DAT108 Oblig3 h22 - Web #1

Sist oppdatert av Lars-Petter Helland 10.10.2022

Øvingen skal gjennomføres i grupper på **2-4** studenter. (Studenter som har fått tillatelse til å levere alene registrerer seg også som en gruppe!)

Dere danner selv grupper i Canvas ved å registrere dere i en av de 80 forhåndsdefinerte gruppene «**DAT108 Oblig3 gruppe x**».

NB! Alle gruppemedlemmer må være registrert før innlevering.

NB! Det er nytt gruppesett for hver oblig. Dvs. at dere må registrere gruppen på nytt i gruppesettet for ny oblig selv om dere er samme gruppe!

Øvingen er lagt ut 11. oktober.

Innleveringsfrist er søndag 23. oktober.

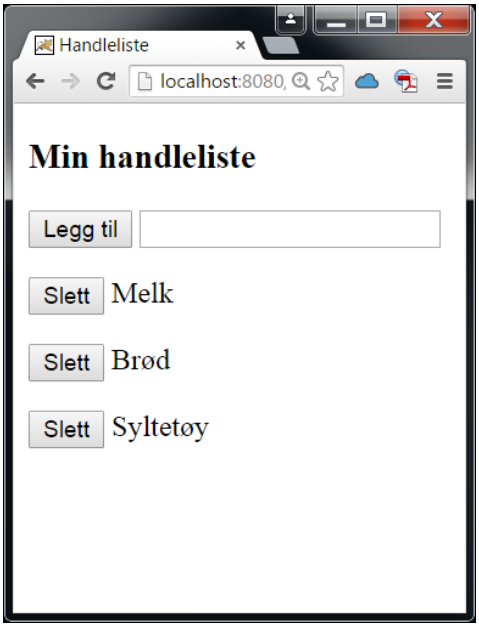
Innleveringen er en zip i Canvas. NB! Zip-en skal hete **Oblig3_gr17.zip** for gruppe 17 osv (dere forstår) ... Denne skal inneholde:

1. Et pdf-dokument som inneholder:
    a) En liste av hvem som er med i gruppen (for å unngå gratispassasjerer som melder seg inn i gruppe uten at det er avtalt).
    b) Skjermutskrift fra kjøringer av programmene, slik at vi kan se at de virker.
2. Et Eclipse-prosjekt for Oppgave 1
3. Et Eclipse-prosjekt for Oppgave 2

Oppgave1 - Tjenerside-programmering - En enkel handleliste

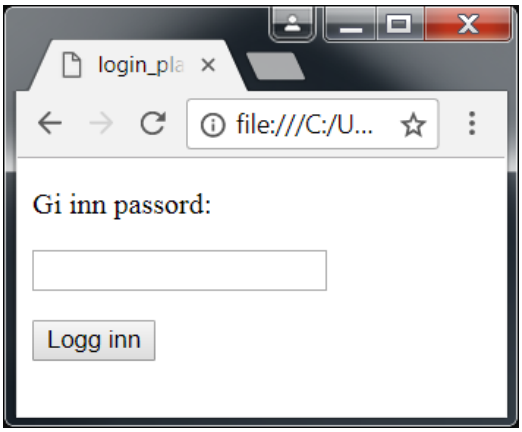
Det skal lages en web-applikasjon med en innloggingsside og en hovedside som inneholder følgende tre brukstilfeller:

1. Bruker kan se elementene i handlelisten.
2. Bruker kan legge til et nytt element i handlelisten.
3. Bruker kan slette et element fra handlelisten.



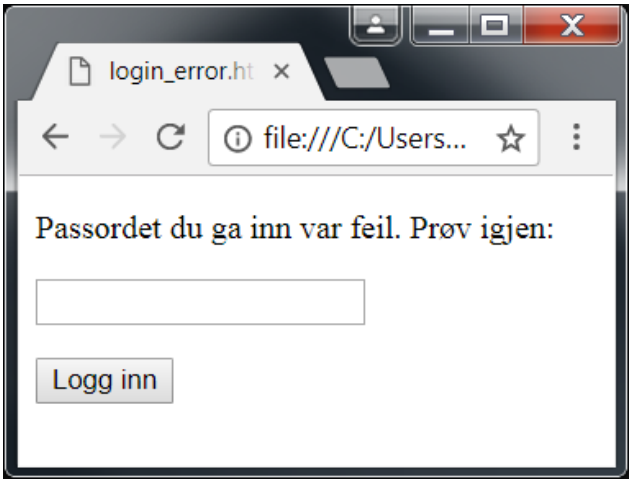
Alle brukstilfeller krever at man er "innlogget". Innlogging gjøres ved å oppgi riktig passord. **Dette er altså en applikasjon med én felles handleliste for alle brukere. Jeg vet det er en teit/urealistisk overforenkling, men jeg ville ikke gjøre det mer komplisert i denne omgang.**

Innloggingssiden kan se slik ut:

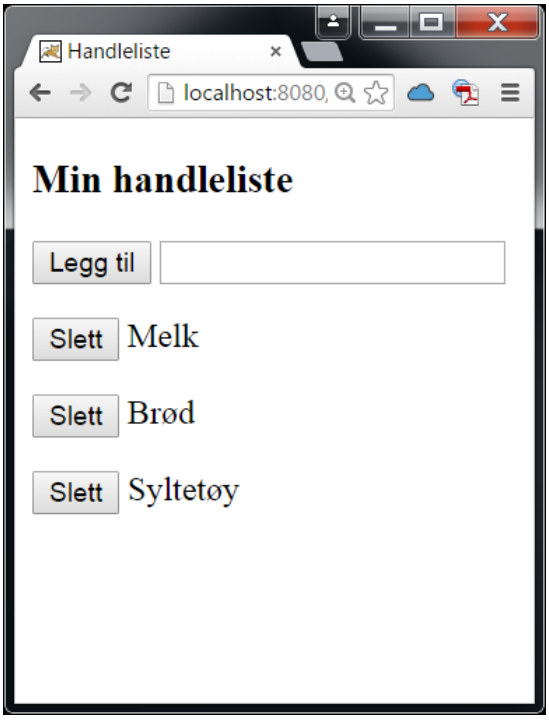


Korrekt passord skal være «hardkodet» i klartekst i **application.properties**.

Hvis du oppgir feil passord får du opp innloggingssiden på nytt med en feilmelding slik:



Hvis du oppgir rett passord går du videre til handlelistesiden der du kan se på, legge til og slette elementer i handlelisten. …



Det er ikke noe knapp for utlogging. Utlogging skjer automatisk hvis handlelisten ikke er brukt på et antall sekunder/minutter som oppgis i application.properties. (Under testing kan dere sette dette til kort tid for å sjekke at det virker)

Hvis man prøver å utføre en av forespørslene til handlelisten (se på / legge til / slette) når man er utlogget blir man omdirigert til login-siden.

Lagring av meldingene

Vi gjør en forenkling og "lagrer" handlelisten i en egnet datastruktur i minnet => Hver gang vi restarter tjeneren er vi altså tilbake til start.

Å lagre handlelisten i en database (med JPA) kan være en frivillig ekstraoppgave. Vi kommer mer inn på JPA på neste øving. :)

POST, GET og PRG

Spesielt i denne applikasjonen er at vi har to ulike typer "postinger" på hovedsiden, nemlig

- Bruker kan legge til et nytt element i handlelisten.
- Bruker kan slette et element fra handlelisten.

Det er opp til dere hvordan dere organiserer controller(ne) og hvordan dere identifiserer hva som ønskes utført.

Det er et krav at dere bruker Post-Redirect-Get (PRG).

Robust mot manglende brukerinput

Vær nøye på robust håndtering av manglende brukerinput (tom streng) når man ønsker å legge til et element i handlelisten. Ved tom streng skal man rett og slett ikke lagre noe. Det skal ikke vises noen feilmelding. (Dere kan godt sette enda strengere krav, f.eks. minst to bokstaver ...)

Ufarliggjøring av brukerinput

Siden brukerinputen skal brukes i resultatsiden må vi kode/escape alle HTML-spesialtegn før resultatsiden skrives.

Flere brukere, trådsikkerhet og håndtering av ugyldig input

Brukerne jobber i prinsippet med den samme handlelisten. Tenk gjennom, og håndter på en god måte problemstillinger knyttet til dette. F.eks. kan flere brukere prøve å slette en vare de begge "ser" i listen. Hva skjer når andremann prøver å slette? ...

Programstruktur

Bruk hjelpeklasser og hjelpemetoder for å få en god ansvarsfordeling mellom klasser og "high cohesion".

Lykke til!

Oppgave2 – JavaScript – Applikasjon for å trille terning

I denne oppgaven skal du lage en web-applikasjon med JavaScript for å trille en terning. Funksjonaliteten til en terning skal implementeres som en JavaScript klasse *Dice*. Terningen må knyttes til HTML elementer for å trille terningen og vise resultatet. Dette gjøres av en JavaScript klasse *DiceController*.

Du svarer på denne oppgaven ved å levere et Eclipse prosjekt med løsningen. Bruk et «Dynamic Web Project» med en TomEE webtjener.

I forelesningene har vi ikke ennå kommet til egendefinert klasser i JavaScript. Koden under viser hvordan du kan lage en klasse **Person**. Klassen sin konstruktør lar oss sette navn og fødselsdato til en person. Klassen inkluderer også en metode *getAge* som returnerer alderen til personen.

```
/**
 * Klasse Person. Alle forekomster av Person må ha et navn og en
 * fødselsdato.
 *
 * @author Bjarte Kileng
 */
class Person {

    // Tegnet '#' forteller at felt eller metode er Private
    #name;
    #borndate;

    /**
     * @param {String} name   - Navnet til personen
     * @param {Date} borndate - Fødselsdato til personen
     */
    constructor(name = null, borndate = null) {
        this.#name = name;
        this.#borndate = borndate;
    }

    /**
     * Get age of person
     * @public
     */
    getAge() {
        const now = new Date ();
        return now.getFullYear() - this.#borndate.getFullYear();
    }
}
```

Eksempelet dokumenterer koden ved å bruke JSDoc. Da kan Eclipse gi feilmelding hvis. f.eks. en metode brukes feil, selv om dette er lov i JavaScript.

Websiden for å trille terning kan inneholde følgende HTML:

```
<div id="root">
    <p><button type="button" data-dicebutton>
        Trill terning
    </button></p>
    <p>Resultat: <span data-diceoutput></span></p>
</div>
```

Klassen DiceController

Klassen *DiceController* knytter terningen til HTML koden over, og kan kalles på følgende måte:

```
const rootElement = new DiceController("root");
new DiceController(rootElement);
```

I koden over er *root* verdien til ID-attributtet til HTML **DIV**-elementet for terningen.

Forekomst av *DiceController* må trille terning etter klikk på knappen som har attributt *data-dicebutton*, og svaret må legges inn i elementet med attributt *data-diceoutput*.

For at løsningen skal være portabel må *DiceController* referer HTML elementer relativt til HTML **DIV** elementet *rootElement* . I koden under benyttes CSS-selektor for å lage en referanse til elementet med attributt *data-dicebutton*:

```
const dicebutton = rootElement.querySelector("*[data-dicebutton]");
```

For å trille terningen må vi kytte en metode for å trille terningen mot klikk på knappen på websiden. Det kan vi gjøre f.eks. med følgende kode i *DiceController*:

```
dicebutton.addEventListener("click",this.#rollDice.bind(this));
```

Koden over forutsetter en metode *#rollDice* i *DiceController*. Betydningen av *bind* vil vi komme tilbake til senere i kurset.

Etter at terningen er trillet skal svaret vises på websiden. Det kan vi få til f.eks. med følgende kode:

```
diceoutputElement.innerText = dice.getValue();
```

I koden over forutsettes det at terningen sin verdi er tilgjengelig ved å kalle metoden *dice.getValue, og diceoutputElement* må inneholde en referanse til HTML elementet som har attributt *data-diceoutput*.

Klassen Dice

Klassen *Dice* må ha en metode *roll* for å trille terning, og en metode *getValue* som returnerer resultatet etter å ha trillet terning. Metoden *roll* må produsere et vilkårlig heltall mellom 1 og 6.

Observer at klassen *Dice* skal ikke vite noe om websiden, og at klassen *DiceController* kun skal aksessere web-elementer relativt til det omsluttende div-elementet, som i koden over har ID verdi *root*.

Tips: Objektet **Math** har en metode for å finne et tilfeldig tall.