

Dajly :: Testing Report

Gruppo 6 - Lin Jie & Pigozzi Stefano

29 Febbraio 2008

Contents

1	Testing Comportamentale	3
1.1	L'interfaccia "amichevole"	3
1.2	Le impostazioni di default	3
1.3	Operazioni standard	4
1.4	Personalizzazione del formato	4
1.5	Gestione casi eccezionali	4
1.6	I vari "dialetti" del linguaggio RSS	4
1.7	La serializzazione dei dati	4
1.8	Selezione e lettura di una notizia	5
1.9	Filtraggio e ricerca di una notizia	6
1.10	Modifica preferenze	6
2	Unit Testing	7
2.1	TestCase - Entry	7
2.2	TestCase - Feed	7
2.3	TestCase - Model	8
2.4	Test Case - FilterType	8

1 Testing Comportamentale

In questa sezione verificheremo che il software prodotto abbia tutte le caratteristiche che ci eravamo prefissati nella fase di Analisi dei Requisiti. A tale scopo riprenderemo alcune tra le specifiche elencate nei paragrafi §1.3 e §1.5 del documento di Analisi dei Requisiti, e per ciascuna di esse mostreremo le funzionalità corrispondenti all'interno dell'applicazione da noi prodotta.

1.1 L'interfaccia "amichevole"

Come si vede dalla figura sottostante, il programma presenta un'interfaccia semplice con poche funzioni. I comandi sono intuitivi e ben visibili. La struttura della finestra è semplice e permette all'utente di visualizzare subito i feed presenti, i messaggi del feed selezionato e il contenuto del messaggio selezionato.

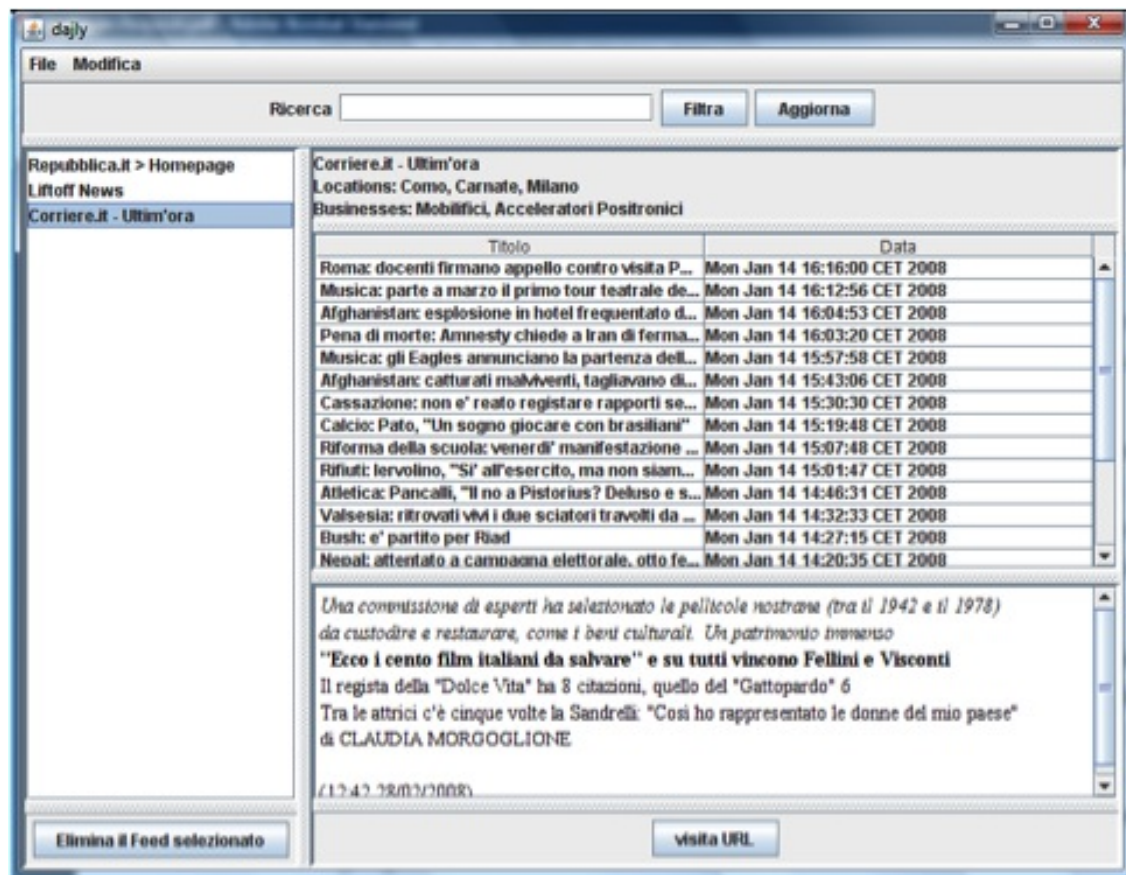


Figure 1: La schermata principale del programma

1.2 Le impostazioni di default

Appena avviato il programma l'utente trova già dei feed preimpostati. Il programma funziona perfettamente al primo avvio senza alcun intervento dall'utente, questo favorisce l'approccio degli utenti alle prime armi. Basta cliccare col mouse un feed per visualizzare i messaggi del feed. L'utente può comunque modificare le impostazioni predefinite dopo il primo avvio.

1.3 Operazioni standard

Anche se il programma presenta pochi comandi, fornisce comunque all'utente la possibilità di fare tutte le operazioni standard di un reader rss, come il filtraggio dei messaggi per stringhe, l'aggiornamento automatico dei feed, l'aggiornamento su richiesta dell'utente, l'eliminazione di un feed, la visualizzazione dell'url del messaggio col browser predefinito, l'inserimento di nuovi feed e la reimpostazione del timer di aggiornamento.

1.4 Personalizzazione del formato

Come è stato detto nell'analisi dei requisiti, il programma è stato progettato per l'Unione Industriali di Como. Quindi il programma è stato personalizzato con l'implementazione del campo businesses e locations (come si vede dall'immagine sottostante).

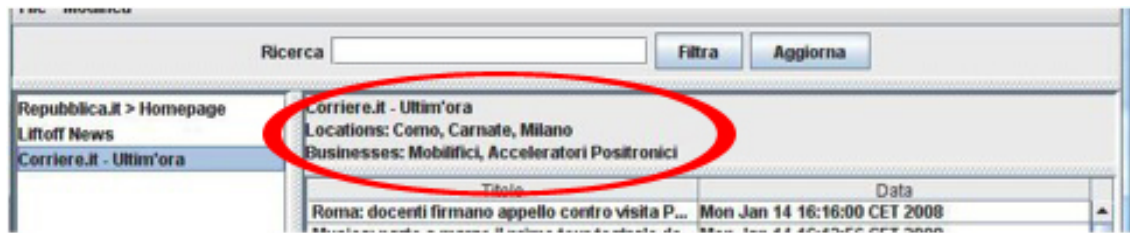


Figure 2: Il programma mostra location e business relativi

1.5 Gestione casi eccezionali

Per un programma come questo che lavora con la rete è molto importante prevedere il caso in cui i feed non esistano più o vengano cambiati. Infatti questo programma gestisce automaticamente le eccezioni dovute a problemi di questo tipo.

1.6 I vari “dialetti” del linguaggio RSS

Poiché il linguaggio RSS presenta varie versioni, il programma utilizza delle librerie che convertono i feed in un formato standard, permettendo una maggiore compatibilità con i feed. Abbiamo anche sviluppato un'applicazione che converte in RSS esteso i feed normali.

1.7 La serializzazione dei dati

Il programma salva automaticamente i dati come le impostazioni e i messaggi letti, permettendo agli utenti di poter ritrovare il programma con le impostazioni da lui scelte negli avvii successivi. Poiché vengono salvati anche i messaggi letti, negli avvii successivi non saranno più in grassetto i messaggi già letti.

Ora andiamo ad analizzare il comportamento del programma nei casi d'uso principali:

1.8 Selezione e lettura di una notizia

Come si vede dalla figura sottostante, possiamo selezionare un messaggio e il contenuto del messaggio viene visualizzato nella parte inferiore della finestra. Per visualizzare il contenuto dell'url del messaggio basta cliccare sul bottone "visita URL" o fare click destro e sarà aperto un pop menu da cui lanciare il browser. I messaggi già letti non vengono più segnalati in grassetto.

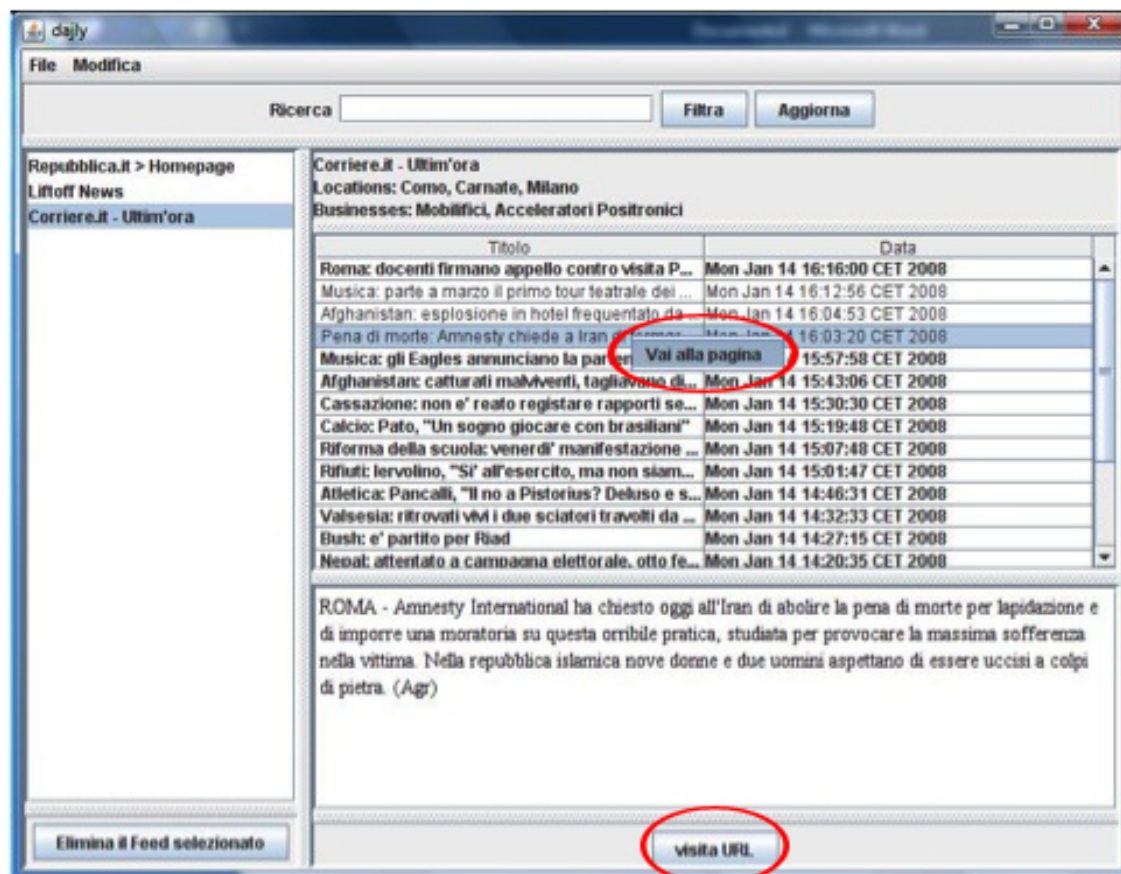


Figure 3: Selezione di una notizia

1.9 Filtraggio e ricerca di una notizia

L'utente può filtrare le notizie scrivendo nel campo "Ricerca" le chiavi di filtraggio e cliccare sul bottone "Filtra". Così rimarranno solo i messaggi che contengono le chiavi inserite.

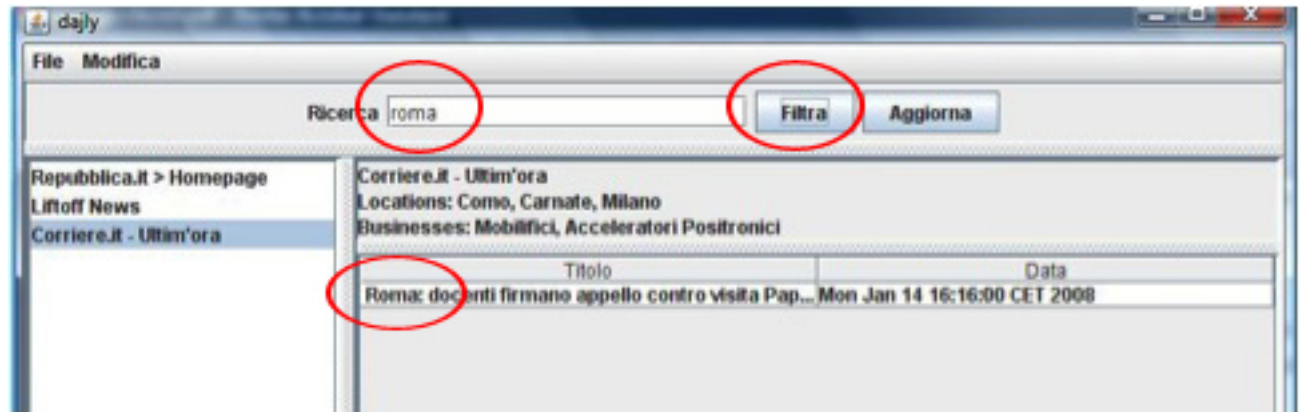


Figure 4: Filtraggio rispetto alla parola "roma"

1.10 Modifica preferenze

Nel menu "Modifica"->"Preferenze"->"Imposta timer" possiamo modificare l'intervallo di aggiornamento automatico. Cliccando su "Imposta timer" si apre una finestra(come mostrato nella



Figure 5: Menù preferenze

figura sottostante) in cui bisogna inserire il valore del tempo in millisecondi, dopo di che cliccando su "OK" il timer viene reimpostato.

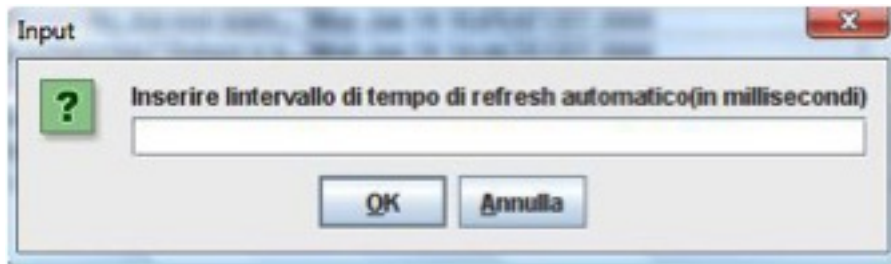


Figure 6: Inserimento tempo di refresh

2 Unit Testing

Abbiamo portato a termine il testing di unità del Model dell'applicazione da noi prodotta, facendo uso della libreria JUnit (v.3). E' possibile reperire i Test Cases assieme ai sorgenti del progetto con un semplice checkout del repository subversion. Per maggiori informazioni rimandiamo al §1.3 del Manuale Utente.

2.1 TestCase - Entry

E' stato il test case più semplice da realizzare (infatti la classe testata non è altro che una struttura dati complessa su cui non è possibile modificare i dati una volta creato l'oggetto). Poco sorprendentemente non sono stati trovati bug nel funzionamento della classe.

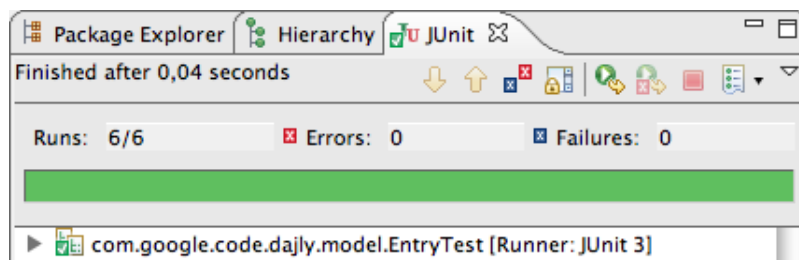


Figure 7: Report dell'esecuzione di EntryTest

2.2 TestCase - Feed

E' stato il test più difficile da realizzare a livello programmatico: data la natura dinamica degli oggetti di tipo Feed non è stato possibile confrontare i risultati ottenuti dalla nostra classe con valori costanti, ma è stato necessario ottenerli dinamicamente. Il test non ha rivelato bug.

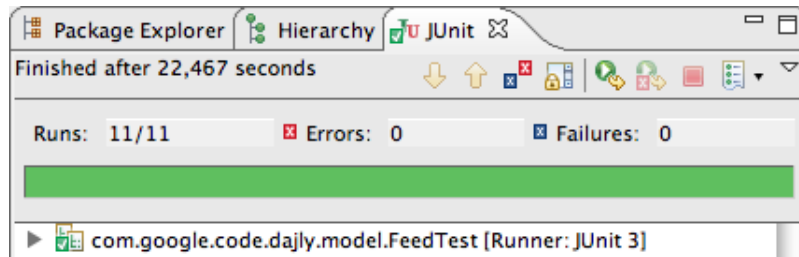


Figure 8: Report dell'esecuzione di FeedTest

2.3 TestCase - Model

Il test case è stato scritto per testare il corretto funzionamento del lancio di eventi, e delle operazioni più generali portate a termine da questa classe. Non sono stati riscontrati bug.

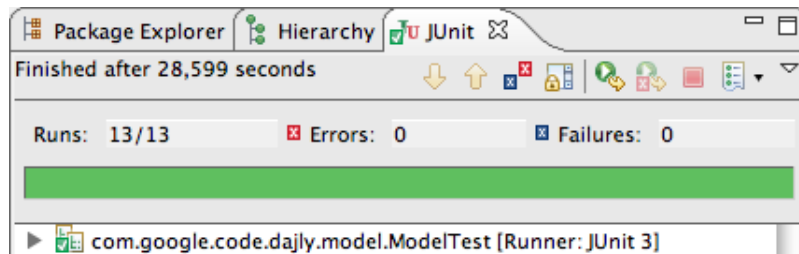


Figure 9: Report dell'esecuzione di ModelTest

2.4 Test Case - FilterType

Nella fase di scrittura del Test Case abbiamo trovato errori in due dei metodi di questa classe, nonostante la classe sia semplice (è una struttura dati) in fase di implementazione non si era pensato alla possibilità di cambiare il tipo di filtri su uno stesso oggetto. Abbiamo provveduto così a correggere l'implementazione per permettere questo utilizzo alternativo della classe.

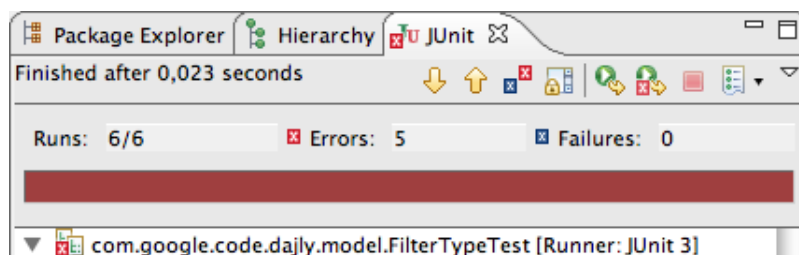


Figure 10: Report dell'esecuzione di FilterTypeTest (prima del bugfix)

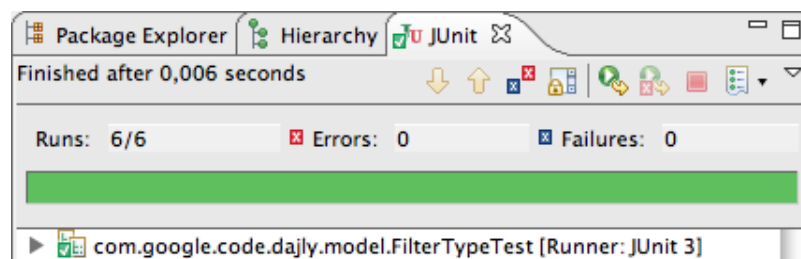


Figure 11: Report dell'esecuzione di FilterTypeTest (dopo il bugfix)