<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=9000

package com.eazybytes.cards;

@SpringBootApplication

@EnableJpaAuditing(auditorAwareRef = "auditAwareImpl")

public class CardsApplication {

    public static void main(String[] args) {

        SpringApplication.run(CardsApplication.class, args);

    }

}

package com.eazybytes.cards.audit;

@Component("auditAwareImpl")

public class AuditAwareImpl implements AuditorAware<String> {

    @Override

    public Optional<String> getCurrentAuditor() {

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

    }

}

package com.eazybytes.cards.constants;

public final class CardsConstants {

    private CardsConstants() {

        // restrict instantiation

    }

    public static final String  CREDIT\_CARD = "Credit Card";

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

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

    public static final String  MESSAGE\_201 = "Card 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.cards.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.cards.entity;

@Entity

@Getter @Setter @ToString @AllArgsConstructor @NoArgsConstructor

public class Cards extends BaseEntity {

    @Id

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

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

    private Long cardId;

    private String mobileNumber;

    private String cardNumber;

    private String cardType;

    private int totalLimit;

    private int amountUsed;

    private int availableAmount;

}

package com.eazybytes.cards.dto;

@Data

public class CardsDto {

    @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 = "Card Number can not be a null or empty")

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

    private String cardNumber;

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

    private String cardType;

    @Positive(message = "Total card limit should be greater than zero")

    private int totalLimit;

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

    private int amountUsed;

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

    private int availableAmount;

}

package com.eazybytes.cards.dto;

@Data @AllArgsConstructor

public class ResponseDto {

    private String statusCode;

    private String statusMsg;

}

package com.eazybytes.cards.dto;

@Data @AllArgsConstructor

public class ErrorResponseDto {

    private  String apiPath;

    private HttpStatus errorCode;

    private  String errorMessage;

    private LocalDateTime errorTime;

}

package com.eazybytes.cards.mapper;

public class CardsMapper {

    public static CardsDto mapToCardsDto(Cards cards, CardsDto cardsDto) {

        cardsDto.setCardNumber(cards.getCardNumber());

        cardsDto.setCardType(cards.getCardType());

        cardsDto.setMobileNumber(cards.getMobileNumber());

        cardsDto.setTotalLimit(cards.getTotalLimit());

        cardsDto.setAvailableAmount(cards.getAvailableAmount());

        cardsDto.setAmountUsed(cards.getAmountUsed());

        return cardsDto;

    }

    public static Cards mapToCards(CardsDto cardsDto, Cards cards) {

        cards.setCardNumber(cardsDto.getCardNumber());

        cards.setCardType(cardsDto.getCardType());

        cards.setMobileNumber(cardsDto.getMobileNumber());

        cards.setTotalLimit(cardsDto.getTotalLimit());

        cards.setAvailableAmount(cardsDto.getAvailableAmount());

        cards.setAmountUsed(cardsDto.getAmountUsed());

        return cards;

    }

}

package com.eazybytes.cards.repository;

@Repository

public interface CardsRepository extends JpaRepository<Cards, Long> {

    Optional<Cards> findByMobileNumber(String mobileNumber);

    Optional<Cards> findByCardNumber(String cardNumber);

}

package com.eazybytes.cards.service;

public interface ICardsService {

    void createCard(String mobileNumber);

    CardsDto fetchCard(String mobileNumber);

    boolean updateCard(CardsDto cardsDto);

    boolean deleteCard(String mobileNumber);

}

package com.eazybytes.cards.service.impl;

@Service

@AllArgsConstructor

public class CardsServiceImpl implements ICardsService {

    private CardsRepository cardsRepository;

    @Override

    public void createCard(String mobileNumber) {

        Optional<Cards> optionalCards= cardsRepository.findByMobileNumber(mobileNumber);

        if(optionalCards.isPresent()){

            throw new CardAlreadyExistsException("Card already registered with given mobileNumber "+mobileNumber);

        }

        cardsRepository.save(createNewCard(mobileNumber));

    }

    private Cards createNewCard(String mobileNumber) {

        Cards newCard = new Cards();

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

        newCard.setCardNumber(Long.toString(randomCardNumber));

        newCard.setMobileNumber(mobileNumber);

        newCard.setCardType(CardsConstants.CREDIT\_CARD);

        newCard.setTotalLimit(CardsConstants.NEW\_CARD\_LIMIT);

        newCard.setAmountUsed(0);

        newCard.setAvailableAmount(CardsConstants.NEW\_CARD\_LIMIT);

        return newCard;

    }

    @Override

    public CardsDto fetchCard(String mobileNumber) {

        Cards cards = cardsRepository.findByMobileNumber(mobileNumber).orElseThrow(

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

        );

        return CardsMapper.mapToCardsDto(cards, new CardsDto());

    }

    @Override

    public boolean updateCard(CardsDto cardsDto) {

        Cards cards = cardsRepository.findByCardNumber(cardsDto.getCardNumber()).orElseThrow(

                () -> new ResourceNotFoundException("Card", "CardNumber", cardsDto.getCardNumber()));

        CardsMapper.mapToCards(cardsDto, cards);

        cardsRepository.save(cards);

        return  true;

    }

    @Override

    public boolean deleteCard(String mobileNumber) {

        Cards cards = cardsRepository.findByMobileNumber(mobileNumber).orElseThrow(

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

        );

        cardsRepository.deleteById(cards.getCardId());

        return true;

    }

}

package com.eazybytes.cards.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.cards.exception;

@ResponseStatus(value = HttpStatus.BAD\_REQUEST)

public class CardAlreadyExistsException extends RuntimeException {

    public CardAlreadyExistsException(String message){

        super(message);

    }

}

package com.eazybytes.cards.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 ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR)

                .body(errorResponseDTO);

    }

    @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(CardAlreadyExistsException.class)

    public ResponseEntity<ErrorResponseDto> handleCardAlreadyExistsException(CardAlreadyExistsException 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.cards.controller;

@RestController

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

@AllArgsConstructor

@Validated

public class CardsController {

    private ICardsService iCardsService;

    @PostMapping("/create")

    public ResponseEntity<ResponseDto> createCard(@Valid @RequestParam

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

                                                      String mobileNumber) {

        iCardsService.createCard(mobileNumber);

            return ResponseEntity

                    .status(HttpStatus.CREATED)

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

    }

    @GetMapping("/fetch")

    public ResponseEntity<CardsDto> fetchCardDetails(@RequestParam

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

                                                               String mobileNumber) {

        CardsDto cardsDto = iCardsService.fetchCard(mobileNumber);

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

    }

    @PutMapping("/update")

    public ResponseEntity<ResponseDto> updateCardDetails(@Valid @RequestBody CardsDto cardsDto) {

        boolean isUpdated = iCardsService.updateCard(cardsDto);

        if(isUpdated) {

            return ResponseEntity

                    .status(HttpStatus.OK)

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

        }else{

            return ResponseEntity

                    .status(HttpStatus.EXPECTATION\_FAILED)

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

        }

    }

    @DeleteMapping("/delete")

    public ResponseEntity<ResponseDto> deleteCardDetails(@RequestParam

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

                                                                String mobileNumber) {

        boolean isDeleted = iCardsService.deleteCard(mobileNumber);

        if(isDeleted) {

            return ResponseEntity

                    .status(HttpStatus.OK)

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

        }else{

            return ResponseEntity

                    .status(HttpStatus.EXPECTATION\_FAILED)

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

        }

    }

}















