

Oblig4 del2 – Spring MVC

Sist oppdatert av Lars-Petter Helland 26.10.2025

For å unngå opphoping av arbeid i slutten av semesteret, har vi delt opp Oblig4 i 2 innleveringer (del1 og del2). Del2 er en utvidelse av del1. Dette er oppgaveteksten for del2.



Innleveringsfrist for Oblig4 del2 er tirsdag 11. november. Vi har som mål å rette innleveringene og godkjenne innen 2 uker etter frist.



Bruk samme grupper som i del1 !



Lever på samme måte som i del1 !

Nytt i denne delen (del2)

- Påmeldte deltagere skal lagres i en database (ref. **F20 - Spring DI og Data JPA**)
- Innlogging og utlogging (ref. **F21 - Sesjoner Innlogging EL JSTL**)
- Autentisering og autorisasjon (ref. **F21 og F22 - Passord og autentisering**)
- Korrekt sikker lagring av passord i database (ref. **F20 og F22 - Passord og autentisering**)
- Bedre applikasjonsstruktur (ref. **F20 - Spring DI og Data JPA**)
- Litt mer testing (ref. **F24 - Enhetstesting og mocking**)
- Bedre feilhåndtering
- Små oppdateringer i brukergrensesnittet

Ressurser til øvingen

I tillegg til malene fra del1, får dere nå også en hjelpeklasse for passord-håndtering **PassordService.java** som skal brukes i løsningen. Det er også lagt ved en **pom.xml** som viser hvilke avhengigheter vi har til eksterne bibliotek. Kopier disse inn i deres egen pom.xml.

Oppgaven: Web-applikasjon Påmelding til fest

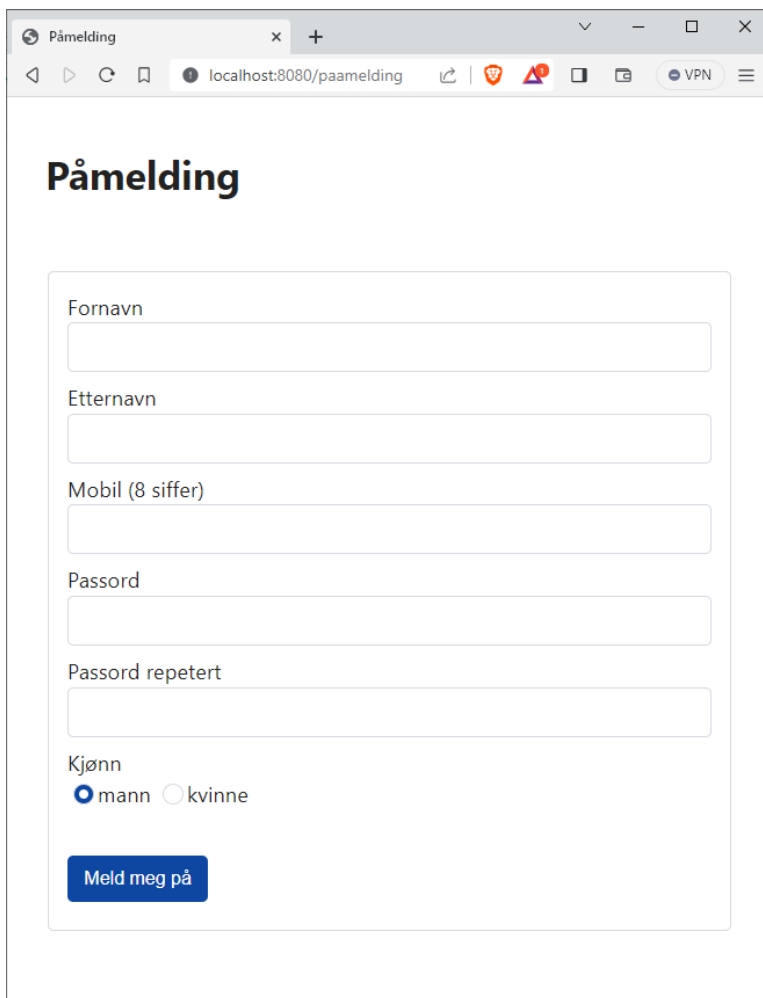
Det skal lages en enkel web-applikasjon som kan brukes til påmelding til fest.

Her er brukstilfellene i grove trekk:

1. En bruker skal kunne melde seg på festen.
2. En **påmeldt** bruker skal kunne se listen over påmeldte, altså **det er nå et krav at du må være innlogget som påmeldt bruker for å kunne se listen**.

Påmelding

Skjema for påmelding skal se ca. slik ut:



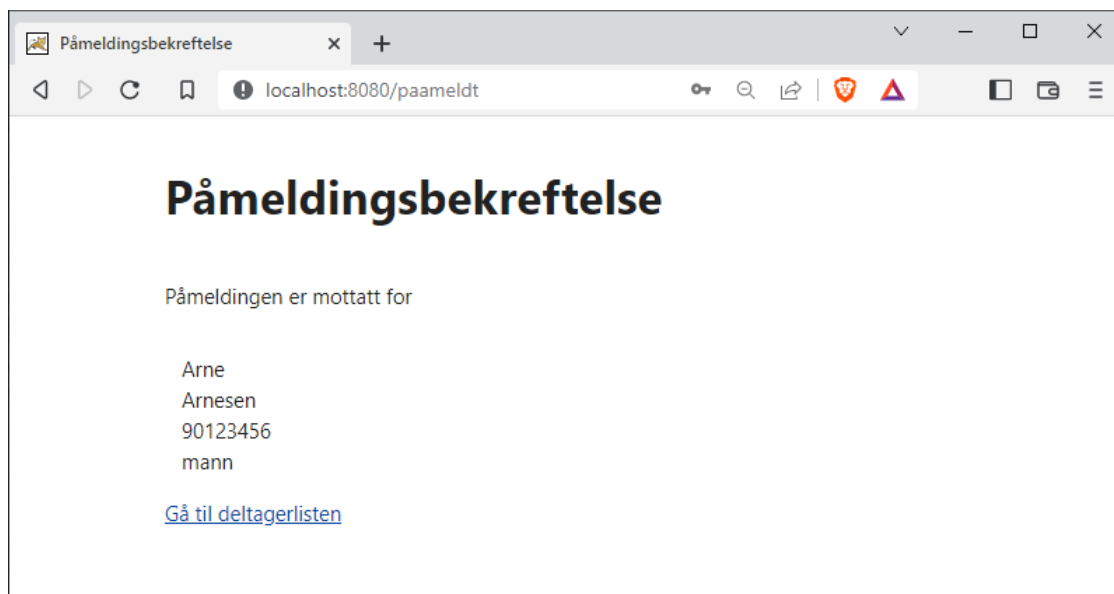
The screenshot shows a web browser window with the address bar displaying 'localhost:8080/paamelding'. The page title is 'Påmelding'. The form contains the following fields and controls:

- Fornavn: Text input field.
- Etternavn: Text input field.
- Mobil (8 siffer): Text input field.
- Passord: Text input field.
- Passord repetert: Text input field.
- Kjønn: Radio buttons for 'mann' (selected) and 'kvinne'.
- Meld meg på: Blue button.

Alle felter er obligatoriske, og bør helst valideres på klientsiden (i nettleser) via HTML/CSS/JavaScript. Det er opp til dere hvor mye dere gjør ut av dette. Poenget er å gi god hjelp til bruker ved utfylling av skjemaet + sikre at skjemaet helst ikke kan sendes av gårde før visse minimumskrav er oppnådd.

All brukerinput SKAL deretter også valideres på tjeneren. Mer detaljer om valideringsregler senere i oppgaveteksten.

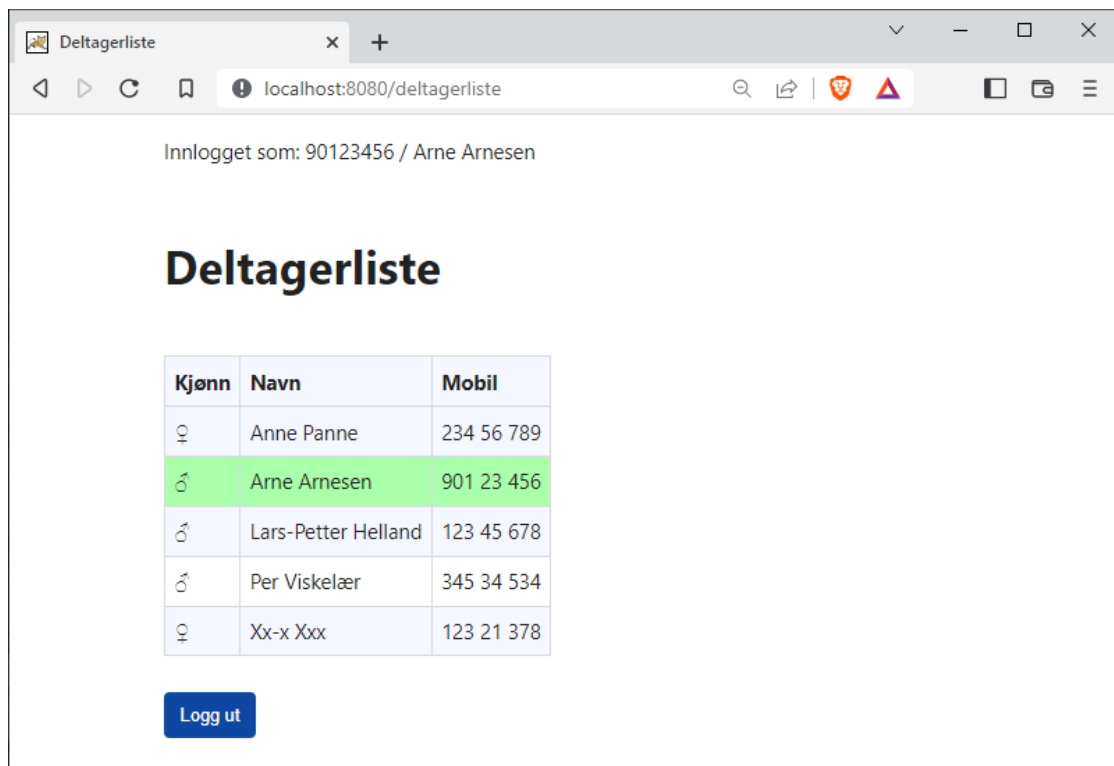
Hvis alt er OK får man en bekreftelse på at påmeldingen er registrert, slik:



Mobilnummeret blir "brukernavn" i applikasjonen, se senere brukstilfelle. Ved vellykket påmelding er man automatisk "logget inn" med mobilnummer som brukernavn.

[Se deltagerlisten](#)

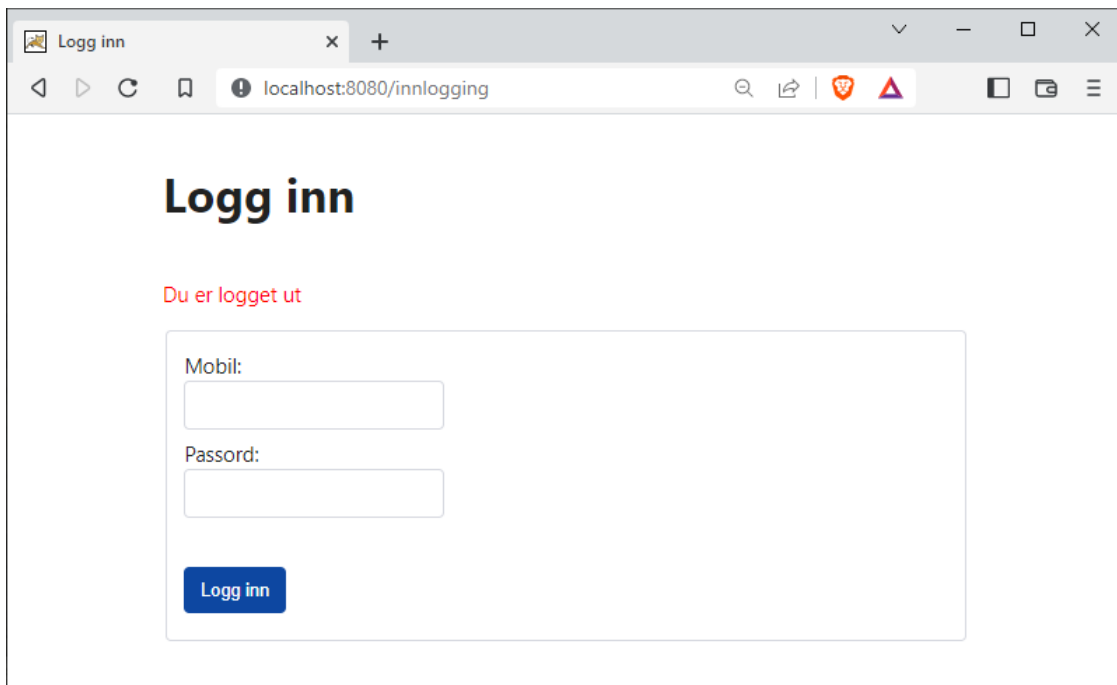
Når man trykker på lenken "Gå til deltagerlisten" kommer man til Deltagerlisten:



Ting å merke seg i deltagerlisten:

- Øverst skal det stå det hvem man er innlogget som.
- Deltagerlisten er sortert stigende på fornavn, deretter etternavn.
- Innlogget deltager er uthevet med grønn bakgrunn.
- Å kunne se deltagerlisten krever at man er innlogget som påmeldt deltager. Sikre mot uautorisert tilgang! Forsøk på uautorisert tilgang gir innloggingssiden (under) med feilmelding.
- Når man trykker på knappen "Logg ut" logges man ut og går til innloggingssiden med en melding om at man er logget ut:

Innloggingssiden



Logg inn

Du er logget ut

Mobil:

Passord:

Logg inn

Fra innloggingssiden kan du logge inn som påmeldt deltager for å se deltagerlisten.

Ved vellykket innlogging kommer man til deltagerlisten (som 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 på tjeneren (Java/Spring) er for å sikre at dataene som mottas og lagres er gyldige. Her må man også f.eks. sjekke om en deltager med gitt mobil (unik id) allerede er påmeldt.

TIPS: Manuell testing av validering på tjeneren kan f.eks. gjøres ved å strippe HTML-en litt og kommentere ut JavaScript-et under testing. En annen måte å teste er JUnit, se senere.

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!
- **Passord** bør ha en viss minimumslengde. Krav utover dette bestemmer du selv.
- **Repetert passord** må være likt passordet. (Ikke så farlig på server. Klientsidevalidering av denne bør gjøres !!)
- **Kjønn** må være "mann" eller "kvinne".

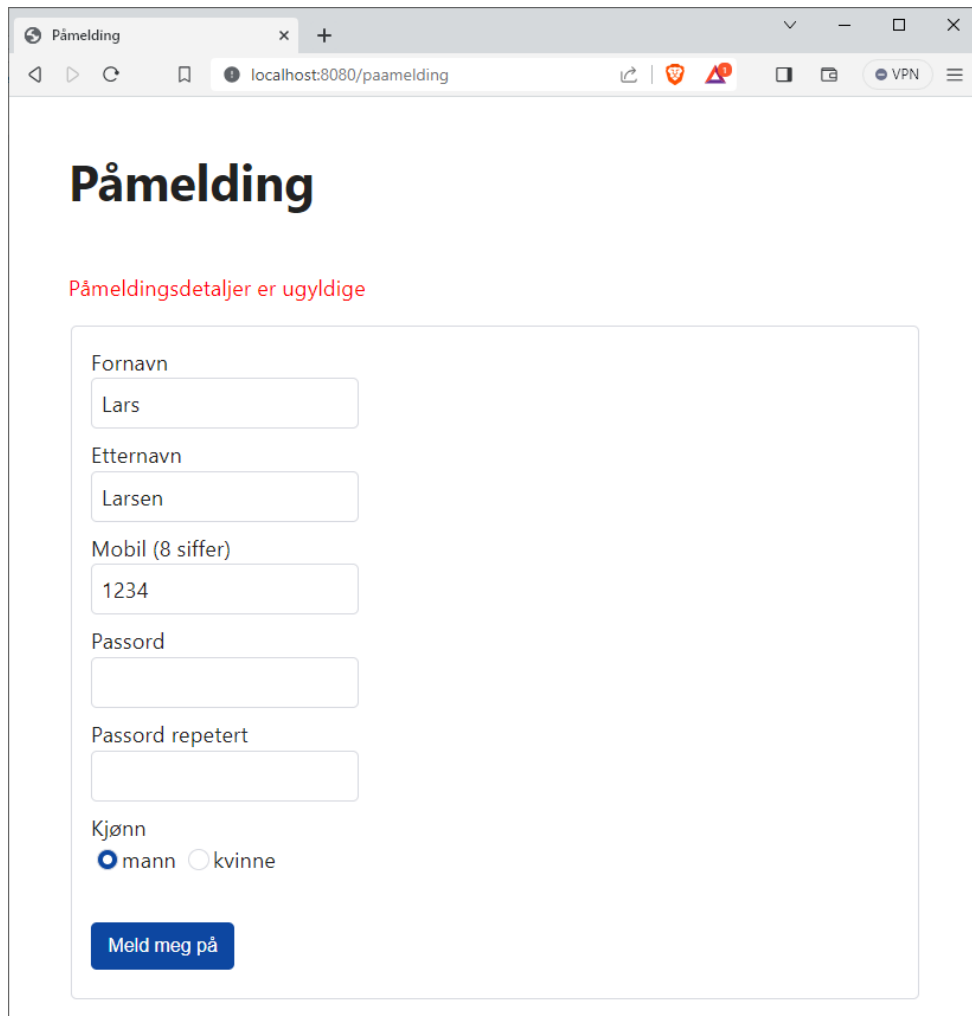
Validering av brukerinput og feilmeldinger på klientsiden

Som tidligere nevnt kan dere selv velge hvordan og hvor mye dere legger i dette.

Validering av brukerinput og feilmeldinger 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.

Ved ugyldige data ved påmelding skal påmeldingsskjemaet presenteres på nytt med en enkel feilmelding, f.eks. "Deltager med dette mobilnummeret er allerede påmeldt" og (for andre feil) "Påmeldingsdetaljer er ugyldige" (se nedenfor).



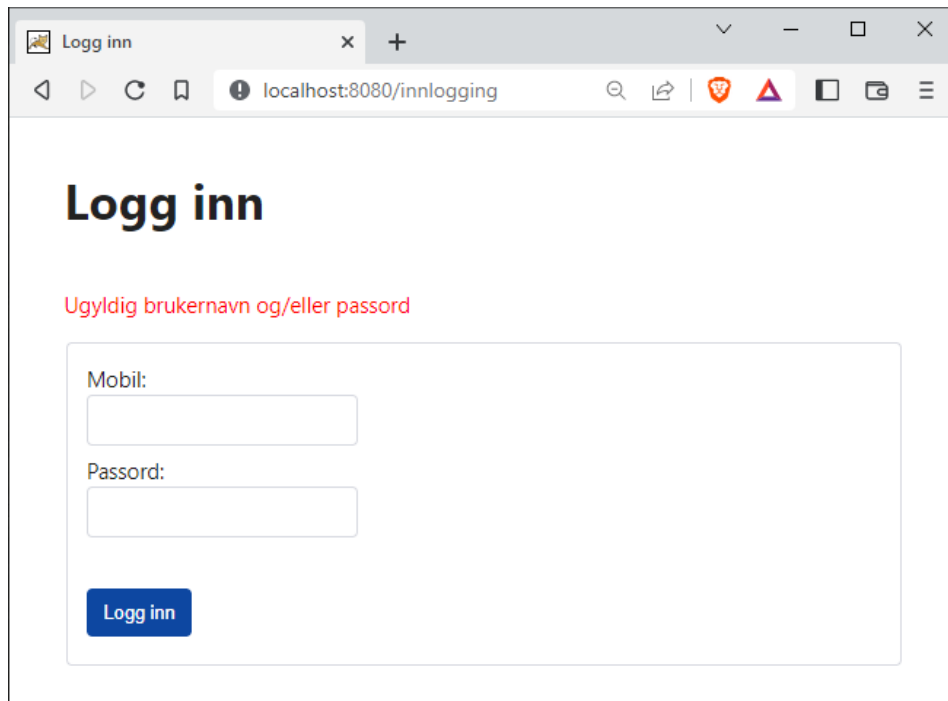
The screenshot shows a web browser window with the title "Påmelding" and the address bar displaying "localhost:8080/paamelding". The page content includes a large heading "Påmelding" and a red error message "Påmeldingsdetaljer er ugyldige". Below the error message is a registration form with the following fields and options:

- Fornavn: Text input containing "Lars"
- Etternavn: Text input containing "Larsen"
- Mobil (8 siffer): Text input containing "1234"
- Passord: Text input (empty)
- Passord repetert: Text input (empty)
- Kjønn: Radio buttons for "mann" (selected) and "kvinne"
- A blue button labeled "Meld meg på"

Dere kan velge å gi mer spesifikke feilmeldinger om dere ønsker det, men det er ikke et krav.

Tanken er at nettleser/klient skal gjøre en komplett validering med hjelp til bruker, og at ugyldige data i requesten da kan antas å skyldes "hacking".

Ved innlogging: Hvis brukernavn er ugyldig (altså uregistrert mobilnummer) eller passord ikke stemmer, skal innloggings-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 a large heading "Logg inn", a red error message "Ugyldig brukernavn og/eller passord", and a login form. The form contains two input fields labeled "Mobil:" and "Passord:", and a blue button labeled "Logg inn".

Blir man sendt til innloggingssiden av andre årsaker skal dette også markeres med passende feilmelding (som nevnt tidligere).

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 JSP Expression Language (EL) og JSP Standard Tag Library (JSTL).

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

PRG

GET/POST skal brukes korrekt.

Hvis forespørselen er en POST skal det ikke gjøres en direkte videreformidling til view, men i stedet en "redirect" til controlleren for dette viewet (Post-Redirect-Get).

Redirect bør også gjøres ved GET når man skal omdirigeres til en "annen" side, f.eks. ved ugyldige data ved registrering.

Lagring av påmeldte deltagere / databasetabell

Dere skal lagre data i PostgreSQL-databasen deres på *azure.com* og bruke Spring Data JPA til å aksessere dataene fra applikasjonen.

Data i applikasjonen er kun de påmeldte deltagerne. Dette kan lagres slik i databasen:

```
CREATE TABLE deltager (  
    mobil          CHARACTER (8) PRIMARY KEY,  
    hash           CHARACTER (64) NOT NULL,  
    salt           CHARACTER (32) NOT NULL,  
    fornavn        CHARACTER VARYING (40),  
    etternavn      CHARACTER VARYING (40),  
    kjonn          CHARACTER VARYING (6)  
);
```


Entitet(er) i Java

Her er nok det mest nærliggende å kun ha én entitetsklasse **Deltager**.

Men hvis dere ønsker det, er det også mulig å gruppere sammen det som tilhører passordet i en egen klasse **Passord**, og bygge denne inn i Deltager, slik:

```
@Embeddable
public class Passord {
    private String hash;
    private String salt;
    ...
}

@Entity
@Table(schema = "...")
public class Deltager {
    @Id private String mobil;
    @Embedded private Passord passord;
    private String fornavn;
    private String etternavn;
    private String kjonn;
    ...
}
```

Passord, innlogging og tilgangskontroll

Bruk klassen **PassordService** som er vedlagt oppgaven. Ved oppretting av ny bruker i databasen er det **passord-hash** og **-salt** som skal lagres, ikke passordet i klartekst!!! Det kan gjøres ca. slik:

```
String salt = passordService.genererTilfeldigSalt();
String hash = passordService.hashMedSalt(passordKlartekst, salt);
Passord passord = new Passord(hash, salt);
...
```

Ved pålogging vil sjekk av passord gjøres ved at oppgitt passord går gjennom samme prosess som ved opprettelse og sammenlignes med lagret **passordhash**. Kan gjøres ca. slik:

```
Deltager deltager = ...; //Hentes fra databasen
boolean riktigPassord = passordService.erKorrektPassord(
    oppgittPassord, deltager.getSalt(), deltager.getHash());
...
```

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 inputvalideringen ved påmelding (deklarativ / programmatisk)

Krav til deployment

Det er tilstrekkelig at applikasjonen kjøres på egen maskin, og at virkemåten dokumenteres i innlevert pdf via kopier av skjermbilder. Det er frivillig å deploye på en annen webtjener, da vi ikke fikk gjennomgått og øvd på dette som planlagt.

Lykke til!

Lars-Petter