Capitolo 1

Deel: Applicazione web per l'hosting, la condivisione e il backup dei file

1.1 Descrizione generale

1.1.1 funzionalit:

Deel implementa le funzionalit pi comuni ricercate in un servizio di hosting di file quali l'upload di file di varie dimensioni, il download, lo sharing, l'organizzazione gerarchica in cartelle e il versionamento automatico dei file caricati sul sistema, gestione del proprio account.

1.1.2 Struttura

L'applicazione stata suddivisa in moduli interdipendenti tra loro, ognuno dei quali indirizza presice funzionalit chiave. Ogni modulo espone agli altri con cui in una relazione di dipendenza solo l'interfaccia, mantenendo privata l'implementazioe, cos da aumentare la possibilit di riutilizzo e manutenzione del codice. I principali moduli sono mostrati nella figura in basso.

Domain : Le classi che fanno parte del dominio rappresentano la modellazione del problema svincolata dalle logiche di accesso ai dati, persistenza, navigazione ecc. Le classi sono strutturate come semplici POJO (plain old java object) cos da non dipendere da nessun tipo di framework particolare. Le clasi si trovano nel package java org.deel.domain.

Data layer :

Si occupa della persistenza dei dati, e di fornire un mapping consistente tra i dati permanenti e i dati manipolati in memoria dall'applicazione. Per salvare i metadati relativi ai file ed i dati relativi agli utenti stato utilizzato un DBMS, in particolare MySQL. I file veri e propri vengono salvati utilizzando le funzionalit di accesso al filesystem dell'sistema operativo. Per consentire il mapping tra oggetti del database e oggetti utilizzati dal sistema sono state utilizzate le funzionalit del framework Hibernate

Service layer:

Il service layer si occupa di implementare i principali casi d'uso dell' applicazione, manipolando sia oggetti del domain sia interagendo con il data layer. Il service layer, come il layer web, utilizza la dependency inkection del framework Spring. In questo modo l'istanziazione degli oggetti di cui questo layer necessita (ad es. gli oggetti con le funzionalit DAO presenti nel datalayer) avviene in maniera automatica e il codice non dipende da essa.

Web Layer : Il web layer si occupa della logica di navigazione e della gestione della sicurezza dell'applicazione. Per l'implementazione della sicurezza (acesso e autenticazione) stato utilizzato il modulo Spring Security, che tramite filtri serlvet permette una acesso selettivo e autenticato al sistema. Per quanto riguarda la navigazione sono state utilizzate le funzionalit offerte dal modulo Spring MVC, funzionante su un servlet container (ad es. tomcat). Quando una richiesta arzriva all'applicazione essa viene mappata all'opportuno *Controller*. Il mapping viene definito tramite annotazioni sul codice java. Il Controller, nel caso di una richiesta HTML, si occupa di validare l'input (anche questo viene definito grazie ad annotazioni), di chiamare gli opportuni metodi sul layer server, di esporre i dati del modello e di indi-

care il nome di una vista da visualizzare all'utente. Se invece la richiesta ti tipo JSON, dopo la validazione dell'input e le interazione con il layer service il controller ritorna le informazioni richieste al client (sotto forma di oggetto JSON). In questo modo abbiamo potuto delegare parte della logica riguardate la User Interface sul lato client, fornendo da una parte una migliore e pi reattiva esperienza utente, dall'altra diminuendo in qualche modo il carico di lavoro sul lato server.

User Interface:

E' il layer pi vicino all'utente e l'unico con cui interagisce direttamente. Nel layer Web, il controller indica il nome di una vista e i dati da esporre, a questo punto il nome della vista viene mappato ad un file jsp, in cui vengono utilizzati i tag JSTL per renderizzare i contenuti nel modello in una pagina finale html. In questo modo facile aggiungere nuove tecnologie di rendering senza modificare il layer web. Le pagine jsp non sono accessibili dall'esterno. Per migliorare l'esperienza utente parte di questa logica stata portata sul lato client tramite l'utilizzo di chiamate asincrone AJAX. L'interfaccia utente in questo caso esegue le sue richiesto in modo asincrono e tramite un meccanismo di callback si aggiorna dinamicamente.

si occupa della logica di navigazione e della gestione della sicurezza. Per la realizzazione di questo layer sono state utilizzate tecnologie di Spring tra le quali Spring MVC e Spring Security. User Interface Layer : ? la parte esposta all? utente e si occupa dell? interazione con esso. Per lo sviluppo sono state utilizzate le tecnologie JSTL per quanto riguardo il lato server e jQuery lato client. Paragrafo 3. DBMS Per la realizzazione del database ? stato utilizzato MySQL. Lo schema E-R dell? applicazione ? definito nella figura in basso. Per modellare una struttura ad albero rappresentante la struttura organizzativa dei file dell? utente sono state utilizzate le tabelle Folder e FilePath, dove un filepath appartiene ad ed una sola directory e Folder ha una relazione ricorsiva. Ogni FilePath sar collegato con un solo File e ci permette una migliore gestione della condivisione. Paragrafo 4. ORM Quasi tutte le relazioni sono state mappate in modo bidirezionale per

motivi di efficienza. E? stato inoltre utilizzato il caricamento Lazy in modo di ridurre i tempi di caricamento ed evitare un catastrofico caricamento ricorsivo di tutto il Database in memoria! Hibernate ? stato configurato per essere integrato con Spring. Infatti l? utilizzo del transactionManager(di Spring) e della sessionFactory verr fatto attraverso dei Bean e con il frequente utilizzo della Dependency Injection. Cascade -¿ Per un controllo maggiore da parte della Business Logic si ? evitate di utilizzare inserimenti a cascate. Questo costrutto ? stato utilizzato solo in particolari casi che prevedevano una minima propagazione. Paragrafo 5. Transaction Strategies and Model Come modello di transazione ? stato utilizzato il modello dichiarativo che consente di non mischiare codice relativo alla logica del programma con codice relativo all? accesso ai dati nonch meno error prone. In questo modello il contenitore(nel nostro caso Spring attraverso Hibernate) si occupa di iniziare, committare e rollbackare le transazioni. Come strategie invece abbiamo definito come Unit Of Work nella nostra applicazione un intero caso d? uso, implementato come metodo nel modulo Service. Utilizzando quindi l? annotazione @Transactional su questi metodi abbiamo ottenuto una non intrusiva e consistente strategia di transazione. Paragrafo 6. Web ed User Interface Per migliorare l? interfaccia utente si ? scelto di mantenere le funzionalit usate pi? frequentemente sulla stessa pagina facendo uso di chiamate asincrone presenti nelle librerie jQuery. Per questo la maggior parte della logica di navigazione riguarda quindi casi d? uso meno frequenti come la registrazione, il login o l? interfaccia amministrazione. Per quanto riguarda autenticazione ed access control ? stato usato un modulo Spring denominato Spring Security che tramite filtri servlet gestisce l? accesso alle pagine ed il login. Quest? ultimo modulo ? configurabile quasi totalmente tramite una pagina xml nella quale vengono fatti dei settaggi. Paragrafo 7. Note sui test Sono state adottate molte tecniche di sviluppo in base ai moduli implementati tra le quali Test-driven Development utilizzando i popolari framework di testing JUnit e Mockito. Paragrafo 8. Esempio di un caso d? uso In questo paragrafo verr descritto un caso d? uso d? esempio. Per l? upload

di un file l? utente quindi trasciner il file nel div dopodich verr mandata una richiesta al controller che tramite il modulo service gestir l? inserimento o il revisionamento del file. Andata a buon fine la transazione la pagina tramite una callback aggiorner la vista dinamicamente senza ricaricare la pagina. Paragrafo 9. Conclusioni Uno dei vantaggi di aver utilizzato Spring ed in particolar modo la dependency injection ? stato quello di aver reso per quanto pi? possibile indipendenti i moduli del sistema che non dovendo conoscere nessun dettaglio implementativo possono essere sia migliorati che testati singolarmente.

La mia introduzione