

Feilhåndtering

Repository

Logger `logger` = `LoggerFactory.getLogger(KundeRepository.class)`;

Metoder som ikke returnerer noe (void):

```
public boolean kjøpBillett(Billett billett) {
    String sql = "INSERT INTO Billett (film, antall, fornavn, etternavn, telefonnr, epost) VALUES (?, ?, ?, ?, ?, ?)";
    try {
        db.update(sql, billett.getFilm(), billett.getAntall(), billett.getFornavn(),
            billett.getEtternavn(), billett.getTelefonnr(), billett.getEpost());
        return true;
    } catch (Exception e) {
        logger.error("Feil i kjøpBillett" + e);
        return false;
    }
}
```

Metoders som returnerer noe:

```
public List<Billett> hentRegisteret() {
    String sql = "SELECT * from Billett";
    try {
        List<Billett> alleBilletter = db.query(sql, new BeanPropertyRowMapper(Billett.class));
        return alleBilletter;
    } catch (Exception e) {
        logger.error("Feil i kjøpBillett" + e);
        return null;
    }
}
```

Feilhåndtering

Controller

I metoder der det ikke returnerer noe:

```
@PostMapping("/kjop")
public void kjøp(Billett billett, HttpServletResponse response) throws IOException {
    if(!rep.kjøpBillett(billett)){
        response.sendError(HttpStatus.INTERNAL_SERVER_ERROR.value(), s: "Feil i DB, prøv igjen senere");
    }
}
```

Alternativt:

```
public void lagreKunde(Kunde kunde, HttpServletResponse response) {
    if(!rep.lagreKunde(kunde)) {
        try {
            response.sendError(HttpStatus.INTERNAL_SERVER_ERROR.value(),
                               "Feil i DB – prøv igjen senere");
        } catch (Exception e) {
        }
    }
}
```

I metoder der det skal returneres noe:

```
@PostMapping("/hentRegister")
public List<Billett> hentRegister(HttpServletResponse response) throws IOException {
    List<Billett> register = rep.hentRegisteret();
    if (register == null){
        response.sendError(HttpStatus.INTERNAL_SERVER_ERROR.value(), s: "Feil i DB, prøv igjen senere");
    }
    return rep.hentRegisteret();
}
```

Metoden tar to parametere: en "**Billett**"-parameter som representerer dataene som klienten sender i POST-forespørselen, og en "**HttpServletResponse**"-parameter som representerer responsen som webapplikasjonen (repositoriet) sender tilbake til klienten.

!rep.kjøpBillett(billett) betyr "ikke kjøp billett", altså repositoriet returnerer en false verdi og det er ikke mulig å kjøpe billett. Da vil altså uttrykket "**!rep.kjøpBillett(billett)**" bli sant eller true.

Hvis det ikke er mulig å kjøpe billetten, sender metoden en HTTP-feilrespons tilbake til klienten ved å kalle metoden "**sendError**" på "**HttpServletResponse**"-objektet ("**response**").

Feilresponsen inneholder en feilmelding som sier "Feil i DB, prøv igjen senere" og en HTTP-statuskode som indikerer at det har oppstått en intern serverfeil (**HttpStatus.INTERNAL_SERVER_ERROR**). Denne statuskoden tilsvarer feilkoden 500.

Kort sagt, koden sjekker om det er mulig å kjøpe en billett og sender tilbake en feilrespons hvis det ikke er mulig.

Feilhåndtering

JavaScript

Feilhåndtering på klient: sender feilmeldingen til brukeren dersom noe med AJAX feiler (generell feilhåndterings-metode som kan stå alene i JS-filen)

```
function lagreKunde() {  
  const kunde = {  
    navn : $("#navn").val(),  
    adresse : $("#adresse").val(),  
  }  
  $.post( "/lagreKunde", kunde, function() {  
    window.location.href = '/';  
  })  
  .fail(function(jqXHR) {  
    const json = $.parseJSON(jqXHR.responseText);  
    $("#feil").html(json.message);  
  });  
}
```

Vanlig å gi navnet jqXHR (jQuery-Xml-Http-Request), men selvsagt ikke nødvendig

Hent ut "message"

JSON-streng i jqXHR.responseText

OSLO METROPOLITAN UNIVERSITY
STORBYUNIVERSITETET

Name	Headers	Preview	Response	Cookies	Timing
localhost					
bootstrap.min.css					
jquery.min.js					
index.js					
hentKunder					
favicon.ico					

▼ (timestamp: "2019-07-14T12:44:16.813+0000", status: 500, error: "Internal error: "Internal Server Error"
message: "Feil i DB - prøv igjen senere"
path: "/hentKunder"
status: 500
timestamp: "2019-07-14T12:44:16.813+0000"

- Legges til bak alle post-kall
- Obs! Husk å ha en id for #feil i html der hvor innholdet skal hentes ut (typ i en oversikt)

Transaksjoner

Eneste en trenger å gjøre for å sikre transaksjoner mot databasen er å anvende dekoratøren

@Transaction

På toppen like under @PostMapping eller @GetMapping (på metodene som gjør kall til databasen)

Inputvalidering

RegEx = Regular Expressions

- Et eksempel: `[a-zæøåA-ZÆØÅ]{3,20}`
^ beskriver alle store og små bokstaver (inkludert norske) på mellom 3 og 20 tegn

- `[]` – spesifiserer enkelttegn som skal godtas
 - `[a-zæøåA-ZÆØÅ. \-]`
 - Alle norske tegn, punktum, mellomrom og minustegn (må escapes med `\`)
- `{ }` – spesifiserer antall ganger enkelttegnene skal gjentas
 - `[a-zæøåA-ZÆØÅ. \-]{2,20}` – mellom 2 og 20 tegn
 - `[a-zæøåA-ZÆØÅ. \-]{6,}` – mer enn 6 tegn
 - `[0-9]{4}` – akkurat 4 siffer
 - `\d{4}` – akkurat 4 siffer (som over)

^for at bindestrek skal være lov må du ta `\` før bindestreken (i java må du ha to)

Vi skal bruke RegEx både i Java og JS (både på server og klient siden)

- **Merk: i JS må vi ha med ^ og \$ etter eller før /. Det trenger ikke Java. I tillegg rammes RegEx-koden inn ved bruk av / isteden for "" som vanlig.**

- Eksempel med en e-postadresse:
 - `[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}`
- Kredittkort:
 - `(?:4[0-9]{12}(?:[0-9]{3})? # Visa`
| `(?:5[1-5][0-9]{2} # MasterCard`
| `222[1-9]|22[3-9][0-9]|2[3-6][0-9]{2}|27[01][0-9]|2720)[0-9]{12}`
| `3[47][0-9]{13} # American Express`
| `3(?:0[0-5]|68[0-9])[0-9]{11} # Diners Club`
)

Inputvalidering

Lag en egen fil for inputvalidering i JS. Den kan se sånn ut:

```
function validerOgLagreKunde() {  
  const navnOK = validerNavn($("#navn").val());  
  const adresseOK = validerAdresse($("#adresse").val());  
  
  if(navnOK && adresseOK) {  
    lagreKunde();  
  }  
}
```

```
function validerNavn(navn) {  
  const regexp = /^[a-zæøåA-ZÆØÅ. \-]{2,30}$/;  
  const ok = regexp.test(navn);  
  
  if(!ok) {  
    $("#feilNavn").html("Navn må bestå av 2 til 30 bokstaver.");  
    return false;  
  } else {  
    $("#feilNavn").html("");  
    return true;  
  }  
}  
  
function validerAdresse(adresse) {  
  const regexp = /^[0-9a-zæøåA-ZÆØÅ. \-]{2,50}$/;
```

Svar i HTML-filen

```


<label for="navn">Navn</label>  
  <input type="text" id="navn" onchange="validerNavn(this.value)" />  
  <span id="feilNavn" style="color: red" />  
</div>


```

Inputvalidering

I controlleren:

```
private boolean validerKunde(Kunde kunde) {  
    String regexpNavn = "[a-zæøåA-ZÆØÅ. \\-]{2,30}";  
    String regexpAdresse = "[0-9a-zæøåA-ZÆØÅ. \\-]{2,50}";  
  
    boolean navnOK = kunde.getNavn().matches(regexpNavn);  
    boolean adresseOK = kunde.getAdresse().matches(regexpAdresse);  
  
    if(navnOK && adresseOK) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

```
@PostMapping("/lagreKunde")  
public void lagreKunde(Kunde kunde, HttpServletResponse response) throws IOException {  
    if(!validerKunde(kunde)) {  
        response.sendError(HttpStatus.NOT_ACCEPTABLE.value());  
    } else {  
        if (!rep.lagreKunde(kunde)) {  
            response.sendError(HttpStatus.INTERNAL_SERVER_ERROR.value());  
        }  
    }  
}
```

Sessions

Fremgangsmåte for session
(her: en session for innlogging)

I controlleren :

```
@Autowired
private HttpSession session;

@GetMapping("/login")
public boolean login(Kunde kunde) {
    if(rep.sjekkNavnOgPassord(kunde)) {
        session.setAttribute(s: "innlogget", kunde);
        return true;
    } else {
        return false;
    }
}

@GetMapping("/logout")
public void logout() {
    session.removeAttribute(s: "innlogget");
}
```

Dersom du ønsker at de innloggede brukerne skal kunne se alle kundene på index-siden - gå inn i den allerede opprettede funksjonen her og legg til dette:

```
@GetMapping("/hentKunder")
public List<Kunde> hentAlle (HttpServletResponse response) throws IOException {
    if(session.getAttribute(s: "innlogget") != null) {
        List<Kunde> alleKunder = rep.hentAlleKunder();
        if (alleKunder == null) {
            response.sendError(HttpStatus.INTERNAL_SERVER_ERROR.value(),
            );
        }
        return alleKunder;
    } else {
        response.sendError(HttpStatus.NOT_FOUND.value());
    }
}
```

Usikker på når denne trengs

Usikker på når denne trengs

Sessions

I JavaScript-filen (gjærne lag en egen fil for innlogging)

```
function login(){
    const kunde = {
        navn : $("#navn").val(),
        passord : $("#passord").val()
    }
    $.get("/login", kunde, function(innlogget) {
        if(innlogget){
            window.location.href = "/";
        }
        else{
            $("#feil").html("Feil brukernavn eller passord");
        }
    })
    .fail(function() {
        $("#feil").html("Serverfeil- prøv igjen senere");
    })
    .done(function() {
    });
}
```


Det er også en knapp for å logge av. Koden for denne på klienten er rett frem:

```
function logout(){
    const url = "/logout";
    $.get(url, function() {
        window.location.href = 'login.html';
    })
}
```

 Husk knapp i HTML !!

Dersom du ønsker at de innloggede brukerne skal kunne se alle kundene på indexsiden - gå inn i den allerede opprettede funksjonen her og legg til dette:

```
function hentAlleKunder() {
    $.get( "/hentKunder", function( kunder ) {
        formaterKunder(kunder);
    })
    .fail(function(status) {
        if(status.status=="404")
            $("#feil").html("Må logge inn for å vise kundene!");
    });
};
```

 Usikker på når denne trengs

I repository

```
public boolean sjekkBrukernavnOgPassord(Bruker bruker) {
    Object[] param = new Object[]{bruker.getBrukernavn(),bruker.getPassord()};
    String sql = "Select count(*) from Bruker where brukernavn = ? and passord = ?";
    try {
        int antall = db.queryForObject(sql,param,Integer.class);
        if(antall > 0){
            return true;
        }
        return false;
    } catch(Exception e){
        logger.error("Feil i sjekkBrukernavnOgPassord:"+e);
        return false;
    }
}
```


Sessions

Repository og controller samlet:

(merk: inparameteren er feil. Må stå Bruker bruker i think)

```
public boolean sjekkBrukeren(bruker){
    Object[] param = new Object[]{bruker.getBrukernavn(), bruker.getPassord()};
    String sql = "Select count (*) from Bruker where brukernavn = ? and passord = ?";

    try {
        int antall = db.queryForObject(sql, param, Integer.class);
        if (antall > 0){
            session.setAttribute("Innlogget", bruker);
            return true;
        }
        return false;
    } catch (Exception e){
        logger.error("Feil i sjekkBrukeren");
        return false;
    }
}
```

Auto_generated

Hva gjør man når man skal sette inn et objekt inn i databasen, som skal ha en autogenerated id? Man kan jo ikke sette inn en id man ikke vet hva er...

```
@PostMapping("/lagre")
@Transactional
public void lagreMelding(Ordre o) {
    String sql1 = "INSERT INTO Kunde (fornavn,etternavn,adresse) VALUES(?,?,?)";
    String sql2 = "INSERT INTO Ordre (KId,varenavn,pris) VALUES(?,?,?)";
    KeyHolder id = new GeneratedKeyHolder();
    try {
        db.update(con -> {
            PreparedStatement par = con.prepareStatement(sql1, new String[]{"KId"});
            par.setString(1, o.getFornavn());
            par.setString(2, o.getEtternavn());
            par.setString(3, o.getAdresse());
            return par;
        }, id);
        int kid = id.getKey().intValue();
        db.update(sql2, kid, o.getVarenavn(), o.getPris());
    }
    catch (Exception e) {
        logger.error("Feil i lagre ordre! " + e);
    }
}
```

Merk:

- O kommer fra inparameteren !! Husk derfor at de skal samsvare