

Documento di Design

Gruppo 6 - Pigozzi Stefano, Lin Jie

14 December 2007

Contents

1	Class Diagrams	3
1.1	Overview	3
1.2	Dettagli sul Modello	3
1.3	View & Controller	4
2	Sequence Diagrams	5
2.1	Lancio Applicazione	5
2.2	Selezione Notizia	6
2.3	Nuovo Feed	6
2.4	Impostazione di un filtro	7

1 Class Diagrams

1.1 Overview

In questa sezione presentiamo un diagramma che mostra gli aspetti generali dell'organizzazione generale delle classi all'interno del progetto. L'architettura generale si ispira al paradigma di programmazione Model-View-Controller (MVC), che permette grande flessibilità ed aiuta a spezzare il problema in tre grossi oggetti logici che comunicano tra loro.

Come si nota nel diagramma in realtà è più corretto parlare di Model-View (MV); infatti avendo scelto di implementare l'applicazione in JAVA, il Controller sarà contenuto nella View, spezzato in molti frammenti e localizzato nei punti del codice dove si risponderà agli eventi generati dall'Utente attraverso la GUI. Si noti che non verrà fatta una implementazione da zero, ma ci si aggancerà al comodo framework offerto dal linguaggio JAVA dove si trovano già implementate la classe `java.util.Observable` e l'interfaccia `java.util.Observer`.

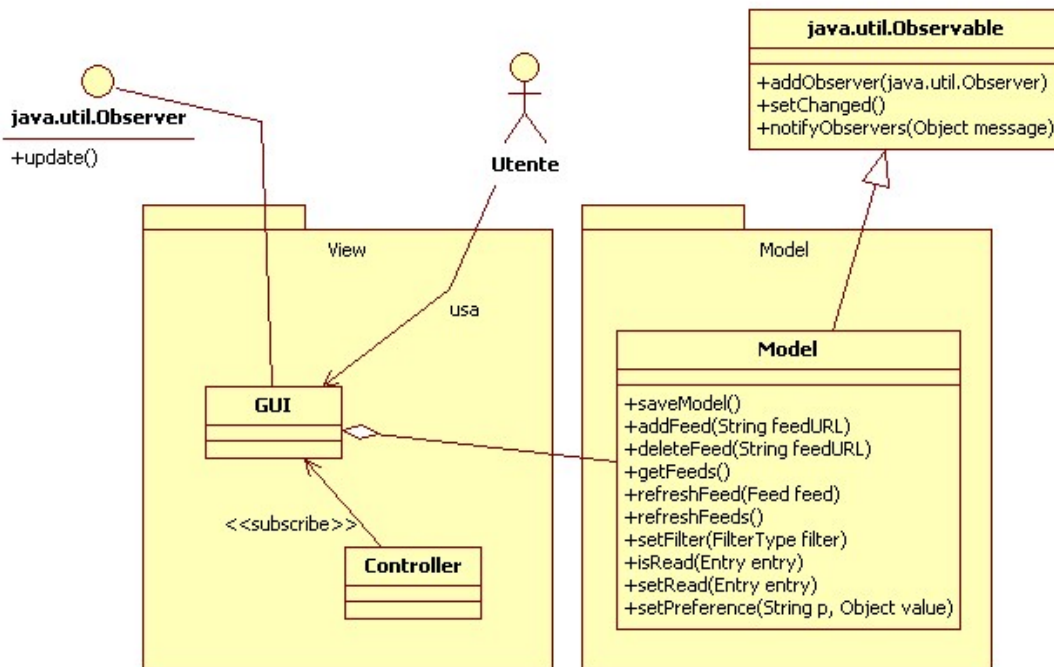


Figure 1: Class Diagram: Classes Overview

1.2 Dettagli sul Modello

Il modello è concettualmente piuttosto semplice. Abbiamo cercato di ridurre al minimo il numero di classi atte a wrappare semplici informazioni dove questo non generasse un'eccessiva complicazione. Per esempio, ci è sembrato opportuno rappresentare `Location` e `Business` con attributi già presenti nel linguaggio piuttosto che generare due classi aggiuntive.

Si noti che la struttura dati che rappresenta l'albero dei `Feed` e `Entries` non è modificabile dall'esterno del `Model`, se non con future estensioni. In questo modo si previene un uso scorretto del `Model` da parte della `View` nel caso un altro sviluppatore dovesse riscrivere l'interfaccia utente. Ogni modifica al `Model` deve essere fatta chiamando un metodo del `Model`.

L'unica tra le classi che possa non risultare autoesplicativa per il lettore è `FilterType`. Semplicemente questa classe permette di wrappare tutte le informazioni sul tipo di filtro da applicare

e sulla stringa da cercare. Questo permette di avere un'interfaccia comune e ben definita da utilizzare sia nel modello. In questo caso il costruttore sarà pubblico in modo da poter costruire i filtri all'interno del Controller in base agli input forniti dall'Utente.

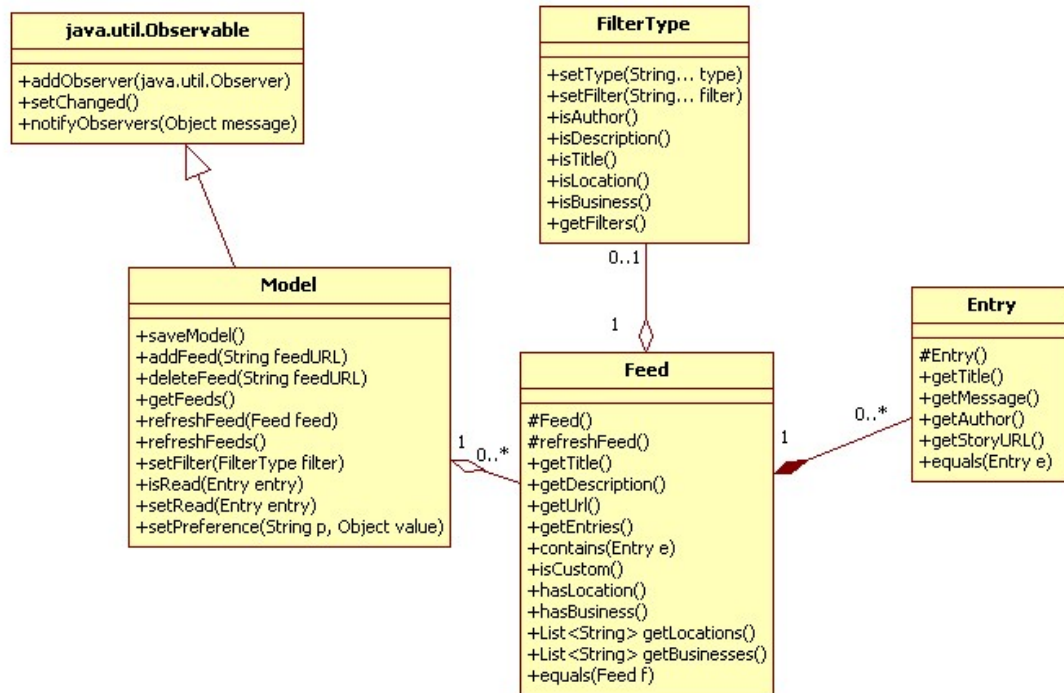


Figure 2: Class Diagram: Dettagli sul Modello

Nella Figura è possibile vedere una versione preliminare dei metodi esposti dalle varie classi. Questo permette di dare un'idea di base delle varie funzionalità di ciascuna delle classi, e nonostante non cambierà la filosofia di base circa i compiti delle varie classi, ci riserviamo la possibilità di cambiare alcune interfacce in fase di implementazione, se questo permetterà di ottenere migliori risultati.

1.3 View & Controller

Come già scritto prima nell'Overview il Controller sarà condensato attraverso classi anonime all'interno della View. E' in queste classi interne che ci si potrà servire dei metodi pubblici esposti da `model.Model` che modificano il Model stesso. Sarà concettualmente corretto usare nella View i metodi di interrogazione (getters), per ottenere le informazioni da organizzare nella GUI.

In questa fase è prematuro elencare le classi all'interno della GUI (esse dipendono quasi esclusivamente dal tipo di presentazione grafica che si vorrà scegliere per le informazioni). Indicativamente si pensa di creare una classe per ogni finestra a video, più eventuali classi accessorie atte a costruire dei Widget Swing complessi originati dalla composizione dei Widget Swing di default.

2 Sequence Diagrams

In questa sezione presentiamo alcune interazioni d'esempio in modo da rendere chiaro in che modo le varie componenti del sistema comunichino tra di loro. Tutte le interazioni sono generalmente piuttosto simili, poiché è una proprietà del MVC disciplinare i sensi in cui viaggiano le informazioni. Si può notare infatti che la GUI richiede cambiamenti al Controller e la richiesta viene propagata sul Model, questo eventualmente risponde notificando gli osservatori iscritti (nel nostro caso uno solo).

NB: Nonostante venga indicato il Controller come oggetto in più di uno dei diagrammi che seguono, il Controller non esisterà fisicamente ma sarà all'interno della View. Abbiamo dato una rappresentazione con Controller staccato per permettere di fissare meglio le idee.

2.1 Lancio Applicazione

Pre-condition	nessuna! il sistema non esiste!
Normal Flow	vengono create le classi principali e vengono collegate per utilizzare il framework MVC, vengono poi invocati i metodi e il sistema viene portato a regime
Post-condition	il sistema è a regime pronto a rispondere all'interazione con l'Utente

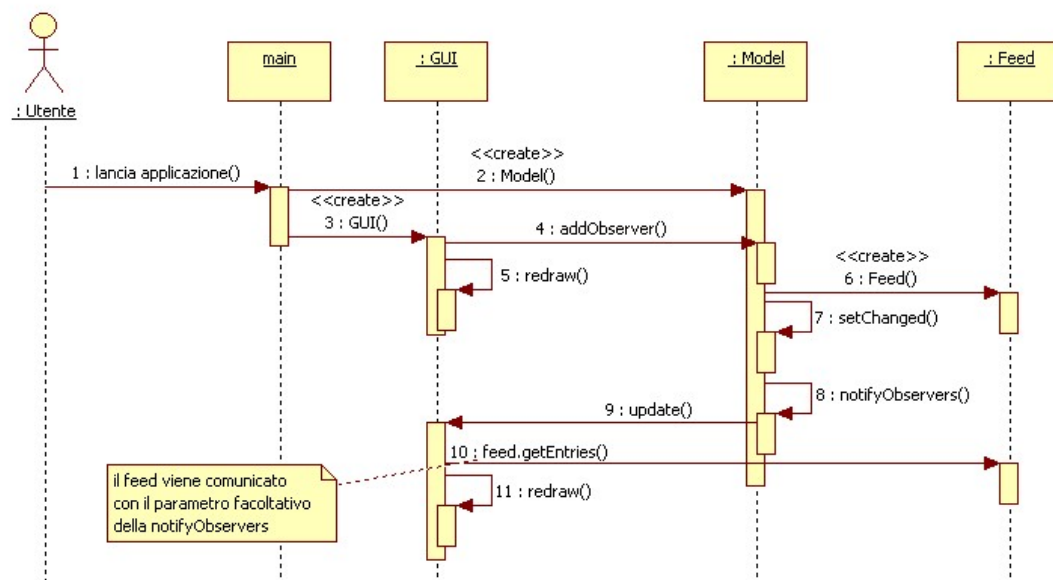


Figure 3: Sequence Diagram: Lancio dell'Applicazione

2.2 Selezione Notizia

- Pre-condition** l'insieme delle notizie visualizzate deve essere non vuoto.
- Normal Flow** viene aperta una finestra del browser che punta all'indirizzo della notizia completa, e se la notizia non è stata letta viene marchiata come tale.
- Post-condition** la notizia scelta alla fine sarà segnata come letta (che lo fosse o no in precedenza)

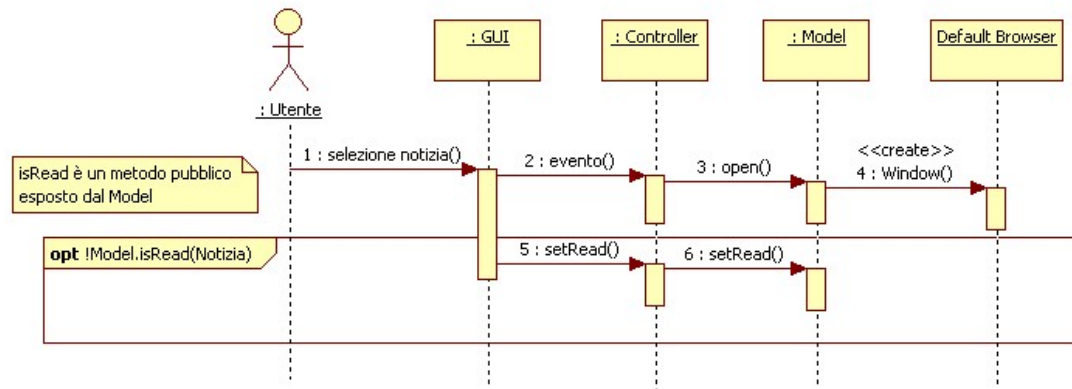


Figure 4: Sequence Diagram: Selezione di una notizia

2.3 Nuovo Feed

- Pre-condition** il Feed non è già presente all'interno del Model (in realtà, si escluderà questo caso con la notifica alla View dell'errore o con il lancio di un'eccezione).
- Normal Flow** se il Feed non è presente nel Model lo si aggiunge e si notifica alla GUI che l'operazione è andata a buon fine. Ora la GUI può prelevare le informazioni aggiornate.
- Post-condition** il Feed specificato dall'utente sarà nella struttura dati del Model.

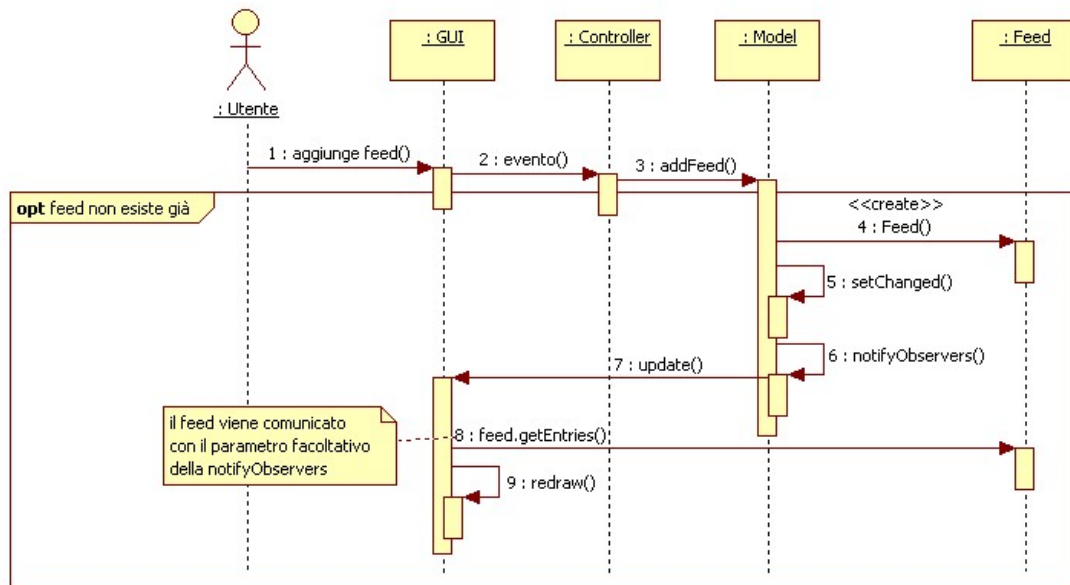


Figure 5: Sequence Diagram: Selezione di una notizia

2.4 Impostazione di un filtro

- Pre-condition** nessuna (se c'è già un filtro applicato viene cambiato in quello nuovo).
- Normal Flow** se il Feed non è presente nel Model lo si aggiunge e si notifica alla GUI che l'operazione è andata a buon fine. Ora la GUI può prelevare le informazioni aggiornate.
- Post-condition** il Feed specificato dall'utente sarà nella struttura dati del Model.

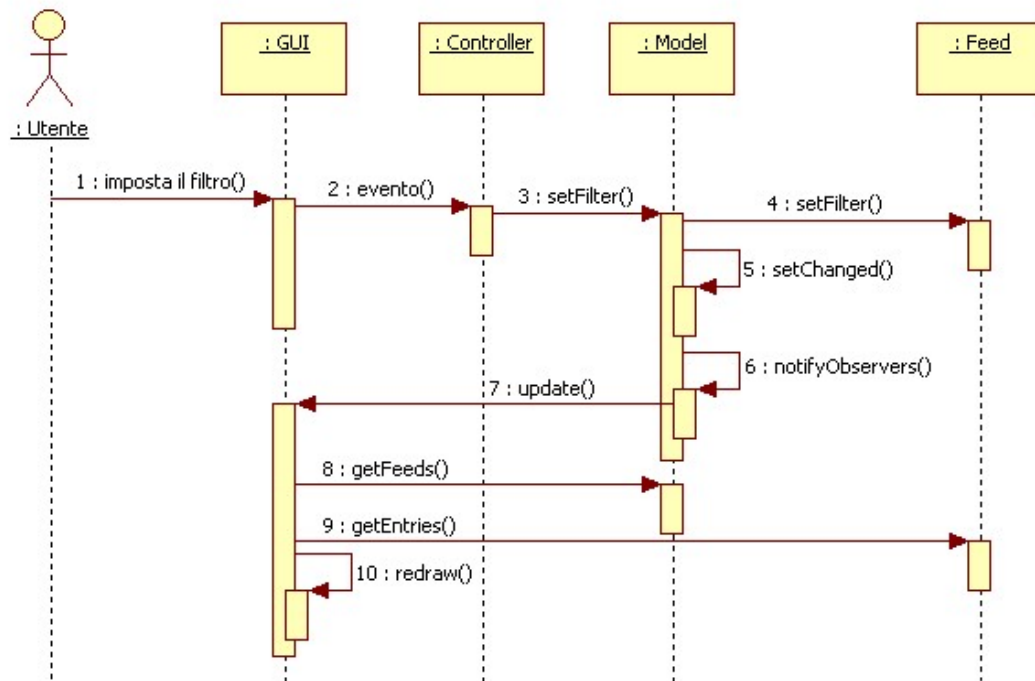


Figure 6: Sequence Diagram: Impostazione di un filtro