

Specifiche del progetto

Fast Press Writer (FPW)

L'applicazione web da sviluppare è un gestore di blog semplificato, che permetta agli utenti registrati come autori di scrivere brevi articoli su notizie o eventi, a utenti registrati come lettori di lasciare commenti sugli articoli, e agli utenti ospiti solamente di leggerli.

Si avranno dunque tre tipologie di utenti:

- Autori che possono
 - inserire le loro informazioni personali
 - Aggiungere, modificare o eliminare i propri articoli sulla piattaforma
 - Inserire commenti su qualsiasi articolo
 - Rimuovere commenti indesiderati sui propri articoli
 - Leggere articoli
 - Vedere le informazioni personali di un utente registrato
- Lettori che possono
 - Inserire le loro informazioni personali
 - Inserire commenti
 - Leggere articoli
 - Vedere le informazioni personali di un utente registrato
- Ospiti che possono
 - Leggere articoli
 - Inserimento dati profilo

Un utente registrato (autore o lettore) ha associati una serie di dati personali (nome, cognome, data di nascita). Inoltre ha una frase di presentazione che appare in cima alla propria pagina di informazioni personali (vedi la funzionalità informazioni personali) ed una immagine del profilo, di cui specifica la URL. Inoltre, ha una password che può modificare.

Una volta registrato, un lettore può decidere di cancellarsi. In questo caso, vengono cancellate tutte le sue informazioni e tutti i suoi commenti agli articoli (vedi sezione Gestione dei commenti).

Gestione degli articoli

Un autore può inserire, cancellare o modificare i propri articoli. Un articolo è costituito da:

- Un titolo
- Una categoria o più categorie, da scegliere fra politica, cronaca, esteri, economia, sport, cultura
- Una data
- Una immagine descrittiva
- Un testo, inferiore ai 300 caratteri

L'autore ha la possibilità di visualizzare in una pagina dedicata i suoi articoli, che gli permette di aggiungerne uno nuovo, modificarne uno esistente o cancellarlo. Nella visualizzazione, gli articoli sono ordinati dal più recente al più vecchio.

Lettura degli articoli

Una volta inserito, un articolo può essere letto da qualsiasi persona che visiti il sito che, di default, mostrerà nella home page tutti gli articoli di tutte le categorie nella home page. Il sito dovrà anche avere una sezione dedicata alla scelta di una determinata categoria: tramite un link l'utente potrà per esempio scegliere di

vedere nella lista degli articoli solo quelli di politica (o qualsiasi altra categoria). La stessa cosa vale per gli autori: l'utente ha la possibilità di ricercare l'autore tramite una barra di ricerca e visualizzare solo gli articoli scritti dall'autore selezionato.




Gestione dei commenti









Visto che non ci sono ancora abbastanza discussioni inutili su internet, i lettori e gli autori hanno la possibilità di commentare il testo di un articolo, tramite una parte dell'interfaccia dedicata a questo immediatamente sotto il testo dell'articolo.

Il commento può contenere solo testo e sarà visualizzato con il nome e cognome di chi l'ha scritto e l'immagine del profilo.

Informazioni personali

Cliccando sul nome dell'autore di un articolo o sul nome dell'autore di un commento, un autore o un lettore può visualizzare le informazioni personali di un altro utente.

<div>Fast Press Writer</div>			
Notizie			<u>LOGIN</u>
<div>Cerca</div> <div>---</div>		<div><div>LOGO</div><div>Fast Press Writer</div></div> <div><div>Nome Utente</div><div>testo</div><div>Password</div><div>****</div><div>Accedi</div></div>	
<div>Categorie</div> <ul style="list-style-type: none">• <u>Politica</u>• <u>Cronaca</u>• <u>Esteri</u>• <u>Economia</u>• <u>Sport</u>• <u>Cultura</u>			
<div>Autori</div> <div> <u>Pinco Pallino</u></div> <div> <u>Sara Bianchi</u></div> <div> <u>Mario Rossi</u></div>			

Notizie		  
Fast Press Writer	Notizie Profilo	Ciao, Davide Logout
Cerca <input type="text" value="..."/>	<h2><u>Auto strada A1 chiusa per neve</u></h2> <div><p>L'auto strada A1 è stata chiusa per neve ... di Pinco Pallino 2/3/18</p><p>cronaca</p></div> <h2><u>Neymar si opera: tre mesi di stop</u></h2> <div><p>L'attaccante ha subito la frattura del... di Sara Bianchi 1/3/18</p><p>Sport, Esteri</p><p>• • •</p></div>	
Categorie <ul style="list-style-type: none">• <u>Politica</u>• <u>Cronaca</u>• <u>Esteri</u>• <u>Economia</u>• <u>Sport</u>• <u>Cultura</u>		
Autori <ul style="list-style-type: none"> <u>Pinco Pallino</u> <u>Sara Bianchi</u> <u>Mario Rossi</u>		

Modifica Profilo		
Fast Press Writer	Notizie	Profilo
	Ciao, Davide <input type="button" value="Logout"/>	
Cerca <input type="text" value="..."/>	<h2>IL MIO PROFILO</h2>	
Categorie <ul style="list-style-type: none">• <u>Politica</u>• <u>Cronaca</u>• <u>Esteri</u>• <u>Economia</u>• <u>Sport</u>• <u>Cultura</u>		Nome <input type="text" value="Davide"/>
		Cognome <input type="text" value="Spano"/>
		Data di nascita <input type="text" value="5/9/1983"/>
		URL Immagine <input type="text" value="http://www...."/>
		Frase di presentazione <input type="text" value="Do or do not"/>
		Password <input type="text" value="****"/>
Autori <ul style="list-style-type: none"> <u>Pinco Pallino</u> <u>Sara Bianchi</u> <u>Mario Rossi</u>	Conferma Password <input type="text" value="****"/>	
	<input type="button" value="Aggiorna"/>	
	Mi sono stancato, <u>Cancella il profilo</u>	

Profilo (di un altro...)

Fast Press Writer

Notizie

Profilo

Cerca

...

Categorie

• Politica

• Cronaca


• Esteri


• Economia


• Sport

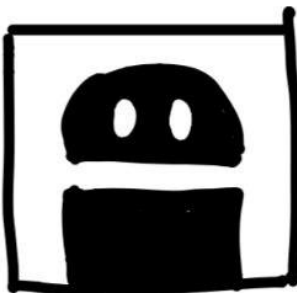
• Cultura

Autori

 Pinco Pallino

 Sara Bianchi

 Mario Rossi



Da vide

Spano

519 / 1983

Do or do not. There is no try

I miei Articoli

Fast Press Writer

Cerca

Categorie

Politica

Cronaca

Esteri

Economia

Sport

Cultura

Autori

😊 Pinco Pallino

👤 Sara Bianchi

👤 Mario Rossi

Notizie

Profilo

I miei Articoli







Ciao, Sara

Logout

I MIEI ARTICOLI

Date	Titolo	Modifica	Cancella
1/3/18	Neymar si opera, tre mesi di stop		
23/2/18	I pronostici della 26ª giornata		
22/2/18	Un articolo		
18/2/18	Un altro articolo		

Nuovo Articolo

Nuovo / Modifica Articolo		  
Fast Press Writer	Notizie	Profilo
	 miei Articoli	
	Ciao, Sara	
	Logout	
Cerca	SCRIVI UN ARTICOLO	
<input type="text"/>		
Categorie	Titolo	<input type="text"/>
• <u>Politica</u>	Data	<input type="text"/>
• <u>Cronaca</u>	Immagine	<input type="text"/>
• <u>Esteri</u>	Sidascalia	<input type="text"/>
• <u>Economia</u>	Testo	<input type="text"/>
• <u>Sport</u>		
• <u>Cultura</u>		
Autori		
 <u>Pinco Pallino</u>		
 <u>Sara Bianchi</u>		
 <u>Mario Rossi</u>		
	Categoria	<input checked="" type="checkbox"/> <u>Politica</u>
		<input type="checkbox"/> <u>Cronaca</u>
		<input type="checkbox"/> <u>Esteri</u>
		<input type="checkbox"/> <u>Economia</u>
		<input type="checkbox"/> <u>Sport</u>
		<input checked="" type="checkbox"/> <u>Cultura</u>
	Salva	

Milestone 1: HTML e CSS

Per questa milestone dovete creare solo il contenuto HTML statico e il layout grafico per il vostro sito web.

Non si possono usare librerie di terze parti per la creazione del layout (es. bootstrap). Sono già convinto che quelli che hanno scritto le librerie siano bravi, io devo valutare voi.

Task 1

Creare nel proprio repository Git Hub un progetto Netbeans del tipo Java -> Web Application per contenere i file da consegnare. Aggiungere una sottocartella di nome "M1" all'interno di "Web Pages". Il risultato deve essere simile al seguente:

Task 2

Creare 4 pagine HTML vuote all'interno della cartella M1:

- login.html
- notizia.html
- articoli.html
- scriviArticolo.html

Creare un file style.css all'interno della stessa cartella. Fare in modo che tutti i file HTML importino il foglio di stile.

Impostare le caratteristiche generali di visualizzazione di una pagina, facendo in modo che le regole che scrivete valgano per tutte le pagine. In particolare:

- Colore di sfondo
- Colore e font per il del testo
- Proprietà dei titoli (almeno da h1 ad h3)
- Proprietà dei link
- Visualizzazione dei campi di input

Lo schema di una generica pagina deve essere simile alla seguente figura. Avete libertà sui colori e tipi di font. Non è quindi necessario che sia in bianco e nero, quello che vedete è solo una bozza, sono anzi caldamente sconsigliati. Utilizzate per quanto possibile le stesse regole su tutte le pagine.

Task 3

Inserire all'interno della pagina di login.html un form per richiedere username e password all'utente, utilizzando i campi di input corretti.

Inserire un link "Notizie" che porta alla pagina notizia.html e collegare il pulsante di login alla pagina articoli.html

Inserite le metainformazioni sulla pagina e validatela.

Task 4

Rendere la pagina del task precedente gradevole dal punto di vista estetico, utilizzando come guida la bozza in figura. In particolare:

- Fare in modo che le label ed i campi di input siano allineati.
- Impostare bordi e colori per i campi di input, in particolare per il focus
- Rendere i pulsanti individuabili e gradevoli, posizionandoli al centro dello spazio riservato al form (oppure in altre posizioni a scelta, che siano però gradevoli e coerenti per tutti i form in tutte le pagine).

Task 5

Inserire all'interno della pagina `notizia.html` il contenuto di una notizia. Non è molto importante quello che scriverete, ma ci deve essere:

- Un titolo
- Una immagine descrittiva, con una didascalia
- Informazioni su autore, data di creazione e categoria dell'articolo (si veda la lista nella pagina di descrizione del progetto 2018)
- Il testo

Task 6

Rendere la pagina del task precedente gradevole dal punto di vista estetico, seguendo la guida della seguente bozza in figura. Per il momento ignorate la parte dei commenti.

Task 7

Inserite una tabella nella pagina `articoli.html` che contenga 3 articoli scritti da uno stesso autore. Per ogni riga, la tabella riporta i seguenti dati:

- Data di pubblicazione
- Titolo
- Un pulsante per la modifica
- Un pulsante per la cancellazione

In fondo alla tabella dovete aggiungere un pulsante per aggiungere un nuovo articolo, collegato alla pagina `scriviArticolo.html`.

Inserite le metainformazioni sulla pagina e validatela.

Task 8

Rendere la pagina del task precedente gradevole dal punto di vista estetico, utilizzando come guida la bozza in figura. In particolare:

- Fate in modo che la tabella si adatti alla larghezza della pagina
- Differenziate le celle di intestazione da quelle dei dati
- Alternate i colori di sfondo righe pari e delle righe dispari, in modo da aiutare la lettura
- Gestite i bordi delle celle

Task 9

Creare un form per la scrittura di un articolo nella pagina `scriviArticolo.html`. Il form deve permettere di inserire:

- Un titolo
- La data di pubblicazione
- La URL dell'immagine descrittiva
- La didascalia per l'immagine
- Il testo dell'articolo
- La selezione della categoria.

Utilizzate i tipi di input corretti.

Task 10

Rendete gradevole la pagina del task precedente dal punto di vista estetico, in particolare gestendo i campi del form e rendendolo omogeneo con quelli delle altre pagine.

Task bonus

Create un layout responsive, da utilizzare per tutte le pagine. In particolare considerate tre configurazioni:

- Per larghezze maggiori o uguali a 1024px utilizzare un layout a due colonne
- Per larghezze minori o uguali a 480px utilizzare un layout ad una sola colonna
- Per quelli intermedi potete scegliere voi se usare una o due colonne.

Nella visualizzazione ad una colonna, della barra di sinistra deve essere visibile:

- La barra di ricerca
- Categorie e Autori che devono diventare due link

Non devono essere visibili invece le singole categorie (politica, cronaca, ecc) e il nome degli autori. Attenzione nel layout a due colonne, Categorie e Autori devono essere due titoli.

Task 11

Eseguite il commit finale su Git Hub per la consegna, utilizzando come messaggio “consegna M1”

Milestone 2: Programmazione Server Side

Per questa milestone dovete creare una parte della logica server side del per il vostro sito web.

Regole:

Sono sempre valide le regole delle milestones precedenti. Validate il codice HTML ed usate un CSS per tutte le pagine.

Potete modificare l'HTML ed i CSS delle pagine della milestone precedente. Prima di farlo però chiedetevi se sia strettamente necessario e, soprattutto, attenzione a non introdurre errori.

Task 1

Create un package dedicato a contenere il modello della vostra applicazione. All'interno di questo package ci deve essere una classe per ogni oggetto del dominio applicativo manipolato dalla vostra applicazione. In particolare:

- Utenti
- Articoli
- Commenti

Inserite all'interno di queste classi tutte le variabili necessarie per descriverli. Fatto questo, create per ognuna una Factory che restituisca istanze della classe popolata con dati fittizi, ottenuti impostandone le proprietà con valori arbitrati e restituendole in base ad un determinato criterio.

Per esempio (vuol dire che non siete costretti ad implementare l'elenco di sotto ma qualcosa di simile), consideriamo la classe News (il nome non è importante, chiamatela come volete) che rappresenta un articolo. La corrispondente factory NewsFactory potrebbe avere i seguenti metodi:

- News getNewsById(int id) che restituisce l'oggetto avente l'identificatore passato per parametro
- List<News> getNewsByAuthor(User usr) che restituisce tutti gli oggetti News scritti da un determinato autore
- List<News> getNewsByCategory(Category c) che restituisce tutti gli oggetti News per una determinata categoria

E tutti gli altri metodi necessari per fare la ricerca di dati nell'applicazione.

Task 2

Trasformate le pagine HTML delle milestones precedenti in JSP, effettuando le seguenti operazioni:

Individuate i pezzi ripetuti di HTML ed isolateli in altre JSP, importandoli all'interno delle altre con le include (p.es. Header, footer, barre di navigazione ecc.)

Rendete dinamiche le parti HTML da generare in base ai dati dell'applicazione, come per esempio l'elenco delle notizie. Per fare questo, assumete che nella request siano stati impostati tutti gli attributi necessari (p.e. la lista delle notizie) dalla servlet che richiama la JSP.

Impostate la url notizie.html come welcome file della vostra applicazione web

Nota: Non preoccupatevi se eseguendo il progetto la url notizie.html vi restituisce 404. Sistemere la cosa al Task 5.

Task 3

Create una servlet Login e mappatela sulla URL login.html. La servlet si deve comportare nel modo seguente:

Nel caso l'utente non sia autenticato, deve mostrare il form di login e verificare username e password nel caso siano inviate tramite il form

Nel caso l'utente sia già stato autenticato (durante la gestione della richiesta corrente o ad una precedente) deve riportare alla pagina con l'elenco delle notizie. Usate una redirect.

Task 4

Create una servlet NewsDetail e mappatela sulla URL notizia.html. La servlet si deve comportare nel modo seguente:

In base ad un parametro nid ricevuto tramite richiesta, deve mostrare nel dettaglio una notizia, come da immagine seguente. Deve essere per esempio possibile scrivere una URL notizia.html?nid=1 per vedere la notizia con id 1 e notizia.html?nid=5 per vedere la notizia 5.

Nel caso l'utente non sia autenticato, deve mostrare nella barra di navigazione principale il link alla pagina di login.

Nel caso l'utente sia autenticato, deve mostrare una zona "personalizzata" per l'utente, con un messaggio di benvenuto personalizzato (p.e. "Ciao, Davide") e il pulsante per fare logout (simile all'immagine di sotto).

Nel caso l'utente sia un autore, la barra di navigazione principale deve mostrare il link alla pagina articoli.html.

N.B. Non è necessario gestire l'inserimento di un commento, ma solo la loro visualizzazione (createne qualcuno nelle factory).

Task 5

Create una servlet Notizie e mappatela sulla URL notizie.html. La servlet si deve comportare nel modo seguente:

Deve mostrare la lista delle notizie presenti nel portale, dalla più recente alla più vecchia. Per ogni notizia mostra l'immagine descrittiva, i primi 100 caratteri del testo, la categoria e il nome dell'autore (si veda l'immagine sotto). Cliccando sul titolo o sull'immagine, si viene riportati al dettaglio del task 4.

Cliccando sui link delle categorie a sinistra, deve mostrare solo gli articoli della categoria selezionata. Il nome della categoria deve essere mostrato come un h1 prima degli articoli. Nel caso l'utente clicchi sul link notizie nella barra di navigazione principale, saranno mostrate nuovamente tutte le notizie.

Opzionale: aggiungete una categoria fittizia "Tutte" nella barra laterale che abbia l'effetto di eliminare il filtro delle categorie.

Nel caso l'utente non sia autenticato, deve mostrare nella barra di navigazione principale il link alla pagina di login.

Nel caso l'utente sia autenticato, deve mostrare una zona "personalizzata" per l'utente, con un messaggio di benvenuto personalizzato (p.e. "Ciao, Davide") e il pulsante per fare logout (simile all'immagine del task 4).

Nel caso l'utente sia un autore, la barra di navigazione principale deve mostrare il link alla pagina articoli.html.

Suggerimenti: gli ultimi tre punti sono in comune con il task precedente. Una implementazione sensata vi permette di riutilizzare quello che avete fatto prima.

Task 6

Create una servlet Articles e mappatela sulla URL articoli.html. La servlet si deve comportare nel modo seguente:

Nel caso l'utente non sia un autore, deve mostrare un messaggio di accesso negato

Nel caso l'utente sia un autore, deve mostrare l'elenco dei suoi articoli.

Gli articoli devono essere visualizzati in una tabella. Premendo sul pulsante modifica, il sito porta alla pagina del task 6, caricando nella form i dati dell'articolo selezionato.

Suggerimenti: Fate attenzione all'utilizzo della sessione. Deve essere possibile aprire più notizie su più tab.

Task 7

Create una servlet NewArticle e mappatela sulla URL scriviArticolo.html. La servlet si deve comportare nel modo seguente:

Nel caso l'utente non sia un autore, deve mostrare un messaggio di accesso negato

Nel caso l'utente sia un autore, deve mostrare nei campi di input i valori dei campi della notizia selezionata come da task 6.

Premendo il pulsante Salva, la pagina viene ricaricata, mostrando nuovamente i campi compilati dall'utente e un messaggio in cima con l'id dell'articolo.

Suggerimenti: Fate attenzione all'utilizzo della sessione. Deve essere possibile aprire più notizie su più tab.

N.B. In realtà nel caso si faccia il reload della pagina, i dati torneranno ad essere quelli che erano prima dell'inserimento. Ciò che scrivete nel form sarà disponibile solo per la risposta immediatamente dopo il submit del form. È un problema che risolveremo nella prossima milestone.

Task 8

Create una servlet Profilo e mappatela sulla URL profilo.html. La servlet si deve comportare nel modo seguente:

Nel caso l'utente non sia autenticato, deve mostrare un messaggio di accesso negato

Nel caso l'utente sia autenticato, deve mostrare il form di inserimento dei dati del profilo.

Nel caso siano inviati i dati relativi al profilo, deve mostrare una conferma dell'avvenuto inserimento ed i dati inseriti.

N.B. In realtà nel caso si faccia il reload della pagina, i dati torneranno ad essere quelli che erano prima dell'inserimento. Ciò che scrivete nel form sarà disponibile solo per la risposta immediatamente dopo il submit del form. È un problema che risolveremo nella prossima milestone.

Task 9

Eseguite il commit finale su GitHub per la consegna, utilizzando come messaggio "consegna M2". Utilizzate lo stesso repository della prima milestone.

Milestone 3: Persistenza dei dati

Per questa milestone dovete creare una parte della logica server side del per il vostro sito web.

Regole:

Sono sempre valide le regole delle milestones precedenti. Validate il codice HTML ed usate un CSS per tutte le pagine.

Potete modificare l'HTML ed i CSS delle pagine della milestone precedente. Prima di farlo però chiedetevi se sia strettamente necessario e, soprattutto, attenzione a non introdurre errori.

Potete modificare JSP e Servlet della milestone precedente. Non rompete però il pattern MVC

Task 1

Individuate entità e relazioni nel dominio della vostra applicazione, insieme con i loro attributi. Una volta che li avete individuati, descriveteli in modo schematico in un file testuale da inserire in una cartella db-definition all'interno di WEB-INF. Per schematico intendo un elenco di entità con relativi attributi e di relazioni con relazione di partenza, di arrivo, cardinalità ed eventuali attributi.

Task 2

Traducete lo schema del task precedente in tabelle.

Create le query per la loro creazione, salvandole in file crea-db.sql da inserire nella cartella WEB-INF/db-definition.

Create delle query di inserimento per i dati iniziali dell'applicazione (quelli che avevate nelle factory: utenti, gruppi, post ecc.)

Task 3

Connettetevi al vostro database messo disposizione dal docente con le istruzioni al link <http://moodle.unica.it/mod/page/view.php?id=20972> tramite Netbeans. Autenticatevi e assicuratevi che la connessione funzioni.

Eseguite le query contenute in crea-db.sql connettendovi al db creato al task 2.

Create nel modello una classe che mantenga le credenziali di autenticazione al database server, contenente un metodo che vi restituisca una connessione pronta all'uso nelle factory.

Modificate i metodi della factory creata per la milestone precedente in modo che legga i dati eseguendo delle query sullo stesso db.

Task 4

Aggiungere alle factory i metodi necessari per salvare in modo persistente gli articoli scritti da un autore. Il programma si dovrà comportare nello stesso modo descritto nel Task 7 della milestone 2, ma le modifiche all'articolo dovranno essere visibile anche fra due visite consecutive della pagina delle bacheca.

Inserire anche la gestione della creazione di un nuovo articolo sul database alla pressione del pulsante "nuovo articolo" del task 6 della milestone precedente. L'effetto deve essere quello di inserire un nuovo articolo nel database e immediatamente dopo aprire la pagina della modifica dell'articolo appena creato.

Per chi non avesse svolto i della milestone precedente, suggeriamo di inserire un semplice pulsante che salvi dei dati generati a programma (per esempio "Articolo delle [data e ora corrente]" come titolo), senza implementare la sequenza descritta nel task della milestone 3, e di verificare prima l'inserimento nel db.

Task 5

Gestire la cancellazione di un utente e degli eventuali articoli che ha scritto con una transazione. Il pulsante per la cancellazione deve essere disponibile nella pagina di modifica dei dati del profilo e deve essere accessibile al solo proprietario del profilo stesso. La cancellazione deve procedere nel modo seguente:

- Devono essere cancellati prima tutti i commenti che ha eventualmente scritto sugli articoli
- Devono essere cancellati tutti gli articoli di cui l'utente è autore
- Dopo di che devono essere cancellati i dati del profilo.

La cancellazione deve andare a buon fine solo nel caso i passi precedenti siano completati in modo corretto. Altrimenti la situazione nel db deve rimanere invariata.

N.B. Nel caso abbiate incluso il trigger ON DELETE nelle chiavi esterne della tabella dei commenti e degli articoli, sarebbe sicuramente possibile fare la cancellazione eliminando direttamente la riga dell'utente. Per motivi didattici (voglio vedere se riuscite a scrivere una transazione) vi chiedo di farlo necessariamente con una transazione in qualsiasi caso.

Nel caso di cancellazione corretta, deve essere mostrata semplicemente la pagina di login.

Task 6

Eseguite il commit finale su GitHub per la consegna, utilizzando come messaggio “consegna M3”

Milestone 4: Programmazione Client-Side

Task preliminare

Librerie client side

Andate alla pagina <https://jquery.com/download/> e scaricate la libreria jQuery.

Rinominate la libreria in jquery.js

Inseritela in una cartella js all'interno della cartella "Web Pages"

Importatela con il tag script nello head delle vostre pagine

Librerie server side

Andate alla pagina <http://json-taglib.sourceforge.net/> e scaricate il jar della libreria json-taglib

Da Netbeans, selezionate la visualizzazione del progetto per File (tab Files, quella che mostra le cartelle "vere" del progetto per intenderci) e create una cartella lib allo stesso livello della cartella src

Copiate il jar di json-taglib all'interno della cartella lib

Tornate nella visualizzazione del progetto (tab Projects), cliccate con il tasto destro sulla cartella Libraries e selezionate "Add JAR/Folder"

Nel file browser che appare selezionate il jar appena copiato nella cartella lib del progetto (è importante che si trovi nella cartella creata al punto 2 per la correzione)

Confermate la selezione.

Task 1

Nella barra laterale, dove è presente la lista degli autori e delle categorie, aggiungete (se non lo avete già) un campo di testo per effettuare una ricerca. Quanto l'utente scrive nel textfield, ad ogni pressione di un tasto la pagina deve inviare una richiesta ajax all'indirizzo filter.json, passando un parametro con chiave q e come valore la stringa inserita dall'utente.

Inserite il codice javascript in un file separato, da mettere all'interno della cartella web/js del progetto. Inserite in questa cartella anche la libreria jQuery nel caso vi serva.

Task 2

Creare una servlet di nome Filter e mapparla sulla URL filter.json. Quando riceve una richiesta da parte di un utente, deve produrre un elenco di autori e categorie presenti nel sistema che abbiano la stringa passata tramite il parametro q all'interno o del campo nome o cognome per gli autori e nome per la categoria. In particolare deve restituire questo elenco in formato json, tramite un array di oggetti aventi come proprietà la tipologia (autore o categoria), il nome per le categorie, il nome, cognome e url della foto del profilo per gli autori. Nel caso la barra di testo sia vuota, deve mostrare tutte le categorie e tutti gli autori (supponiamo che siano un numero non troppo alto, che non renda necessaria la paginazione della lista).

N.B. Fate attenzione al pattern MVC e importate la libreria di tag della json-taglib.

Suggerimento:

Ho notato di aver cancellato per sbaglio due slides della parte del database che sono utili per implementare questa funzionalità. Vi riporto qui il contenuto:

A volte è necessario fare delle ricerche all'interno di alcuni campi, senza conoscerne l'esatto valore

Nelle query SQL, l'operatore LIKE nella WHERE permette di "fissare" solo una parte del valore

Il pattern viene specificato tramite due wildcards (caratteri che si sostituiscono ad altri)

Il carattere _ sostituisce un solo carattere qualsiasi

Il carattere % sostituisce una sequenza di caratteri qualsiasi

Si possono inserire prima o dopo una stringa fissa (o anche prima e dopo)

Esempi

LIKE "Mari_" seleziona sia Maria che Mario, ma non Mariottide

LIKE "Mari%" seleziona Maria, Mario e Mariottide

LIKE "%Mari%" seleziona Caro Mario e Ciao Mariottide

Ricerchiamo tutti gli studenti il cui nome contenga una i

```
SELECT * FROM studenti WHERE nome LIKE '%i%';
```

Task 3

Scrivete il codice che, a partire dal json ricevuto da filter.json, popoli dinamicamente la lista delle categorie e degli autori visualizzata a lato della bacheca, senza ricaricare la pagina. In particolare:

I collegamenti relativi alle categorie devono mostrare solo le news appartenenti a quella categoria

I collegamenti relativi agli autori devono mostrare solo le news scritte da un determinato autore

Suggerimento: se non li avete già aggiungete un parametro alla servlet News per caricare solo quelle di un determinato autore.

Task 4

Eseguite il commit finale su Git Hub per la consegna, utilizzando come messaggio "consegna M4"