## Applicazioni Internet 2019-20

# Esercitazione 1

Un'applicazione web prevede una parte pubblica, accessibile a tutti, ed una privata soggetta a verifica delle credenziali di accesso.

1. Si crei un progetto Spring Initializr includendo i moduli Lombok, Spring Web, Thymeleaf

La URL / è pubblica e contiene un collegamento alla pagina di registrazione per chi non dispone di credenziali e un form per l'inserimento di utente e password, per chi è già registrato.

2. All'interno del progetto si crei una classe chiamata HomeController e la si annotti con @Controller. Tale classe deve contenere un metodo (home() ) annotato con @GetMapping("/"). Esso ritorna una stringa che indica il nome del template che dovrà essere caricato quando viene ricevuta una richiesta per la URL indicata. Nella cartella resources/templates si crei un file HTML con lo stesso nome ritornato dal metodo home(). Si modifichi il contenuto per inserire il collegamento alla pagina di registrazione ed il modulo di login. Si lanci il progetto e si verifichi che alla URL http://localhost:8080/ venga mostrata la pagina creata

La pagina di registrazione (che risponde alla URL /register) presenta un modulo per conferire un insieme di dati quali nome, cognome, indirizzo e-mail, password prescelta (inserita due volte per verifica) e consenso di accettazione delle norme sulla privacy. I vari campi sono soggetti a vincoli di lunghezza minima e massima e a correttezza formale nel caso dell'indirizzo e-mail.

3. Si aggiunga il metodo registrationPage() alla classe HomeController. Tale metodo deve rispondere alla URL "/register" quando viene richiesta tramite il metodo GET (inserire annotazione opportuna). Si aggiunga il template corrispondente, lo si modifichi inserendo un modulo per la raccolta dati e rilanci il progetto, verificando che è possibile navigare dalla pagina principale a quella di navigazione.

Compilando tale modulo, i dati sono inviati al server (URL /register, metodo POST) che verifica il rispetto dei vincoli, l'uguaglianza delle due password, l'espressione del consenso sulla privacy, l'assenza di utenti con la stessa email già registrati sul sistema.

4. Si aggiunga alla classe HomeController l'annotazione @Log(topic = "HomeController"), dove Log appartiene al package lombok.extern.java. Tale annotazione rende disponibile a tutti i metodi della classe l'oggetto "log" che offre metodi per emettere righe di log etichettate con l'argomento HomeController.
   Alla classe HomeController si aggiunga il metodo register(). Tale metodo deve accettare i dati inviati alla URL "/register" con il metodo POST (annotazione). Inizialmente tale metodo non ha parametri, stampa un messaggio di log sulla console e ritorna la stringa "redirect:/". Tale stringa indica che al termine dell'elaborazione occorre rispondere al browser con il codice 302 – Temporary Redirect indicando come nuova URL "/". Questo server ad implementare il pattern "POST then GET" che evita, se l'utente cerca di ricaricare la pagina frutto di una POST, che vengano inviati nuovamente i dati al server (la pagina ricaricata sarà infatti quella a cui si è stati ridiretti, che viene caricata sempre con GET). Verificare che inviando qualsiasi dato dalla pagina di registrazione si venga effettivamente rimandati alla home page, che venga emessa una riga di log sulla console e che eventuali tentativi di ricaricare la pagina ottenuta dall'invio dei dati di registrazione non determinino ulteriori righe di log.

In caso positivo aggiunge alla lista degli utenti che tiene internamente (in una mappa singleton contenuta in memoria – senza per il momento usare una base dati) un record con i dettagli dell'utente e comunica che la registrazione è andata a buon fine.

5. Si crei una classe chiamata RegistrationCommand. Tale classe deve contenere le informazioni che sono inserite nel modulo di registrazione. Si annoti tale classe con @Data (di Lombok).
   Ai metodi registrationPage() e register() della classe HomeController, si aggiunga un parametro di tipo RegistrationCommand, preceduto dall'annotazione @ModelAttribute("command"). Tale annotazione provvede ad inserire nel ViewModel che verrà passato al motore di gestione delle viste un oggetto di tipo RegistrationCommand cui le viste potranno accedere usando la chiave "command". Si modifichi il corpo di tali metodi stampando in ciascuno di essi una riga di log che indichi il contenuto del parametro.
   Si modifichi il template della pagina di registrazione aggiungendo nel tag <html> il riferimento al namespace thymeleaf (xmlns:th=www.thymeleaf.org) e si modifichi il template di registrazione per fare riferimento ai dati contenuti nel ViewModel. Si faccia riferimento alla guida riportata come primo riferimento in calce a questo documento per impostare il contenuto del tag <form> e quelli dei singoli campi di tipo <input>.
   Si esegua il progetto e si verifichi che navigando alla pagina di registrazione viene stampato nel log un oggetto vuoto e che inviando un form con dati di qualche tipo, questi vengono correttamente passati al metodo register(…).
   Si crei la classe RegistrationDetails che contiene i campi di RegistrationCommand (tranne la seconda copia della password) e un campo di tipo Date chiamato registrationDate. Si annoti tale classe con @Value e con @Builder (entrambi del package lombok). Questa classe sarà usata per memorizzare le informazioni di un singolo utente registrato sul sistema.
   Si crei anche la classe RegistrationManager che extende ConcurrentHashMap<String, RegistrationDetails>. Tale classe ha il compito di memorizzare (in modo thread-safe) la lista degli utenti registrati. Si annoti tale classe con @Component, così da renderne disponibile un'istanza come singleton che possa essere iniettata dove occorre.
   Nella classe HomeController, si aggiunga un attributo autowired di tipo RegistrationManager. Utilizzando il pattern builder, si crei nel metodo register(…) un'istanza di RegistrationDetails estraendo i dati rilevanti dal parametro di tipo RegistrationCommand ed aggiungendo la data corrente. Si utilizzi il metodo putIfAbsent(…) usando lo username come chiave e controllando che il metodo ritorni null. In caso contrario si rimandi alla pagina "registration".
   Si verifichi che, lanciando il programma, sia possibile inserire un dato username una sola volta (tornando alla pagina principale), mentre eventuali ulteriori tentativi di usare lo stesso nome lasciano l'utente nella pagina di registrazione.

In caso di errore, rimanda alla pagina di registrazione, popolando i diversi campi del modulo con i valori precedentemente riempiti dall'utente, indicando i diversi tipi di errore che si sono verificati.

6. Seguendo le indicazioni contenute nel secondo riferimento riportato in calce, si aggiungano alla classe RegistrationCommand annotazioni sui singoli metodi indicando la dimensione minima richiesta (@Size(min=…)) o la necessità di esprimere un indirizzo valido di posta elettronica (@Email).
   Nel metodo register(…) di HomeController, si annoti il parametro di tipo RegistrationCommand anche con l'attributo @Valid e lo si faccia seguire, IMMEDIATAMENTE, da un nuovo parametro di tipo BindingResult. La presenza dell'annotazione fa sì che il framework Spring provveda a verificare, prima di invocare il metodo, se i dati memorizzati nei singoli attributi dell'oggetto RegistrationCommand rispettano i vincoli sintattici espressi tramite le annotazioni sui singoli campi. In assenza del successivo parametro, eventuali discrepanze lancerebbero un'eccezione. La presenza del parametro di tipo BindingResult fa sì che tali errori siano descritti al suo interno.
   Si modifichi il corpo del metodo register(…) così da controllare se le due password sono uguali o meno. In caso di differenze, aggiungere, all'oggetto BindingResult un errore (addError(new FieldError(<nome oggetto>,<nome campo>,<messaggio di default>) ) che indichi il problema riscontrato. Si ripeta l'operazione anche per il campo della privacy (che deve risultare spuntato).
   Si verifichi se sono presenti errori (usando il metodo hasErrors() dell'oggetto BindingResult) nel qual

caso si rimandi alla pagina “registration”.
Nel codice già presente che prova ad aggiungere alla mappa delle registrazioni il nuovo oggetto, si aggiunga – in caso di utente già presente – un errore all’oggetto BindingResult che segnali la natura del problema e si rimandi alla pagina “registration”.
Si modifichi il template della pagina registration.html per includere accanto ai singoli campi di ingresso, gli eventuali messaggi di errore loro associati.
Si verifichi che tutto funziona come atteso.

Nella parte pubblica è presente anche la pagina per la verifica delle credenziali di accesso (URL: /login, metodo POST). Compilando correttamente il modulo con l’indirizzo e-mail e la password scelti in fase di registrazione, il server abilita l’accesso alla parte privata (URL: /private), memorizzando nella sessione corrente l’indicazione che l’utente è autorizzato. Se le credenziali sono errate, si rimane sulla pagina di login. Eventuali tentativi di accesso alla sezione privata senza avere conferito delle credenziali valide devono essere ridiretti verso la pagina principale.

7. Si aggiunga la classe LoginCommand, annotata con @Data, analoga per scopi e meccanismi a quanto fatto per @RegistrationCommand.
   Si aggiunga in HomeController il metodo login() che riceve come parametri un oggetto di tipo LoginCommand e un oggetto di tipo HttpSession. Tale metodo deve rispondere alla URL “/login” richiesta con il metodo POST. Al suo interno, validare che l’oggetto LoginCommand abbia un campo utente non nullo e usare tale campo per trovare l’oggetto RegistrationDetails corrispondente tramite le funzioni offerte dall’oggetto RegistrationManager. Si verifichi la correttezza della password indicata e, se tutti i controlli sono positivi, si aggiunga all’oggetto HttpSession l’attributo “username” indicando come valore quello contenuto nel LoginCommand e si ridiriga verso la URL “/private”. In caso contrario si ridiriga verso la URL “/”.
   Si aggiunga il metodo privatePage(…) che accetta come parametro un oggetto di tipo HttpSession. Si verifichi che tale parametro contenga un attributo “username” non nullo e si ritorni il riferimento al template “private”. In caso contrario si rimandi alla pagina “/”.
   Si aggiunga il template “private.html”.

Nella sezione privata è presente un form (che rimanda alla URL /logout, metodo POST) che permette di eliminare dalla sessione l’informazione di accesso e rimanda alla pagina principale.

8. Si aggiunga ad HomeController il metodo logout(…) che accetta un parametro di tipo HttpSession. Tale metodo rimuove dalla sessione l’attributo “username” e rimanda alla pagina “/”. Si annoti il metodo in modo da farlo rispondere alla URL /logout con il metodo post.
   Si modifichi il template private.html per includere una form composta da un singolo bottone che permette di invocare il metodo logout(…).
   Si verifichi che il sistema funziona correttamente provando ad accedere – con browser differenti o utilizzando una finestra in incognito – alla pagina privata senza aver prima fatto accesso tramite le credenziali registrate.

Si realizzi tale sistema utilizzando il framework SpringBoot.

## Riferimenti

- Spring Boot + Thymeleaf HTML Form Handling (Part 1) (https://medium.com/@grokwich/spring-boot-thymeleaf-html-form-handling-762ef0d51327)
- Spring Boot + Thymeleaf HTML Form Handling (Part 2) (https://medium.com/@grokwich/spring-boot-thymeleaf-html-form-handling-part-2-b4c9e83a189c)

- How to integrate Spring Boot with Bootstrap and Thymeleaf (https://medium.com/@omeryazir/how-to-integrate-spring-boot-with-bootstrap-and-thymeleaf-5744fc8475d)
- Serving Web Content with Spring MVC (https://spring.io/guides/gs/serving-web-content/)