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

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

server.port=8090

package com.eazybytes.loans.entity;

@MappedSuperclass

@EntityListeners(AuditingEntityListener.class)

@Getter

@Setter

@ToString

public class BaseEntity {

    @CreatedDate

    @Column(updatable = false)

    private LocalDateTime createdAt;

    @CreatedBy

    @Column(updatable = false)

    private String createdBy;

    @LastModifiedDate

    @Column(insertable = false)

    private LocalDateTime updatedAt;

    @LastModifiedBy

    @Column(insertable = false)

    private String updatedBy;

}

package com.eazybytes.loans.entity;

@Entity

@Getter @Setter @ToString @AllArgsConstructor @NoArgsConstructor

public class Loans extends BaseEntity {

    @Id

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

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

    private Long loanId;

    private String mobileNumber;

    private String loanNumber;

    private String loanType;

    private int totalLoan;

    private int amountPaid;

    private int outstandingAmount;

}

package com.eazybytes.loans.repository;

@Repository

public interface LoansRepository extends JpaRepository<Loans, Long> {

    Optional<Loans> findByMobileNumber(String mobileNumber);

    Optional<Loans> findByLoanNumber(String loanNumber);

}

loans\src\main\resources\schema.sql

CREATE TABLE IF NOT EXISTS `loans` (

  `loan\_id` int NOT NULL AUTO\_INCREMENT,

  `mobile\_number` varchar(15) NOT NULL,

  `loan\_number` varchar(100) NOT NULL,

  `loan\_type` varchar(100) NOT NULL,

  `total\_loan` int NOT NULL,

  `amount\_paid` int NOT NULL,

  `outstanding\_amount` int NOT NULL,

  `created\_at` date NOT NULL,

  `created\_by` varchar(20) NOT NULL,

  `updated\_at` date DEFAULT NULL,

  `updated\_by` varchar(20) DEFAULT NULL,

  PRIMARY KEY (`loan\_id`)

);

package com.eazybytes.loans.dto;

@Data

public class LoansDto {

    @NotEmpty(message = "Mobile Number can not be a null or empty")

    @Pattern(regexp="(^$|[0-9]{10})",message = "Mobile Number must be 10 digits")

    private String mobileNumber;

    @NotEmpty(message = "Loan Number can not be a null or empty")

    @Pattern(regexp="(^$|[0-9]{12})",message = "LoanNumber must be 12 digits")

    private String loanNumber;

    @NotEmpty(message = "LoanType can not be a null or empty")

    private String loanType;

    @Positive(message = "Total loan amount should be greater than zero")

    private int totalLoan;

    @PositiveOrZero(message = "Total loan amount paid should be equal or greater than zero")

    private int amountPaid;

    @PositiveOrZero(message = "Total outstanding amount should be equal or greater than zero")

    private int outstandingAmount;

}

package com.eazybytes.loans.dto;

@Data @AllArgsConstructor

public class ResponseDto {

    private String statusCode;

    private String statusMsg;

}

package com.eazybytes.loans.dto;

@Data @AllArgsConstructor

public class ErrorResponseDto {

    private  String apiPath;

    private HttpStatus errorCode;

    private  String errorMessage;

    private LocalDateTime errorTime;

}

package com.eazybytes.loans.constants;

public final class LoansConstants {

    private LoansConstants() {

        // restrict instantiation

    }

    public static final String  HOME\_LOAN = "Home Loan";

    public static final int  NEW\_LOAN\_LIMIT = 1\_00\_000;

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

    public static final String  MESSAGE\_201 = "Loan 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.loans.service;

public interface ILoansService {

    void createLoan(String mobileNumber);

    LoansDto fetchLoan(String mobileNumber);

    boolean updateLoan(LoansDto loansDto);

    boolean deleteLoan(String mobileNumber);

}

package com.eazybytes.loans.service.impl;

@Service

@AllArgsConstructor

public class LoansServiceImpl implements ILoansService {

    private LoansRepository loansRepository;

    @Override

    public void createLoan(String mobileNumber) {

        Optional<Loans> optionalLoans= loansRepository.findByMobileNumber(mobileNumber);

        if(optionalLoans.isPresent()){

            throw new LoanAlreadyExistsException("Loan already registered with given mobileNumber "+mobileNumber);

        }

        loansRepository.save(createNewLoan(mobileNumber));

    }

    private Loans createNewLoan(String mobileNumber) {

        Loans newLoan = new Loans();

        long randomLoanNumber = 100000000000L + new Random().nextInt(900000000);

        newLoan.setLoanNumber(Long.toString(randomLoanNumber));

        newLoan.setMobileNumber(mobileNumber);

        newLoan.setLoanType(LoansConstants.HOME\_LOAN);

        newLoan.setTotalLoan(LoansConstants.NEW\_LOAN\_LIMIT);

        newLoan.setAmountPaid(0);

        newLoan.setOutstandingAmount(LoansConstants.NEW\_LOAN\_LIMIT);

        return newLoan;

    }

    @Override

    public LoansDto fetchLoan(String mobileNumber) {

        Loans loans = loansRepository.findByMobileNumber(mobileNumber).orElseThrow(

                () -> new ResourceNotFoundException("Loan", "mobileNumber", mobileNumber)

        );

        return LoansMapper.mapToLoansDto(loans, new LoansDto());

    }

    @Override

    public boolean updateLoan(LoansDto loansDto) {

        Loans loans = loansRepository.findByLoanNumber(loansDto.getLoanNumber()).orElseThrow(

                () -> new ResourceNotFoundException("Loan", "LoanNumber", loansDto.getLoanNumber()));

        LoansMapper.mapToLoans(loansDto, loans);

        loansRepository.save(loans);

        return  true;

    }

    @Override

    public boolean deleteLoan(String mobileNumber) {

        Loans loans = loansRepository.findByMobileNumber(mobileNumber).orElseThrow(

                () -> new ResourceNotFoundException("Loan", "mobileNumber", mobileNumber)

        );

        loansRepository.deleteById(loans.getLoanId());

        return true;

    }

}

package com.eazybytes.loans.mapper;

public class LoansMapper {

    public static LoansDto mapToLoansDto(Loans loans, LoansDto loansDto) {

        loansDto.setLoanNumber(loans.getLoanNumber());

        loansDto.setLoanType(loans.getLoanType());

        loansDto.setMobileNumber(loans.getMobileNumber());

        loansDto.setTotalLoan(loans.getTotalLoan());

        loansDto.setAmountPaid(loans.getAmountPaid());

        loansDto.setOutstandingAmount(loans.getOutstandingAmount());

        return loansDto;

    }

    public static Loans mapToLoans(LoansDto loansDto, Loans loans) {

        loans.setLoanNumber(loansDto.getLoanNumber());

        loans.setLoanType(loansDto.getLoanType());

        loans.setMobileNumber(loansDto.getMobileNumber());

        loans.setTotalLoan(loansDto.getTotalLoan());

        loans.setAmountPaid(loansDto.getAmountPaid());

        loans.setOutstandingAmount(loansDto.getOutstandingAmount());

        return loans;

    }

}

package com.eazybytes.loans.exception;

@ResponseStatus(value = HttpStatus.NOT\_FOUND)

public class ResourceNotFoundException extends RuntimeException {

    public ResourceNotFoundException(String resourceName, String fieldName, String fieldValue){

        super(String.format("%s not found with the given input data %s : '%s'", resourceName, fieldName, fieldValue));

    }

}

package com.eazybytes.loans.exception;

@ResponseStatus(value = HttpStatus.BAD\_REQUEST)

public class LoanAlreadyExistsException extends RuntimeException {

    public LoanAlreadyExistsException(String message){

        super(message);

    }

}

package com.eazybytes.loans.exception;

@ControllerAdvice

public class GlobalExceptionHandler extends ResponseEntityExceptionHandler {

    @Override

    protected ResponseEntity<Object> handleMethodArgumentNotValid(

            MethodArgumentNotValidException ex, HttpHeaders headers, HttpStatusCode status, WebRequest request) {

        Map<String, String> validationErrors = new HashMap<>();

        List<ObjectError> validationErrorList = ex.getBindingResult().getAllErrors();

        validationErrorList.forEach((error) -> {

            String fieldName = ((FieldError) error).getField();

            String validationMsg = error.getDefaultMessage();

            validationErrors.put(fieldName, validationMsg);

        });

        return new ResponseEntity<>(validationErrors, HttpStatus.BAD\_REQUEST);

    }

    @ExceptionHandler(Exception.class)

    public ResponseEntity<ErrorResponseDto> handleGlobalException(Exception exception,

                                                                  WebRequest webRequest) {

        ErrorResponseDto errorResponseDTO = new ErrorResponseDto(

                webRequest.getDescription(false),

                HttpStatus.INTERNAL\_SERVER\_ERROR,

                exception.getMessage(),

                LocalDateTime.now()

        );

        return new ResponseEntity<>(errorResponseDTO, HttpStatus.INTERNAL\_SERVER\_ERROR);

    }

    @ExceptionHandler(ResourceNotFoundException.class)

    public ResponseEntity<ErrorResponseDto> handleResourceNotFoundException(ResourceNotFoundException exception,

                                                                            WebRequest webRequest) {

        ErrorResponseDto errorResponseDTO = new ErrorResponseDto(

                webRequest.getDescription(false),

                HttpStatus.NOT\_FOUND,

                exception.getMessage(),

                LocalDateTime.now()

        );

        return new ResponseEntity<>(errorResponseDTO, HttpStatus.NOT\_FOUND);

    }

    @ExceptionHandler(LoanAlreadyExistsException.class)

    public ResponseEntity<ErrorResponseDto> handleLoanAlreadyExistsException(LoanAlreadyExistsException exception,

                                                                             WebRequest webRequest){

        ErrorResponseDto errorResponseDTO = new ErrorResponseDto(

                webRequest.getDescription(false),

                HttpStatus.BAD\_REQUEST,

                exception.getMessage(),

                LocalDateTime.now()

        );

        return new ResponseEntity<>(errorResponseDTO, HttpStatus.BAD\_REQUEST);

    }

}

package com.eazybytes.loans.controller;

@RestController

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

@AllArgsConstructor

@Validated

public class LoansController {

    private ILoansService iLoansService;

    @PostMapping("/create")

    public ResponseEntity<ResponseDto> createLoan(@RequestParam

                                                      @Pattern(regexp="(^$|[0-9]{10})",message = "Mobile number must be 10 digits")

                                                      String mobileNumber) {

        iLoansService.createLoan(mobileNumber);

        return ResponseEntity

                .status(HttpStatus.CREATED)

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

    }

    @GetMapping("/fetch")

    public ResponseEntity<LoansDto> fetchLoanDetails(@RequestParam

                                                               @Pattern(regexp="(^$|[0-9]{10})",message = "Mobile number must be 10 digits")

                                                               String mobileNumber) {

        LoansDto loansDto = iLoansService.fetchLoan(mobileNumber);

        return ResponseEntity.status(HttpStatus.OK).body(loansDto);

    }

    @PutMapping("/update")

    public ResponseEntity<ResponseDto> updateLoanDetails(@Valid @RequestBody LoansDto loansDto) {

        boolean isUpdated = iLoansService.updateLoan(loansDto);

        if(isUpdated) {

            return ResponseEntity

                    .status(HttpStatus.OK)

                    .body(new ResponseDto(LoansConstants.STATUS\_200, LoansConstants.MESSAGE\_200));

        }else{

            return ResponseEntity

                    .status(HttpStatus.EXPECTATION\_FAILED)

                    .body(new ResponseDto(LoansConstants.STATUS\_417, LoansConstants.MESSAGE\_417\_UPDATE));

        }

    }

    @DeleteMapping("/delete")

    public ResponseEntity<ResponseDto> deleteLoanDetails(@RequestParam

                                                                @Pattern(regexp="(^$|[0-9]{10})",message = "Mobile number must be 10 digits")

                                                                String mobileNumber) {

        boolean isDeleted = iLoansService.deleteLoan(mobileNumber);

        if(isDeleted) {

            return ResponseEntity

                    .status(HttpStatus.OK)

                    .body(new ResponseDto(LoansConstants.STATUS\_200, LoansConstants.MESSAGE\_200));

        }else{

            return ResponseEntity

                    .status(HttpStatus.EXPECTATION\_FAILED)

                    .body(new ResponseDto(LoansConstants.STATUS\_417, LoansConstants.MESSAGE\_417\_DELETE));

        }

    }

}

package com.eazybytes.loans.audit;

@Component("auditAwareImpl")

public class AuditAwareImpl implements AuditorAware<String> {

    @Override

    public Optional<String> getCurrentAuditor() {

        return Optional.of("LOANS\_MS");

    }

}

package com.eazybytes.loans;

@SpringBootApplication

@EnableJpaAuditing(auditorAwareRef = "auditAwareImpl")

public class LoansApplication {

    public static void main(String[] args) {

        SpringApplication.run(LoansApplication.class, args);

    }

}

















