

DAT108 h20 - Frivillige oppgaver før Oblig3

Sist oppdatert av Lars-Petter Helland 01.10.2022

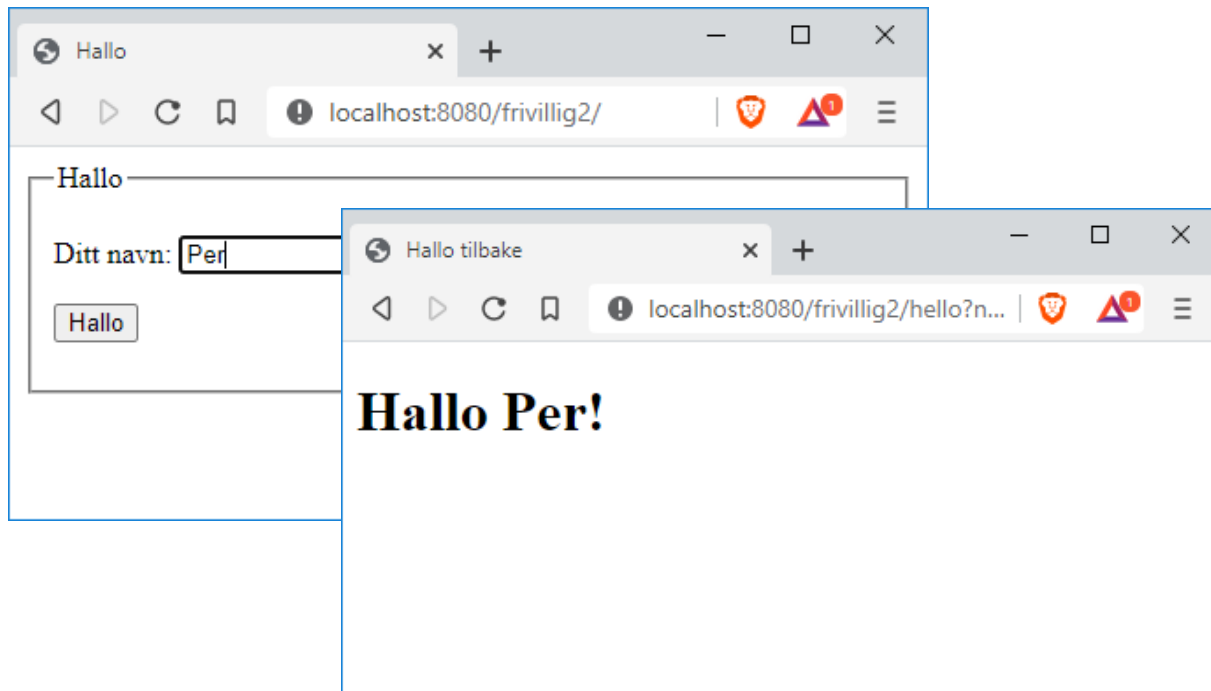
Arbeidsmengden på denne øvingen er beregnet til ca. **5-15 timer**.

Jobb med oppgavene minst én time per dag slik at du kommer gjennom alt! :)
Nå går toget!!!

NB! Forsøk å løse så mye du klarer før du kommer på labben. Arbeidsmengden er **mye større** enn tiden du har på labben. Bruk lab-tiden til å få hjelp til det du ikke har fått til.

Oppgave 1 - Request-headere

Lag en helt enkel "hallo-applikasjon" som lar bruker taste inn et navn, og som svarer med en side med teksten «Hallo navn!» slik:



En av tingene en nettleser sender med er hvilke(t) språk som foretrekkes. Det angis f.eks. som: "no,en;q=0.5". Dette eksemplet skal tolkes som at norsk (no) foretrekkes (implisitt vekt 1.0), og at engelsk (en) er andrevalget med vekt 0.5.

Dere skal fikse applikasjonen slik at den støtter språkene norsk, engelsk og tysk. Hvis norsk er foretrukket svares "God dag <navn>!". Engelsk "Hello <navn>!" Tysk "Guten tag <navn>". Default språk skal være engelsk.

Tips! Request-headeren Accept-Language er den som inneholder denne informasjonen. Enten kan dere bruke den direkte, eller sjekke om Spring MVC har enda en enklere måte (Locale).

Oppgave 2 - Brukerinput via <form>, data binding, validering og håndtering av ugyldig input

Denne oppgaven handler om

- Brukerinput via <form>
- Data binding mot et form-backing objekt
- Spring sitt <form:> tag-bibliotek
- Validering med Bean Validation (JSR 380)
- Håndtering av ugyldig brukerinput
- Post-Redirect-Get

Du skal lage en liten applikasjon som har to dynamiske sider: Et skjema for å registrere opplysninger, og en side som er en kvittering for at alle opplysninger er gyldige og "sendt inn". Ved ugyldig input skal skjemaet vises igjen med meldinger om hva som er feil/mangler.

Kvitteringen skal vise alle dataene som er mottatt, og skal kunne refreshes uten at skjemadata sendes på nytt. Skjemaet kan f.eks. se slik ut (uten og med feilmeldinger):

Dine data

Fornavn

Etternavn

Fødselsår

Postnr

Mobil

Epost

Registrer

Dine data

Fornavn
 Må være mellom 3 og 20 tegn

Etternavn
 Må være mellom 3 og 20 tegn

Fødselsår

Postnr
 Må være 4 siffer

Mobil
 Må være 8 siffer

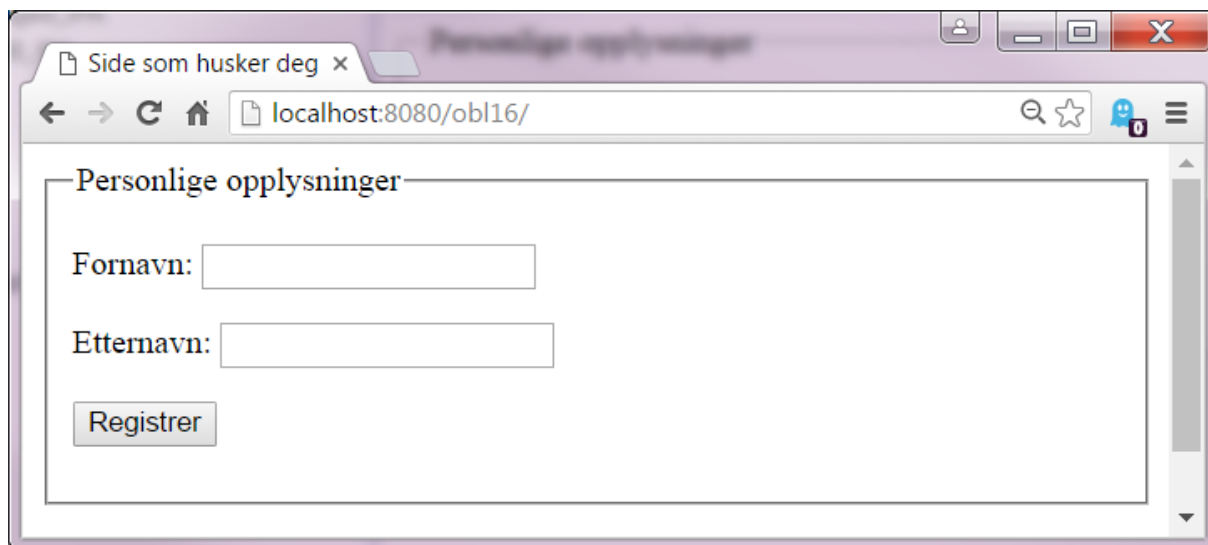
Epost
 Må være en gyldig epost-adresse

Registrer

Bruk GET/POST slik det er naturlig. All input er tekst, unntatt fødselsår, som er et tall.

Oppgave 3 - Cookies

Dere begynner med å lage en dynamisk side som ser slik ut (dvs. en controller + et view):



The screenshot shows a web browser window with a single tab titled 'Side som husker deg x'. The address bar displays 'localhost:8080/obl16/'. The page content is a form titled 'Personlige opplysninger' (Personal information). The form contains two input fields: 'Fornavn:' (First name) and 'Etternavn:' (Last name). Below these fields is a button labeled 'Registrer' (Register). The browser's status bar at the bottom is empty.

Når man trykker på Registrer-knappen skal det POST-es til samme URL som deretter redirecter tilbake til skjemaet (PRG).

Det vi ønsker å få til er at skjemaet er forhåndsutfylt med det (fornavn og etternavn) som evt. ble registrert forrige gang (gjerne for et år siden!!). Til dette bruker vi cookies.

Litt hint om hvordan det må løses:

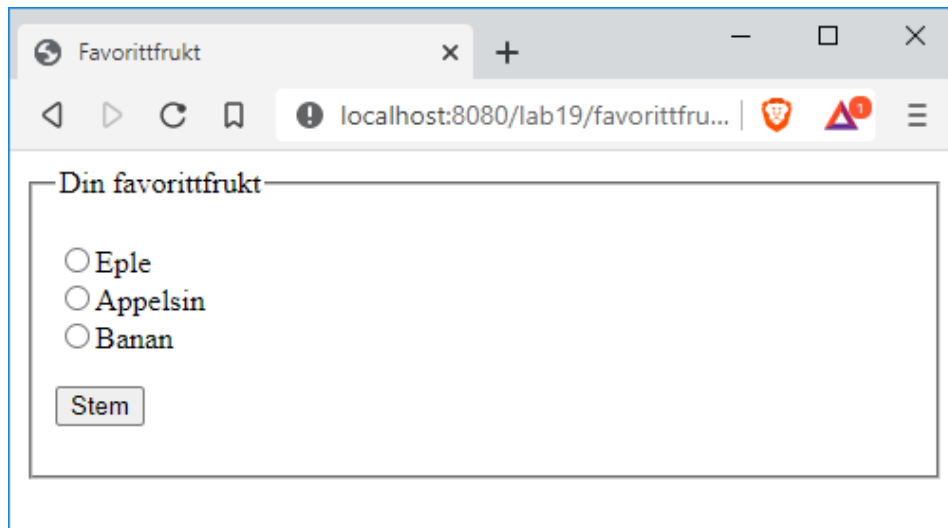
- I controller+view som genererer skjemaet må man hente inn evt. cookies fra requesten og legge verdiene inn i input-boksene.
- I controller som mottar input må man sende med cookies i redirect-responsen, dvs. de må legges til i responsen før vi gjør redirect.

Husk:

- Cookie-navn og -verdier kan ikke inneholde diverse spesialtegn, inkl. æøå. Cookie-verdier må derfor alltid URL-kodes og -dekodes for å sikre gyldige verdier. Tips: Se siden «Cookie-encoding» på forelesningen om hvordan dette gjøres.
- Test gjerne dette med JUnit.

Oppgave 4 - Favorittfrukt

Du skal lage en side der folk kan "stemme på" sin favorittfrukt. Den kan se ut ca. slik (ikke bry deg om URL-en på bildet), f.eks. kalt **stemmeskjema.html**:



Din favorittfrukt

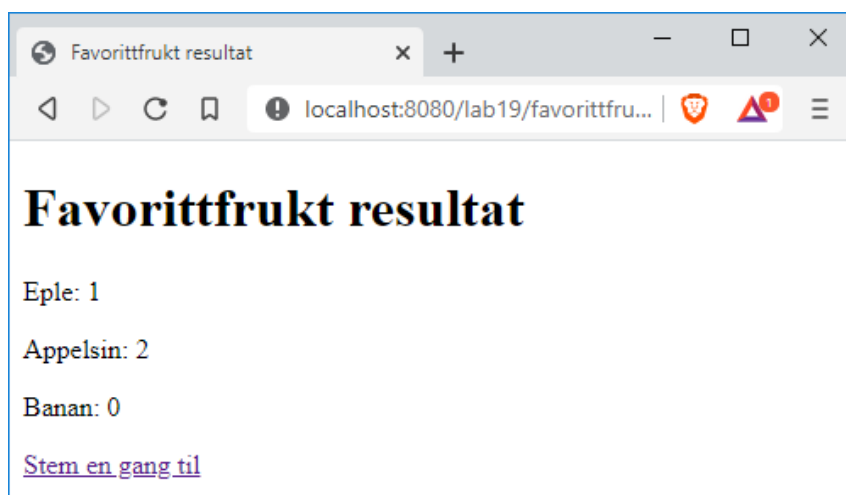
☐ Eple

☐ Appelsin

☐ Banan

Web-applikasjonen skal holde orden på stemmeresultatet. For enkelhets skyld kan vi kun ha dem lagret i minnet (ved omstart av tjeneren vil jo disse nullstilles, men ...).

Resultat-viewet skal generere en HTML-side som ser ca. slik ut:



Favorittfrukt resultat

Eple: 1

Appelsin: 2

Banan: 0

[Stem en gang til](#)

Post-Redirect-Get: Det å stemme på favorittfrukt bør kanskje være en POST-request. Hvis responsen fra denne requesten er å vise resultatet direkte vil refreshing av resultatet føre til en dobbeltposting. Implementér en PRG-løsning på dette problemet.

Prøv gjerne å stemme fra flere weblesere (Safari, Chrome, Firefox, Edge, Brave, Opera) på samme maskin, og refresh resultatside for å se at "andre" har stemt.

Dynamisk stemmeskjema: Vi skal nå gjøre listen av stemmealternativer "dynamisk" i stedet for at den er hardkodet i en html-fil (stemmeskjema.html).

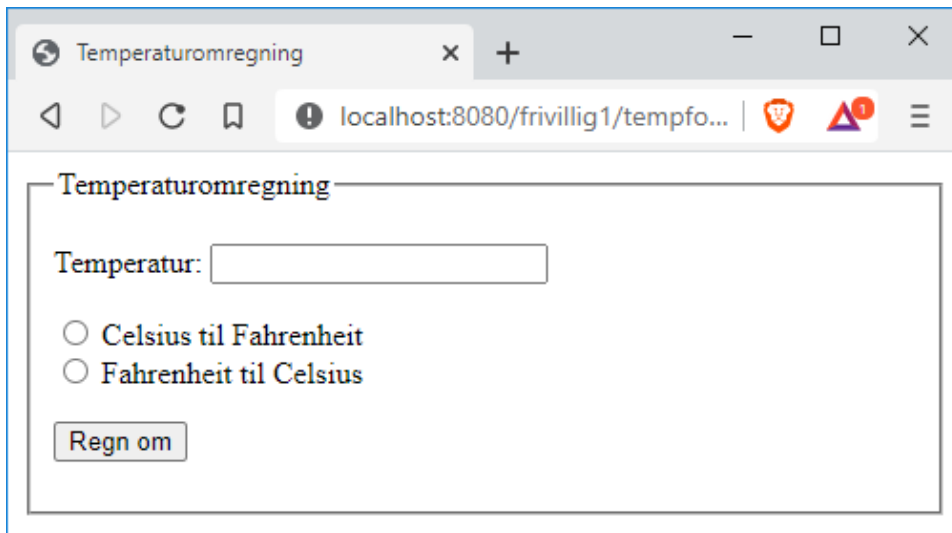
Det finnes flere muligheter på hvor stemmealternativene skal hentes fra. I denne oppgaven gjør vi det enkelt og lagrer disse alternativene i **application.properties** som en liste av verdier.

Lag en controller og et view som genererer stemmeskjemaet dynamisk basert på innholdet i **application.properties**, og bruk dette i stedet for den statiske siden **stemmeskjema.html** fra i sted. URL-mapping for det nye dynamiske stemmeskjemaet kan f.eks. være **/stemmeskjema**.

Oppgave 5 - (Manuell) validering, robust håndtering av input og formatering av output.

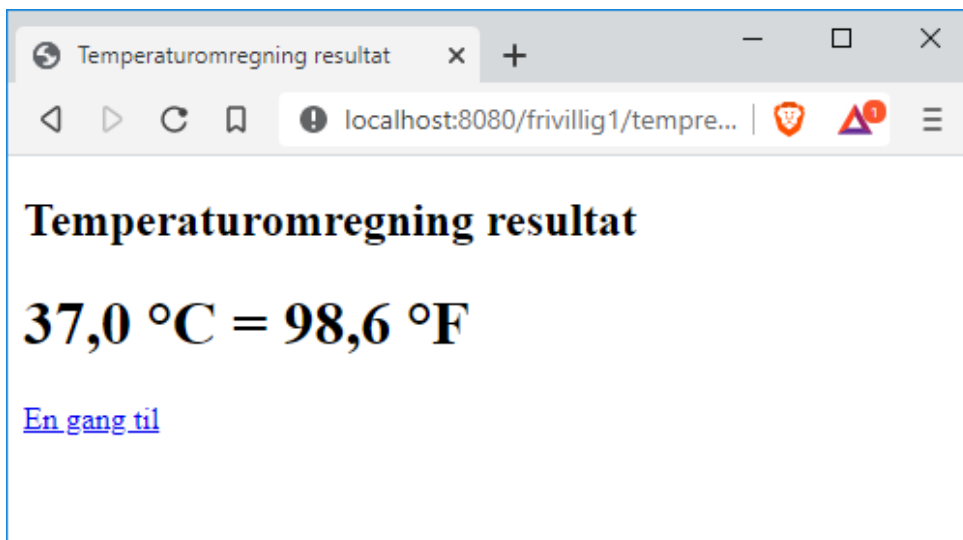
I denne oppgaven skal dere øve på robust håndtering av brukerinput. Dere skal håndtere manglende og feilformatert input, og gi respons som gir beskjed om at noe var feil.

Dere skal lage en liten applikasjon som regner om temperaturer fra Celsius til Fahrenheit eller omvendt. Inputskjemaet kan se ut omtrent slik:



The screenshot shows a web browser window titled "Temperaturomregning". The address bar shows "localhost:8080/frivillig1/tempfo...". The form has a title "Temperaturomregning" and a label "Temperatur:" followed by a text input field. Below the input field are two radio buttons: "Celsius til Fahrenheit" (selected) and "Fahrenheit til Celsius". At the bottom of the form is a button labeled "Regn om".

Ved normal bruk (input: 37 | Celsius til Fahrenheit) skal responsen se omtrent slik ut:



The screenshot shows a web browser window titled "Temperaturomregning resultat". The address bar shows "localhost:8080/frivillig1/tempre...". The page displays the result of the conversion: "37,0 °C = 98,6 °F". Below the result is a blue link labeled "En gang til".

Legg merke til følgende:

- Temperaturene i resultatet er formatert med én desimal
- Det er brukt et symbol for grader. Tips: Bruk HTML-navn eller Unicode-nummer for å få dette symbolet.

Dere skal bygge inn robust håndtering av brukerinput, dvs. at applikasjonen skal oppføre seg "skikkelig" selv om det er feil i brukerinput. Typer feil dere skal ta høyde for:

- Manglende input (null) for en eller begge input-parameterne
- Tom streng eller streng med kun blanke for temperaturen
- Ugyldig tallinput. Temperaturen skal være et desimaltall. Tallet må være ≥ -273.15 (Celsius) og ≥ -459.67 (Fahrenheit).

Så tilbake til web-applikasjonen.

Hvis det er feil i brukerinput skal en feilmelding presenteres omtrent slik:

