email;

    @Column(name = "mobile\_number")

    private String mobileNumber;

}

package com.eazybytes.accounts.model;

@Entity

@Getter @Setter @ToString @AllArgsConstructor @NoArgsConstructor

public class Accounts extends BaseEntity{

    @Column(name = "account\_number")

    @Id

    private long accountNumber;

    @Column(name = "customer\_id")

    private int customerId;

    @Column(name = "account\_type")

    private String accountType;

    @Column(name = "branch\_address")

    private String branchAddress;

}

package com.eazybytes.accounts.repository;

@Repository

public interface AccountsRepository extends CrudRepository<Accounts, Long>{

}

package com.eazybytes.accounts.repository;

public interface CustomerRepository extends JpaRepository<Customer, Integer> {

   // There are many method in this interface

}

Step 3 – Create DTO classes

package com.eazybytes.accounts.dto;

@Data

public class AccountsDto {

    private long accountNumber;

    private String accountType;

    private String branchAddress;

}

package com.eazybytes.accounts.dto;

@Data

public class CustomerDto {

    private String name;

    private String email;

    private String mobileNumber;

}

package com.eazybytes.accounts.dto;

@Data @AllArgsConstructor

public class ResponseDto {

    private String statusCode;

    private String statusMessage;

}

package com.eazybytes.accounts.dto;

@Data @AllArgsConstructor

public class ErrorResponseDto {

    private String apiPath;

    private HttpStatus errorCode;

    private String errorMessage;

    private LocalDateTime errorTime;

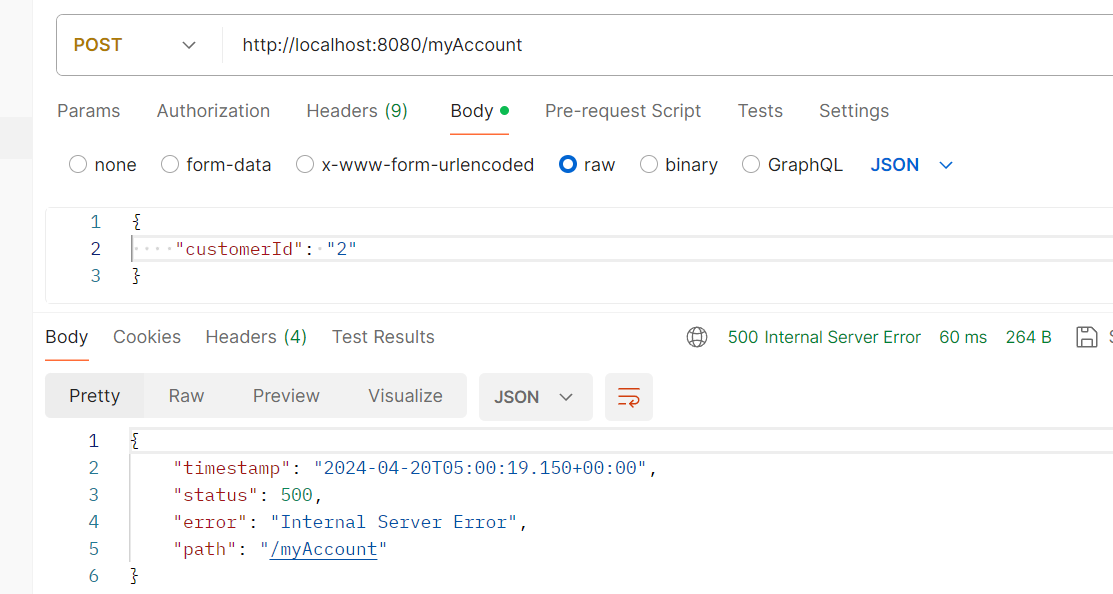
}

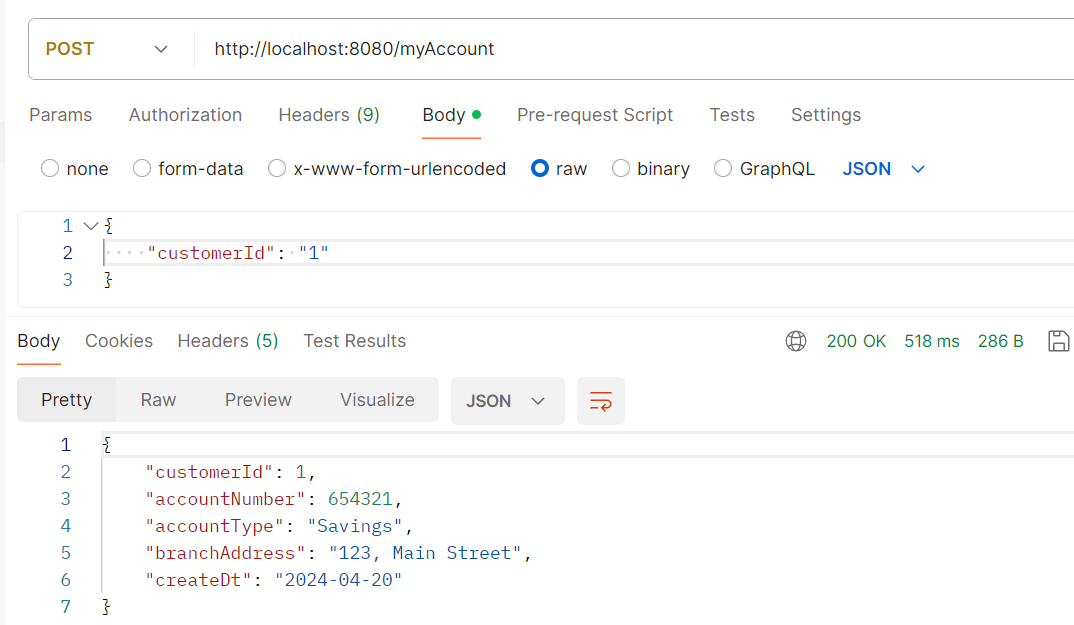
Step – 3

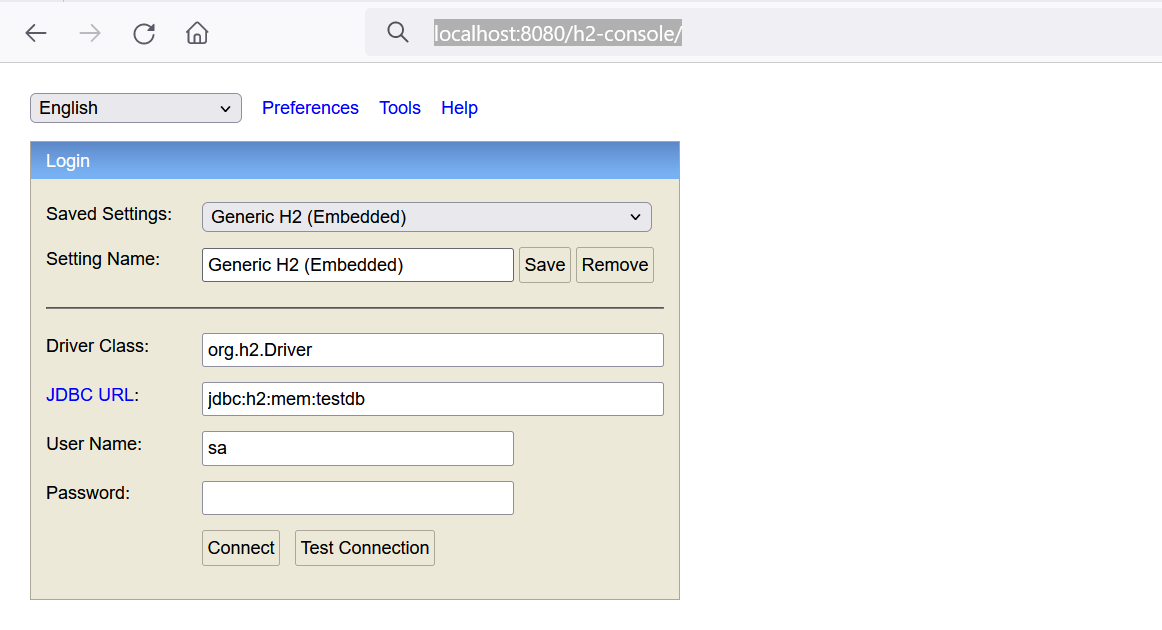
Create basic SB microservice project

<groupId>com.eazybytes</groupId>

<artifactId>accounts</artifactId>







Step – 3

Create basic SB microservice project

<groupId>com.eazybytes</groupId>

<artifactId>loans</artifactId>

