# Projektuppgift IT- och informationssäkerhet

- Använd ett gemensamt github-repository för hantering av koden
- Försök att dela upp arbetet på ett lämpligt vis
- Varje dag som vi har lektioner stämmer jag av med gruppledaren hur arbetet går, detta är obligatoriskt. Om inte gruppledaren kan får denne delegera till annan person i gruppen.
- Schema för tider för avstämningar finns här: https://docs.google.com/spreadsheets/d/1i-tyoSOaQBMqXZUs0kVovZiY2HOeEkkdOUMoU7E4Ms8/edit?usp=sharing
- Om ni behöver byta tid, kolla med annan grupp och skriv in i dokumentet
- Avslutningsvis ska varje grupp lämna in en länk till github-repositoryt som innehåller koden

# Översikt

- Ni ska skapa en säker webbapplikation med hjälp av Spring boot och Spring security
- Webbapplikationen har in- och utloggningssida, förstasida, registreringssida, sida för borttagning av användare, och - för betyg VG - sida för uppdatering av användare
- Sidorna ska vara HTML-sidor och skapas med hjälp av thymeleaf
- Registrering, borttagning och uppdatering av användare är funktioner som bara ADMIN-användaren får utföra
- Förstasidan får även en USER-användare se
- Loggning ska utföras i applikationen - email-adresser ska anonymiseras med hjälp av funktion i en hjälpklass
- Testning ska utföras i applikationen

# Instruktioner betyg G

- Ni behöver följande dependencies: Spring web, spring security, thymeleaf, validation (hittas under I/O när man skapar projektet i IntelliJ Ultimate)
- Applikationen är en traditionell webapplikation, inte en REST-applikation, därav användandet av thymeleaf och @Controller-annotationer
- Ni behöver skapa upp en @Bean för SecurityFilterChain, en UserDetailsService och en PasswordEncoder
- För godkänt betyg används en InMemoryUserDetailsManager-@Bean som implementation av UserDetailsService och hantering av användare
- @Bean InMemoryUserDetailsManager används då för att skapa upp en ADMIN-användare från start. Denna användare är den som framför allt används i applikationen

# Instruktioner betyg G

- Applikationen ska ha följande funktioner:
  - En login-sida (egen eller med Spring securitys inbyggda formLogin)
  - En första-sida (för localhost:8080/) som skriver ut att man är inloggad
  - En registreringssida. Med csrf-skydd
  - En sida för borttagning av en användare. Med csrf-skydd
  - En sida som visas när man loggat ut och som länkar till inloggningssida

# Instruktioner betyg G

- Första-sidan (/) får ses av alla autentiserade användare, både ADMIN och USER, sidan ska innehålla en knapp för utloggning (se exempel i Lektion 7 på Totara, oauth2_med_spring_ovning.pdf)
- Registrering: dessa sidor får bara ses av användare med ADMIN-roll.
    - Endast admin kan registrera nya användare, nya användare har rollen USER.
    - Det ska finnas en sida för registrering av ny användare
    - Det ska finnas en sida som visas när registrering lyckats (även denna får bara admin se)
    - Validering av email och lösenord ska göras i DTO-objekt (se Lektion 5 på Totara) och om det blir valideringsfel ska dessa visas på registreringssidan

# Instruktioner betyg G

- Borttagning av användare: dessa sidor får bara ses av användare med ADMIN-roll
  - Endast admin kan ta bort en användare (ej sig själv)
  - Det ska finnas en sida för borttagning av användare med ett fält för inmatning av befintlig användares email
  - Det ska finnas en sida som visar när borttagning lyckats
  - Om användaren inte hittas ska en felsida visas med meddelande om att användaren inte kunde hittas
  - Tips: det finns passande metoder i InMemoryUserDetailsManager för att kolla om användaren finns, samt för att ta bort användare
- Registrerings- och borttagningsfunktionerna ska använda klassen HTMLUtils för att 'tvätta bort' eventuell skadlig kod från email och password.
- Registreringsfunktionen ska hasha lösenord innan de sparas

# Instruktioner betyg G

- Utöver dessa sidor och funktioner ska detta implementeras:
  - En hjälp-klass (ex med namnet MaskingUtils) med en statisk metod som anonymiserar en användares e-mail innan den loggas
  - Metoden ska förvandla en vanlig e-mail-adress till följande format:

```
//visa bara första och sista bokstaven av användarnamnet
h**********n@gmail.com
```

# Instruktioner betyg G

- Skapa en konfigurationsfil som loggar dina klassers logg-meddelanden till en separat fil (se Lektion 7 loggning.pdf på Totara)
- Loggning i alla endpoints. När en användare inte kan hittas ska detta loggas med WARN-loggning, ex:

```
    logger.warn("Användare med mejl-adress: " + MaskingUtils.anonymize(user.getEmail()) + " kunde inte hittas");
```

- Minst en debug-loggning ska göras i varje endpoint och som förklarar vad endpointen gör och vilken data som behandlas. I exempelvis endpointen som tar bort en användare kan detta loggas:

```
    logger.debug("Borttagning av användare med mejl: " + MaskingUtils.anonymize(user.getEmail()) + " utförs.");
```

# Instruktioner betyg G

- Testning av följande endpoints med klassen MockMvc: första-sidan, registreringsfunktionen, samt borttagningsfunktionen (se Lektion 4 övning 2, samt Lektion 5 övning 1 på Totara för info om testning med MockMvc)
- Under testning av POST-requests måste csrf-skyddet tas bort i din SecurityFilterChain. Dessutom måste formLogin tas bort och ersättas med httpBasic-inloggning (se Lektion 4 övning 2 på Totara för info om detta)
- Första-sidan (/) ska testas med och utan autentiserad användare
- Registreringsfunktionen ska testas med framgångsrik registrering och med en misslyckad registrering
- Borttagningsfunktionen ska testas med framgångsrik borttagning och misslyckad borttagning

# Klasskommentarer

- Ni ska slutligen kommentera era klasser med en klasskommentar som förklarar vad just denna klass gör
- Lägg vikt vid att kommentera och förklara vad @Configurations-klassen gör och beskriv även de @Beans som finns i denna klass
- Klasskommentarer läggs in ovanför klassnamnet på det här sättet:

```
/**
 * Info om HakansClass här, bla bla bla, så här funkar den klassen osv.
 * Mer info om denna klass..
 */
public class HakansClass {
}
```

# Instruktioner för betyg VG

- Användning av H2-databas istället för InMemoryManager
- Därför behövs två till dependencies: Spring Data JPA och H2 database
- Skapa en egen User-klass (som gärna kan heta AppUser för att undvika namnkonflikter), denna klass sparas i databasen
- Klassen ska, förutom fälten username och password även ha fälten: id, firstname, lastname, age och role
- Dessa fält ska även finnas med vid registrering och i DTO-objektet
- Alla fält ska valideras på passande vis vid registrering och uppdatering
- Implementering av UserDetailsService-klass som ansvarar för hämtning av användare från databasen

# Instruktioner för betyg VG

- Lägg till en funktion för uppdatering av användares lösenord
- Uppdatering av lösenord: dessa sidor får bara ses av användare med ADMIN-roll
  - Det ska finnas en sida för uppdatering av användares lösenord med fält för inmatning av befintlig användares email och ett nytt lösenord
  - Validering ska göras med hjälp av en särskild DTO-klass (med endast fälten användarnamn och lösenord) och ev valideringsfel visas i uppdateringssidan
  - Det ska finnas en sida som visar när uppdatering lyckats
  - Om användaren inte hittas ska en sida visas med meddelande om att användaren inte kunde hittas
  - För att uppdatera en användare behöver ni först hämta användaren, därefter uppdatera lösenordet, därefter spara användaren igen. Denna logik läggs i UserService-klassen, se nästa sida.

# Instruktioner för betyg VG

- En UserService-klass ska skapas och annoteras med @Service. Denna klass fungerar som mellanhand mellan din @Controller och ditt databas-@Repository. @Controller-klassen har tillgång till UserService. UserService har tillång till databas-@Repository:t. Ex: när en användare ska registreras skickar @Controllern vidare DTO-objektet till metod i UserService, som i sin tur kallar på metod i @Repository:t.
- Vid uppdaterings- och borttagningsfunktionerna ska UserService ansvara för kasta ett UsernameNotFoundException om användaren inte hittas.
- Detta exception ska fångas i din @Controllers metod med try/catch. I catch:blocket ska ni logga en varning (logger.warn) och förutom ett meddelande även logga stacktracet från exceptionet. Därefter skickar ni användaren till sidan som meddelar att användaren inte hittas, precis som beskrivet tidigare.

# Instruktioner för betyg VG

- Förutom dessa ändringar och tillägg följer ni instruktionerna för betyg G