Building Microservices with Spring Boot  
**Step -**

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-validation</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

package com.eazybytes.accounts;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class AccountsApplication {

public static void main(String[] args) {

SpringApplication.run(AccountsApplication.class, args);

}

}

**Step –**

package com.eazybytes.accounts.controller;

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

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

@RestController

public class AccountsController {

@GetMapping("sayHello")

public String sayHello(){

return "Hello World!";

}

}

GET <http://localhost:8080/sayHello>  
  
**Step -**server:

port: 8080

spring:

datasource:

url: jdbc:h2:mem:testdb

driverClassName: org.h2.Driver

username: sa

password: ''

h2:

console:

enabled: true

jpa:

database-platform: org.hibernate.dialect.H2Dialect

hibernate:

ddl-auto: update

show-sql: true

CREATE TABLE IF NOT EXISTS `customer` (

`customer\_id` int AUTO\_INCREMENT PRIMARY KEY,

`name` varchar(100) NOT NULL,

`email` varchar(100) NOT NULL,

`mobile\_number` varchar(20) NOT NULL,

`created\_at` date NOT NULL,

`created\_by` varchar(20) NOT NULL,

`updated\_at` date DEFAULT NULL,

`updated\_by` varchar(20) DEFAULT NULL

);

CREATE TABLE IF NOT EXISTS `accounts` (

`customer\_id` int NOT NULL,

`account\_number` int AUTO\_INCREMENT PRIMARY KEY,

`account\_type` varchar(100) NOT NULL,

`branch\_address` varchar(200) NOT NULL,

`created\_at` date NOT NULL,

`created\_by` varchar(20) NOT NULL,

`updated\_at` date DEFAULT NULL,

`updated\_by` varchar(20) DEFAULT NULL

);

<http://localhost:8080/h2-console>  
  
Step –

package com.eazybytes.accounts.entity;

import jakarta.persistence.Column;

import jakarta.persistence.MappedSuperclass;

import lombok.Getter;

import lombok.Setter;

import lombok.ToString;

import java.time.LocalDateTime;

@MappedSuperclass

@Getter @Setter @ToString

public class BaseEntity {

@Column(updatable = false)

private LocalDateTime createdAt;

@Column(updatable = false)

private String createdBy;

@Column(insertable = false)

private LocalDateTime updatedAt;

@Column(insertable = false)

private String updatedBy;

}

package com.eazybytes.accounts.entity;

import jakarta.persistence.\*;

import lombok.\*;

import org.hibernate.annotations.GenericGenerator;

@Entity

@Getter @Setter @ToString @AllArgsConstructor @NoArgsConstructor

public class Customer extends BaseEntity {

@Id

@GeneratedValue(strategy= GenerationType.AUTO,generator="native")

@GenericGenerator(name = "native",strategy = "native")

@Column(name="customer\_id")

private Long customerId;

private String name;

private String email;

@Column(name="mobile\_number")

private String mobileNumber;

}

package com.eazybytes.accounts.entity;

import jakarta.persistence.\*;

import lombok.\*;

@Entity

@Getter @Setter @ToString @AllArgsConstructor @NoArgsConstructor

public class Accounts extends BaseEntity {

@Column(name="customer\_id")

private Long customerId;

@Column(name="account\_number")

@Id

// There will be a logic to generate 10 digit the account number, since accounts numbers cannot be 1,2,3 etc

private Long accountNumber;

@Column(name="account\_type")

private String accountType;

@Column(name="branch\_address")

private String branchAddress;

}  
  
package com.eazybytes.accounts.repository;

import com.eazybytes.accounts.entity.Accounts;

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

import org.springframework.stereotype.Repository;

@Repository

public interface AccountsRepository extends JpaRepository<Accounts, Long> {

}

package com.eazybytes.accounts.repository;

import com.eazybytes.accounts.entity.Customer;

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

import org.springframework.stereotype.Repository;

@Repository

public interface CustomerRepository extends JpaRepository<Customer, Long> {

}

Step –

package com.eazybytes.accounts.dto;

import lombok.Data;

@Data

public class AccountsDto {

private Long accountNumber;

private String accountType;

private String branchAddress;

}

package com.eazybytes.accounts.dto;

import lombok.Data;

@Data

public class CustomerDto {

private String name;

private String email;

private String mobileNumber;

private AccountsDto accountsDto;

}

package com.eazybytes.accounts.dto;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data @AllArgsConstructor

public class ResponseDto {

private String statusCode;

private String statusMsg;

}

package com.eazybytes.accounts.dto;

import lombok.AllArgsConstructor;

import lombok.Data;

import org.springframework.http.HttpStatus;

import java.time.LocalDateTime;

@Data @AllArgsConstructor

public class ErrorResponseDto {

private String apiPath;

private HttpStatus errorCode;

private String errorMessage;

private LocalDateTime errorTime;

}

Step –

package com.eazybytes.accounts.service;

import com.eazybytes.accounts.dto.CustomerDto;

public interface IAccountsService {

void createAccount(CustomerDto customerDto);

}

package com.eazybytes.accounts.service.impl;

import org.springframework.stereotype.Service;

import com.eazybytes.accounts.dto.CustomerDto;

import com.eazybytes.accounts.repository.AccountsRepository;

import com.eazybytes.accounts.repository.CustomerRepository;

import com.eazybytes.accounts.service.IAccountsService;

import lombok.AllArgsConstructor;

@Service

@AllArgsConstructor

public class AccountsServiceImpl implements IAccountsService {

//No need to @Autowired since there is only single constructor accepting parameters in the class

private AccountsRepository accountsRepository;

private CustomerRepository customerRepository;

@Override

public void createAccount(CustomerDto customerDto) {

}

}

package com.eazybytes.accounts.constants;

public class AccountsConstants {

private AccountsConstants() {

// restrict instantiation

}

public static final String SAVINGS = "Savings";

public static final String ADDRESS = "123 Main Street, New York";

public static final String STATUS\_201 = "201";

public static final String MESSAGE\_201 = "Account created successfully";

public static final String STATUS\_200 = "200";

public static final String MESSAGE\_200 = "Request processed successfully";

public static final String STATUS\_417 = "417";

public static final String MESSAGE\_417\_UPDATE= "Update operation failed. Please try again or contact Dev team";

public static final String MESSAGE\_417\_DELETE= "Delete operation failed. Please try again or contact Dev team";

// public static final String STATUS\_500 = "500";

// public static final String MESSAGE\_500 = "An error occurred. Please try again or contact Dev team"

}

package com.eazybytes.accounts.mapper;

import com.eazybytes.accounts.dto.AccountsDto;

import com.eazybytes.accounts.entity.Accounts;

public class AccountsMapper {

// java libraries modelmapper and mapStruct

public static AccountsDto mapToAccountsDto(Accounts accounts, AccountsDto accountsDto) {

accountsDto.setAccountNumber(accounts.getAccountNumber());

accountsDto.setAccountType(accounts.getAccountType());

accountsDto.setBranchAddress(accounts.getBranchAddress());

return accountsDto;

}

public static Accounts mapToAccounts(AccountsDto accountsDto, Accounts accounts) {

accounts.setAccountNumber(accountsDto.getAccountNumber());

accounts.setAccountType(accountsDto.getAccountType());

accounts.setBranchAddress(accountsDto.getBranchAddress());

return accounts;

}

}

package com.eazybytes.accounts.mapper;

import com.eazybytes.accounts.dto.CustomerDto;

import com.eazybytes.accounts.entity.Customer;

public class CustomerMapper {

// java libraries modelmapper and mapStruct

public static CustomerDto mapToCustomerDto(Customer customer, CustomerDto customerDto) {

customerDto.setName(customer.getName());

customerDto.setEmail(customer.getEmail());

customerDto.setMobileNumber(customer.getMobileNumber());

return customerDto;

}

public static Customer mapToCustomer(CustomerDto customerDto, Customer customer) {

customer.setName(customerDto.getName());

customer.setEmail(customerDto.getEmail());

customer.setMobileNumber(customerDto.getMobileNumber());

return customer;

}

}

package com.eazybytes.accounts.controller;

import org.springframework.http.HttpStatus;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

import com.eazybytes.accounts.constants.AccountsConstants;

import com.eazybytes.accounts.dto.CustomerDto;

import com.eazybytes.accounts.dto.ResponseDto;

@RestController

@RequestMapping(path = "/api", produces = MediaType.APPLICATION\_JSON\_VALUE)

public class AccountsController {

@GetMapping("sayHello")

public String sayHello(){

return "Hello World!";

}

@PostMapping("/create")

public ResponseEntity<ResponseDto> createAccount(@RequestBody CustomerDto customerDto) {

return ResponseEntity

.status(HttpStatus.CREATED)

.body(new ResponseDto(AccountsConstants.STATUS\_201, AccountsConstants.MESSAGE\_201));

}

}