

Oblig4 - Webapplikasjoner

Sist oppdatert av Lars-Petter Helland 07.11.2022

Lagt til en oppgave om enhetstesting



Øvingen skal gjennomføres i grupper på **2-4** studenter.

Dere danner selv grupper i Canvas ved å registrere dere i en av de **80** forhåndsdefinerte gruppene «**DAT108 Oblig4** gruppe x». Legg helst til alle studentene i gruppen samtidig. (Gå inn på «Personer» | «DAT108 Oblig4 gruppe», og legg til dere selv i en tom gruppe)



Øvingen har veiledning på labbene E443/E403 og Discord i lab-tiden. **Innleveringsfrist er om kvelden søndag 13. november.** Vi har som mål å rette innleveringene og godkjenne innen 1+ uke. **Merk! For å kunne gå opp til eksamen må alle obligs være rettet og godkjent innen 30. november.**



Innlevering i Canvas. Denne skal inneholde:

Et dokument (**pdf**) med

- opplisting av hvem som er med i gruppen
- en fungerende URL til applikasjonens "hjemmeside"
- eksempelskjermbilder fra kjøring

Et Eclipse-prosjekt (**zip**) med kildekode til en kjørbare applikasjon, inkl. databaseskript for oppretting av databasen.

Ressurser til øvingen

Siden DAT108 ikke primært er et HTML-kurs, har vi samlet HTML-koden fra skjermbildene i oppgaveteksten pluss et lite stilsett i **Obl4-html-maler.zip**. Dere slipper derfor å bruke tid på å gjette dere frem til hvilken HTML-kode som ligger bak web-sidene i oppgaveteksten.

Dere gjerne bruke hjelpeklasser etc. som er lagt ut tidligere i kurset. NB! Husk likevel at innholdet i disse hjelpeklassene er pensum til eksamen.

Oppgaven: Web-applikasjon Påmelding til fest

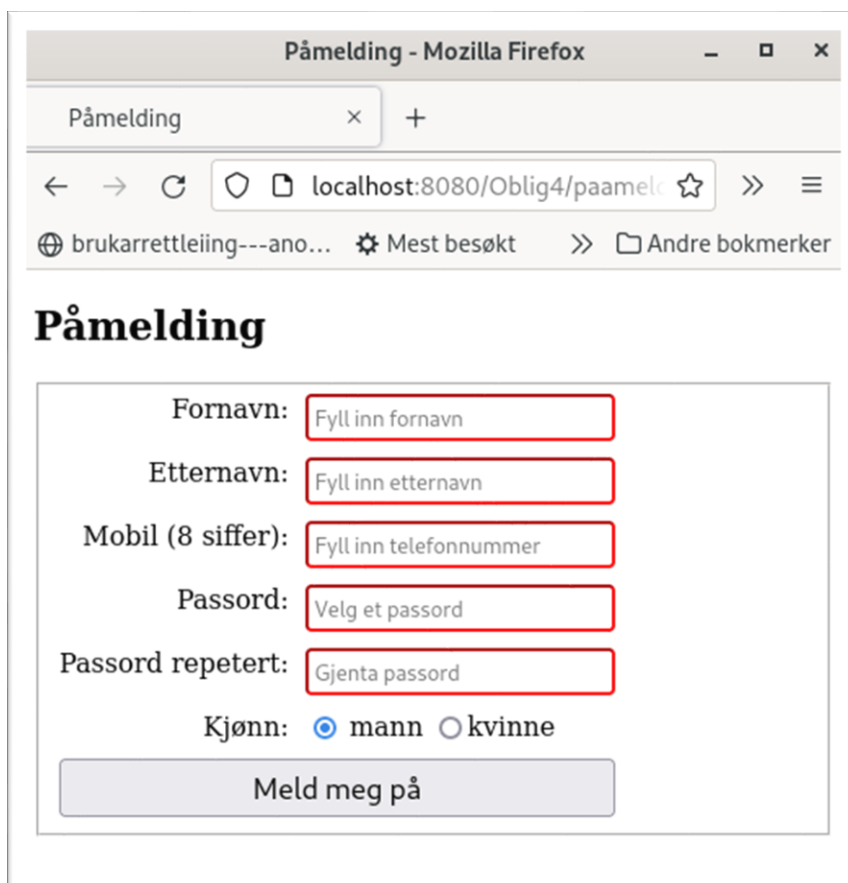
Det skal lages en enkel web-applikasjon som kan brukes til påmelding til fest.

Her er brukstilfellene i grove trekk:

1. En bruker skal kunne melde seg på festen.
2. En påmeldt deltager skal kunne se listen over påmeldte.

[Påmelding](#) / [Se deltagerlisten](#) / [Logge ut](#)

Skjema for påmelding skal se ca. slik ut:



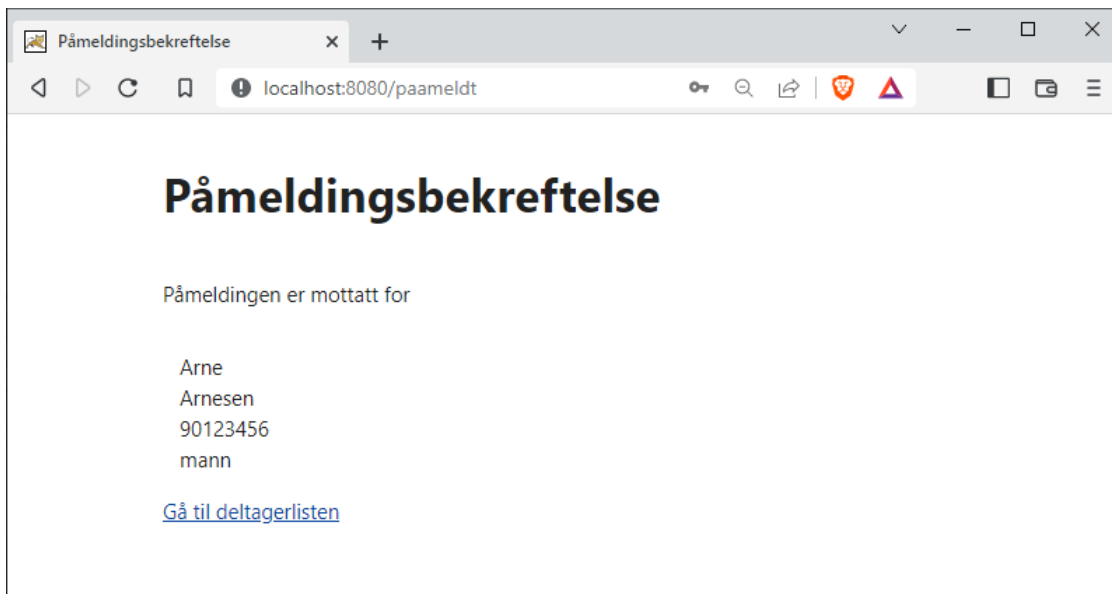
The screenshot shows a web browser window titled "Påmelding - Mozilla Firefox". The address bar shows "localhost:8080/Oblig4/paamelc". The page title is "Påmelding". The form contains the following fields and controls:

- Fornavn:
- Etternavn:
- Mobil (8 siffer):
- Passord:
- Passord repetert:
- Kjønn: ☒ mann ☐ kvinne
-

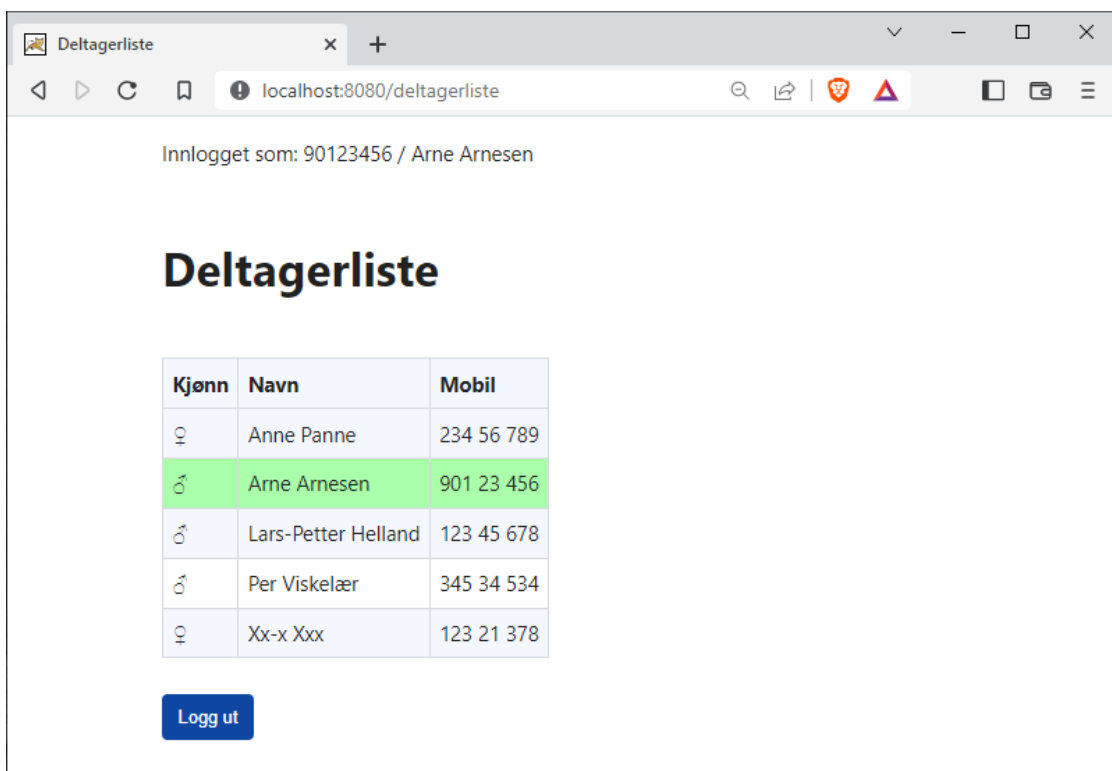
Ting å merke seg:

- Alle felter er obligatoriske og skal valideres ved påmelding, se detaljer senere.
- Mobilnummeret blir "brukernavn" i applikasjonen, se senere brukstilfelle.
- Ved påmelding blir man automatisk "logget inn" med mobilnummer som brukernavn.

Hvis alt er OK får man en bekreftelse på at påmeldingen er registrert, slik:



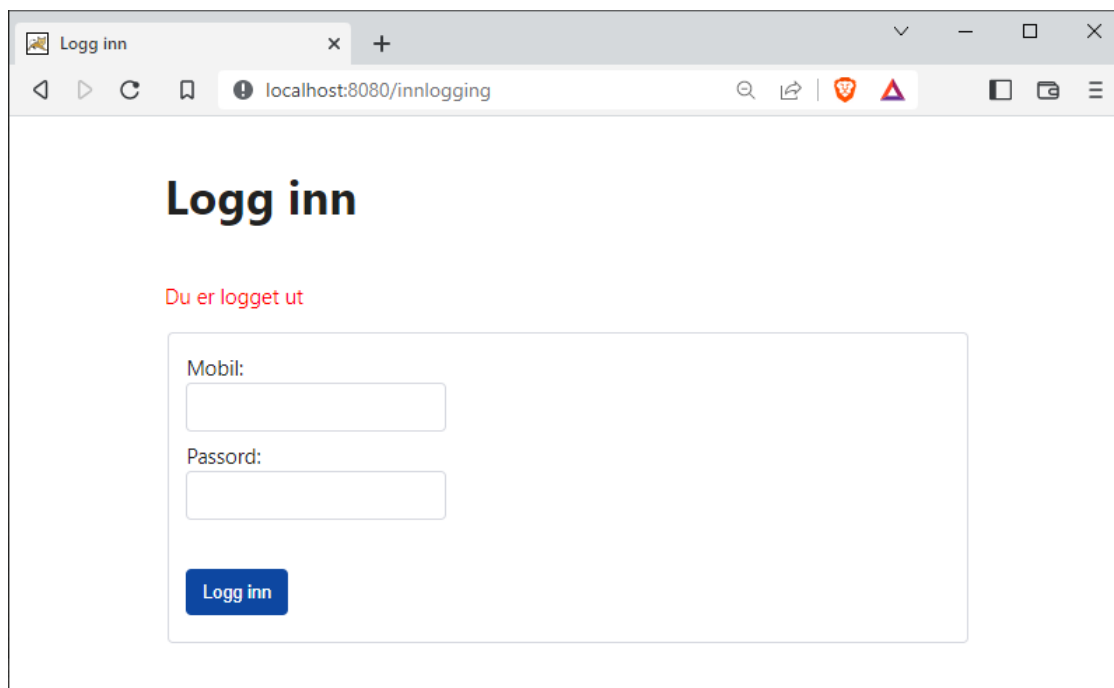
Når man trykker på lenken kommer man til deltagerlisten:



Ting å merke seg:

- Å kunne se deltagerlisten krever at man er innlogget som påmeldt deltager. Sikre mot uautorisert tilgang! Forsøk på uautorisert tilgang gir innloggingssiden, se nedenfor.
- Deltagerlisten er sortert stigende på fornavn, deretter etternavn.
- Innlogget deltager er uthøvet med grønn bakgrunn.

Når man trykker på lenken "Logg ut" logges man ut og går til innlogging:



Logg inn

Du er logget ut

Mobil:

Passord:

Logg inn

Hvis du er påmeldt kan du logge inn for å se deltagerlisten.

Etter vellykket innlogging kommer man til deltagerlisten (vist tidligere).

Ved ulike feilsituasjoner som f.eks.:

- Forsøk på requester som krever at du er innlogget (når du ikke er det)
- Ugyldig bruker og/eller passord ved innlogging
- ...

skal man sendes til innloggingssiden med feilmelding (i rødt).

Validering av input ved påmelding av ny deltager

Validering av input ved påmelding av ny deltager skal gjøres **BÅDE** i nettleser/klient **OG** på tjener.

Validering i nettleser (HTML/CSS/JavaScript) vil kunne gi bruker umiddelbar tilbakemelding mens det tastes, f.eks. om styrken på passordet. Validering på tjeneren (Java/Spring) er for å sikre at dataene som mottas og lagres i database er gyldige, f.eks. om en deltager med gitt mobil allerede er påmeldt.

For å teste validering på tjeneren må nok HTML strippes og JavaScriptet midlertidig "kommenteres ut" slik at man får sendt potensielt ugyldige data med requesten!

Valideringsregler

Regler for gyldig brukerinput ved validering på tjenersiden:

- **Fornavn** skal være 2-20 tegn og kan inneholde bokstaver (inkl. æøåÆØÅ), bindestrek og mellomrom. Første tegn skal være en stor bokstav.
- **Etternavn** skal være 2-20 tegn og kan inneholde bokstaver (inkl. æøåÆØÅ) og bindestrek (IKKE mellomrom). Første tegn skal være en stor bokstav.
- **Mobil** skal være eksakt 8 siffer, ingenting annet. Et tilleggskrav ved påmelding er at mobilnummeret IKKE må tilhøre en allerede påmeldt deltager. Alle mobilnumre i deltagerlisten skal være unike! Det siste skal kun sjekkes på serveren!
- **Passord** bør ha en viss minimumslengde. Krav utover dette bestemmer du selv. Styrken på valgt passord skal angis med farge mens man taster. Det siste skal kun sjekkes i nettleseren!
- **Repetert passord** må være likt passordet.
- **Kjønn** må være "mann" eller "kvinne".

Validering i nettleser med HTML/CSS/JavaScript

Før data sendes til tjener skal de valideres på klienten. Kun hvis alle data er gyldige skal de bli sendt til tjener.

Validering på klient gir bruker tilbakemeldinger med en gang. Bruker kan da kan rette på ting og da blir det mindre feil i dataene som går til tjeneren.

Rammefarger på input-elementene skal fortløpende vise tilstanden til data:

- Rød rammefarge skal angi at data er ugyldige.
- Grønn farge skal angi at data er gyldige.
- For «Passord» brukes også gul rammefarge som betyr at passordet er gyldig, men svakt.
- Grønn rammefarge på «Passord» betyr at passordet er gyldig og sterkt.

Noe av valideringen som beskrives under må gjøres med JavaScript, men mye kan gjøres med kun HTML5 og CSS.

Form kontroll-elementer, f.eks. input elementer har en tilstand som enten er *valid* eller *invalid*. Tilstanden er *valid* hvis utfylte data er gyldig, ellers *invalid*. Tilstanden kan styres med JavaScript, men normalt brukes HTML-attributter.

CSS har pseudoklasser *valid* og *invalid* som kan brukes for å markere tilstanden til form-data. Normalt trenger vi kun å bruke pseudoklassen *invalid*, og lar normalt tilstanden være *valid*.

CSS-koden nedenfor vil markere med rød rammefarge alle input-elementer av type *text* og *password* som har tilstanden *invalid*, og ellers er rammen grønn:

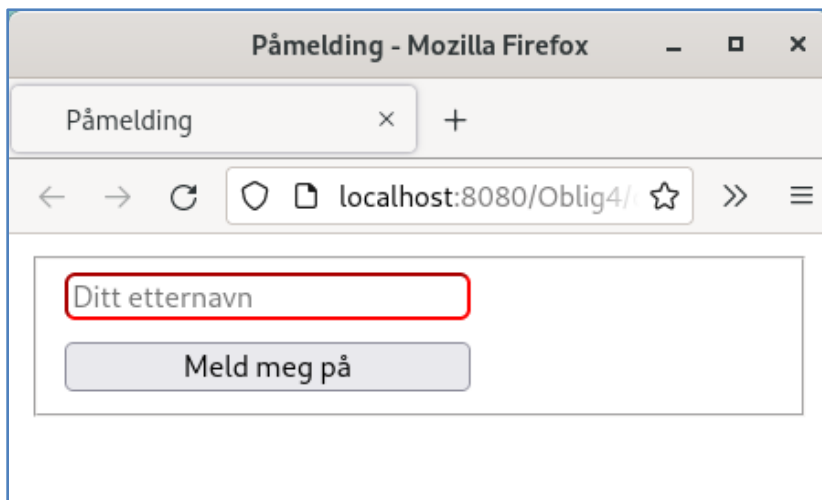
```
input[type="text"], input[type="password"] {  
    border-color: green;  
}  
  
input[type="text"]:invalid, input[type="password"]:invalid {  
    border-color: red;  
}
```

HTML attributter som *pattern*, *required*, *minlength*, *maxlength*, *min*, *max* og *type* kan brukes for å angi hva som er gyldige data i et form kontroll-element. HTML-koden nedenfor viser et input-element der vi bruker attributtene *required* og *pattern* for å angi hva som er gyldige data:

```
<input type="text"  
    required pattern="\s*\p{Lu}\p{Ll}+\s*"  
    placeholder="Ditt etternavn"  
    title="Etternavn må starte med en STOR bokstav, så små">  
</input>
```

Med den angitte CSS-en vil elementet få rød ramme når bruker skriver inn data som ikke er gyldige.

I figuren nedenfor vises HTML input-elementet fra eksempelet, og en submit-knapp.

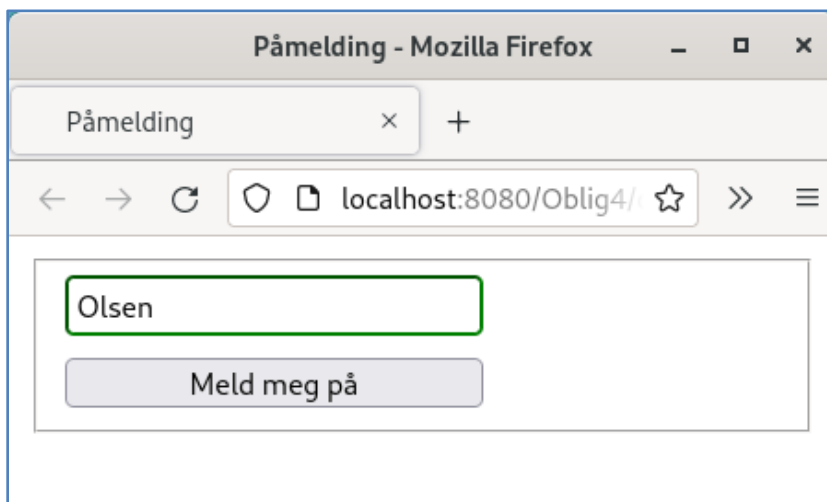
A screenshot of a Mozilla Firefox browser window titled "Påmelding - Mozilla Firefox". The address bar shows "localhost:8080/Oblig4/". The page content includes a text input field with the placeholder text "Ditt etternavn" and a submit button labeled "Meld meg på". The input field has a red border, indicating it is invalid.

Input element med ugyldige data

I figuren er tilstanden til input elementet *invalid* da ingen data er fylt inn, og da er ikke attributtet *required* oppfylt. Et tomt element stemmer heller ikke med den angitte verdien for *pattern*.

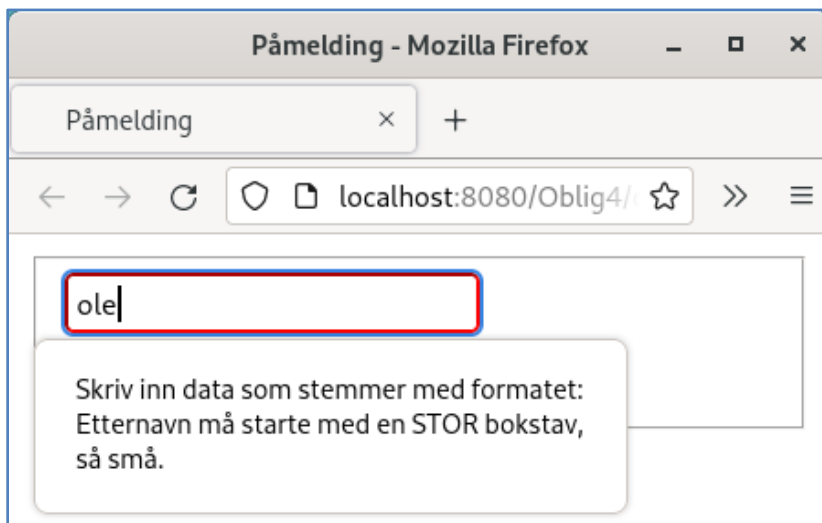
Figuren over viser også bruk av attributtet *placeholder*. Den angir tekst som vises i et tomt element.

I figuren under er gyldige data fylt inn. Tilstanden er ikke lenger *invalid*, og CSS-en viser da en grønn rammefarge på elementet.

A screenshot of the same Mozilla Firefox browser window. The text input field now contains the text "Olsen" and has a green border, indicating it is valid. The submit button "Meld meg på" remains below it.

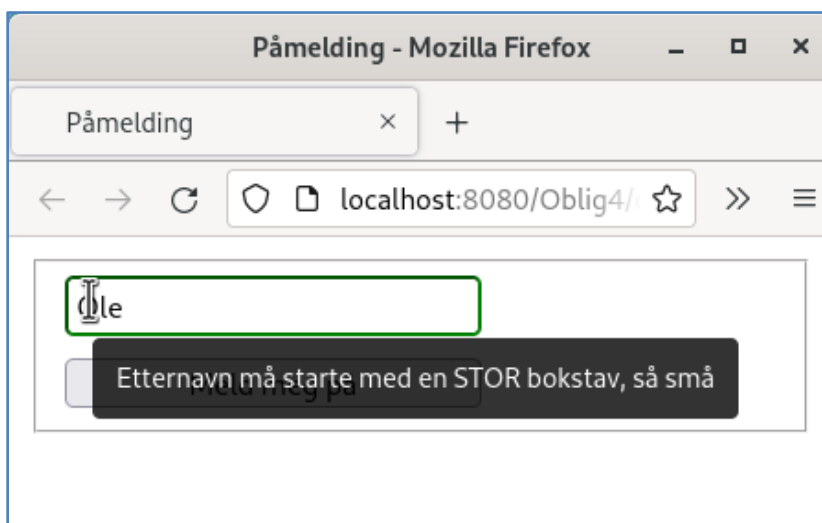
Input element med gyldige data

Hvis data er ugyldige, og submit-knappen klikkes vil nettleser automatisk kansellere submit-hendelsen. Nettleser vil ikke sende ugyldige data til tjener! Samtidig vil nettleser vise et valideringsvindu med informasjon som viser teksten fra *title* attributtet til elementet.



Valideringsvindu ved ugyldige data

Nettleser vil også vise verdien til *title* attributtet når musen holdes over elementet.



Elementet sin title vises når musen er over elementet

Nedenfor vises påmeldingsskjema til applikasjonen med både gyldige og ugyldige data.

The screenshot shows a Mozilla Firefox browser window with the title 'Påmelding - Mozilla Firefox'. The address bar shows 'localhost:8080/Oblig4/'. The page has a title 'Påmelding' and a form with the following fields:

- Fornavn: Ole (green border)
- Etternavn: Olsen (green border)
- Mobil (8 siffer): 1234 (red border)
- Passord: (yellow border)
- Passord repetert: ... (red border)
- Kjønn: ☒ mann ☐ kvinne

At the bottom of the form is a button labeled 'Meld meg på' with a mouse cursor hovering over it.

Skjema for påmelding

HTML5 attributtet *required* er brukt på alle feltene, slik at et tomt felt alltid er *invalid*. Også HTML attributtet *pattern* er brukt for å skille mellom gyldige og ugyldige data, unntatt for «Passord repetert».

En verdi for «Kjønn» skal alltid være valgt, så her er det ingen sjekk og tilstanden er alltid *valid*.

Et «Passord» som er *valid* har to undertilstander, «middels passord» og «sterkt passord». Dette skillet må gjøres med JavaScript. Applikasjonen må fortløpende sjekke verdiene som fylles inn. Det kan gjøres ved å lytte på hendelsen «input».

Kun hvis «Passord» er *valid* må JavaScript sjekke styrken på passordet. F.eks. kan du bruke at et *valid* passord med færre enn 14 tegn også er et «middels passord», ellers er det et *valid* og samtidig «sterkt passord».

Hvis et input element har tilstanden *valid* vil elementet sin egenskap *validity.valid* være sann:

```
sjekkPassordStyrke(event) {  
    const validity = event.target.validity;  
    if (validity.valid) {  
        // Sjekk for middels eller sterkt passord  
    }  
}
```

For å angi på websiden om et passord er et «middels passord» eller et «sterkt passord» skal dere bruke elementet sin *classList* egenskap og definere utseendet med CSS. Det holder å bruke en CSS klasse, f.eks. *mediumPassword*, som slås av eller på. Dere kan bruke følgende CSS:

```
input[type="password"].mediumPassword {  
    border-color: yellow;  
}
```

Feltet «Passord repetert» er valid hvis det har en ikke-tom verdi som er lik med «Passord». Sjekken for at verdiene stemmer overens må gjøres med JavaScript og ved å lytte på hendelsen «input». Husk at sjekken må gjøres både når bruker fyller data i «Passord» og i «Passord repetert».

Dersom feltene ikke stemmer overens må JavaScript-kode sette «Passord repetert» sin tilstand til *invalid*. CSS-en tidligere i oppgaven vil da gjøre rammefargen rød, og nettleser vil ved submit kansellere submit-hendelsen og i stedet vise et valideringsvindu.

JavaScript kan sette et input-felt til *invalid* ved å gi det en ikke-tom valideringsmelding. Ved hendelse «input» på «Passord repetert», der *event* er input-hendelsen kan du f.eks. bruke:

```
event.target.setCustomValidity("Repetert passord er feil!");
```

Når elementet har en ikke-tom valideringsmelding vil denne vises i stedet for elementet sin *title* i valideringsvinduet ved ugyldige data.

Tilstanden til elementet forblir *invalid* helt til meldingen blir fjernet:

```
event.target.setCustomValidity("");
```

Du skal organisere JavaScript-koden din på en god måte. Samle alle metoder for å arbeide med Form-skjema i en klasse *FORMController*, eller i et objekt. Du kan også lage en klasse *PassordValidator* som utfører selve valideringen av «Passord» og «Passord repetert». Ut over dette kan du selv velge hvordan du vil organisere koden, men unngå absolutte DOM-referanser inne i klassene. En klasse *PassordValidator* skal kun arbeide med selve passord-verdiene, og ikke vite noe om input-elementer eller websiden.

Validering av brukerinput og feilmeldinger på tjenersiden

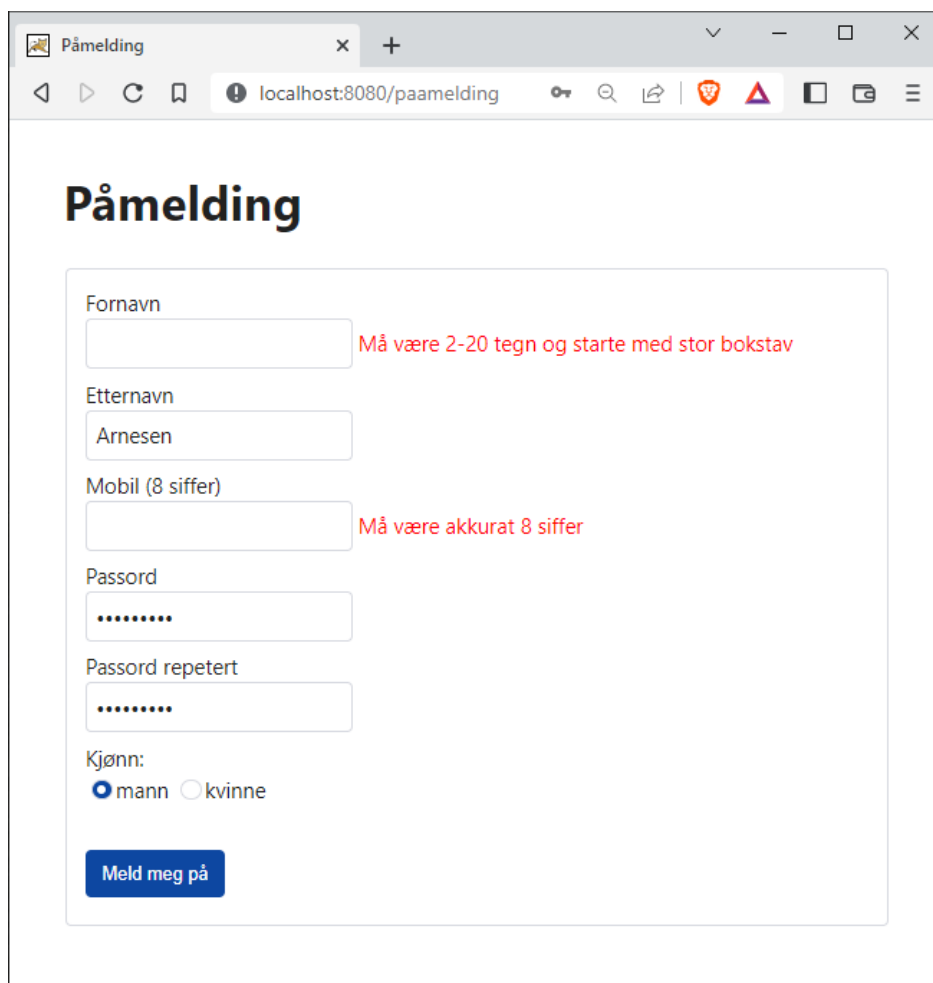
Forespørsler som inneholder inputdata fra bruker bør også alltid valideres og håndteres skikkelig på tjenersiden.

Mens validering i nettleseren handler mest om brukervennlighet, handler validering på tjeneren mest om sikkerhet og robusthet. Man kan i teorien tenke seg at data er sendt inn som har omgått valideringen i nettleseren.

Det er også ting som ikke kan valideres direkte i nettleseren, f.eks. om en bruker med gitt brukernavn er registrert i databasen eller ikke.

I vårt tilfelle skal vi gjøre validering av brukerinput som kommer i requesten både ved påmelding og innlogging.

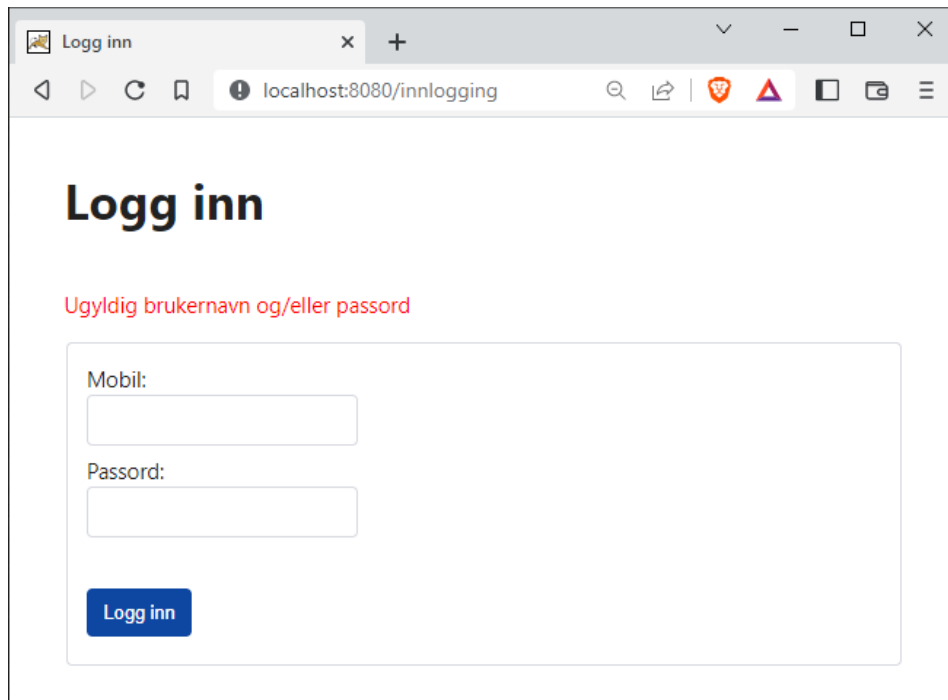
Ved ugyldige data ved påmelding skal påmeldingsskjemaet presenteres på nytt med informasjon om hva som er feil, f.eks. som vist under. Hvis du føler det blir kronglete å gi detaljerte feilmeldinger her kan enkel feilmelding for skjemaet gis i stedet, f.eks. "Deltager med dette mobilnummeret er allerede påmeldt" og (for andre feil) "Påmeldingsdetaljer er ugyldige". Tanken da er at webleser/klient skal gjøre en komplett validering, og at ugyldige data da er et resultat av "hacking".



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/paamelding'. The page title is 'Påmelding'. The form contains the following fields and validation messages:

- Fornavn**: A text input field with a red error message: 'Må være 2-20 tegn og starte med stor bokstav'.
- Etternavn**: A text input field containing the value 'Arnesen'.
- Mobil (8 siffer)**: A text input field with a red error message: 'Må være akkurat 8 siffer'.
- Passord**: A password input field showing seven dots.
- Passord repetert**: A password input field showing seven dots.
- Kjønn:**: Radio buttons for 'mann' (selected) and 'kvinne'.
- Meld meg på**: A blue button at the bottom of the form.

Ved ugyldig brukernavn (altså uregistrert mobilnummer) eller passord ved pålogging skal påloggings-skjemaet presenteres på nytt med en generell feilmelding, slik:



The screenshot shows a web browser window with the title "Logg inn". The address bar displays "localhost:8080/innlogging". The page content includes the heading "Logg inn" in a large, bold font. Below the heading, a red error message reads "Ugyldig brukernavn og/eller passord". Underneath this message is a form with two input fields: "Mobil:" and "Passord:". A blue button labeled "Logg inn" is positioned below the "Passord:" field.

Blir man sendt til innloggingssiden av andre årsaker skal dette også markeres med passende feilmelding.

Arkitekturkrav

Spring MVC og Spring Boot skal brukes som rammeverk.

MVC, EL og JSTL

Alle forespørsler inkl. "redirects" skal gå til Controllere.

All HTML skal genereres fra JSP-sider! Bruk EL, JSTL og evt. Spring sine <form:>-tagger.

For å sikre mot direkte forespørsler til JSP-sider skal alle disse plasseres under mappen /WEB-INF, der de er utilgjengelig mot direkte forespørsler fra webleser.

PRG

GET/POST skal brukes korrekt. Hvis forespørselen er en POST skal det i stedet gjøres en "redirect" til controlleren som forwarder til siden. Redirect bør også gjøres ved GET når man skal omdirigeres til en "annen" side, f.eks. ved uautorisert forespørsel.

Passord, innlogging og tilgangskontroll

Passord, innlogging og tilgangskontroll skal gjøres slik dere har lært om korrekt bruk og lagring av passord. Altså: Oppgitt passord skal valideres for minimum styrke, saltes og hashes (f.eks. med SHA256) før det lagres i database. Sjekk av passord gjøres ved at oppgitt passord går gjennom samme prosess og sammenlignes med lagret passordhash.

Lagring av data

Data i applikasjonen er i hovedsak påmeldte deltagere. Det som er naturlig å lagre om hver påmeldt deltager er:

- mobil (unik id)
- passord, dvs.
 - hash
 - salt
- fornavn
- etternavn
- kjønn

Dere skal lagre data i PostgreSQL-databasen deres på *azure.com* (eller *localhost* hvis dere ikke får tilgang) og bruke Spring Data JPA til å aksessere dataene fra applikasjonen.

Testkrav

Innleveringen SKAL inneholde litt **enhetstesting** med JUnit, minimum 5 ulike "asserts" om hvordan ulike ting i applikasjonen er forventet å virke.

Ting man kan teste kan f.eks. være:

- Noe av inputvalideringen ved påmelding (deklarativ / programmatisk)
- Hjelpklasser, f.eks. for håndtering av pålogging / autentisering
- Service-klasser som behandler data inn/ut mellom Controller og Repository
- Feilhåndtering i Controllere?
- ... andre ting dere kommer på ...

Det er **ikke** et krav at man skal utføre black box- eller integrasjonstester.

Deploymentkrav

Nei. Det er tilstrekkelig at applikasjonen kjøres på egen maskin, og at virkemåten dokumenteres i innlevert pdf via kopier av skjermbilder.

Bjarte og Lars-Petter