Università degli studi di Catania

Corso di Laurea Magistrale in informatica

Compilatori A.A 2014 - 2015

Antonio Fischetti W82-000021

Realizzazione di un mailParse in nodejs

Introduzione

Lo scopo del progetto è stato quello di realizzare un **parser** per l'estrazione di mail contenute all'interno di un file di testo, oppure inserite manualmente dall'utente. Per lo sviluppo del progetto è stato utilizzato il linguaggio di programmazione **nodejs** Con il quale è stata gestita tutta la parte riguardante il parsing del file di testo e le operazioni sul database. Per quanto riguarda il database utilizzato per la memorizzazione delle mail estratte è stato utilizzato **mongodb.**

Download ed Installazione (Mac Osx 10.10.3)

Prima di spiegare il funzionamento del progetto, mi soffermo sulla parte relativa all'installazione delle componenti utilizzate. E' stato necessario installare:

- L'ambiente di sviluppo per **nodejs** disponibile su: https://nodejs.org
- Il database non relazionale **mongodb** disponibile su: http://www.mongodb.com
- La libreria **mongoose** di interfaccia tra db e nodejs disponibile su: http://mongoosejs.com
- Alcuni moduli **npm** di nodejs: "file-extractor" - "express" - "node-regexp" - "ping" - "regex" il cui download è disponibile mediante l'utilizzo di npm direttamente di terminale.

Esecuzione del database

Dopo aver correttamente installato **mongodb** nella nostra macchina, ed aver settato opportunamente tutti i file ed i path necessari, il database non relazionale deve essere mandato in esecuzione. Questo viene effettuato direttamente da terminale utilizzando il comando **mongod - - path** (il path di riferimento che abbiamo scelto durante la creazione del database) . Il processo rimarrà attivo in locale nella nostra macchina e ci permetterà attraverso opportuni file di log di monitorare in tempo reale il database.

Sviluppo Applicativo

Lo sviluppo è stato suddiviso in diverse fasi. La prima è stata quella di implementare lo schema ed il modello del **database**.
Poiché non relazionale, lo schema che il database utilizza non è di tipo tabellare , ma ad oggetti, ovvero uno schema che segue il concetto di chiave-valore per la memorizzazione dei dati. Nel mio caso era necessario memorizzare il path della directory e l'indirizzo mail ed ho creato appunto una struttura contenente queste coppie di chiave valore.

Il vantaggio dell'utilizzo di **mongodb** sta nel fatto che gli oggetti che vengono creati in **nodejs** (che ricordiamo è javacript lato server), sono già confezionati in modo da essere facilmente passati al database, pronti per essere memorizzati.

Una volta aver creato la struttura del database, ho creato un piccolo menù che permette all'utente di capire come effettuare diverse scelte per le operazioni disponibili dall'applicazione.
E' importante ricordare che poiché è stato programmato utilizzando un linguaggio di programmazione lato server, tutte le operazioni vengono effettuate attraverso delle richieste **http**. Ovviamente richieste che possono essere effettuate sia da browser che da terminale.

Quando mandiamo in esecuzione l'applicativo, tramite il comando **node parser.js** , questo resta in ascolto alla porta **4000**. Per cui tutte le richieste dovranno essere reindirizzate verso questa porta di comunicazione.

E' possibile eseguire diverse operazioni: inserimento tramite directory, ovvero, inserendo l'opportuno **path** (se si lavora in locale) nel quale è contenuto il file da parsare, viene effettuato il parsing da directory. Il modo più semplice per far questo se si lavora in locale è aprire il terminale ed inserire:

curl http://localhost:4000/parseMailFromDirectory?directory='mydirectory'

Mentre se volessimo inserire una mail in modo manuale è possibile farlo utilizzando il seguente comando:

curl http://localhost:4000/parseMail?mail='mail@dominio.com'

L'applicativo inoltre permette anche di effettuare delle operazioni di ricerca e cancellazione sul database. Queste anch'esse eseguibili mediante richieste get.

curl http://localhost:4000/searchMail?mail='mail@dominio.com'
curl http://localhost:4000/removeMail?mail='mail@dominio.com'

Infine è possibile anche effettuare un ping verso i domini delle mail. In questo caso verranno ricercati sul database e successivamente verrà effettuato il ping.

curl http://localhost:4000/ping

Operazione di parsing da file

Le funzionalità del processo della mail permette di analizzare il file di testo ed estrarne tutte le mail valide. Sono supportati i file txt doc ed rtf. Per l'analisi del file di testo è stato necessario scaricare (come detto in precedenza) il pacchetto '**file-extractor**'. Questo permette di leggere uno stream da file e successivamente effettuare una fase di estrazione di tutte le mail utilizzando un **regex** (espressione regolare) per il parsing delle mail.
In rete sono disponibili diverse espressioni regolari per effettuare il parsing delle mail da un file di testo oppure da una stringa. Dopo un'accurata analisi, ho scelto di utilizzare il seguente regex:

REGEX (Parsing mail)

/[a-z0-9!#$%&'*+=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+(?:[A-Z]{2}|com|org|net|edu|gov|mil|biz|info|it)\b/

Questo è stato in grado di effettuare un corretto parsing delle mail.
Per quanto riguarda il parsing della mail contenute all'interno delle directory è stato necessario suddividere i file in due diverse directory. Poiché ci sono alcuni file che contengono mail separate da virgole, nei quali dopo aver estratto il testo contenuto al suo interno, viene effettuato uno splitting delle mail e successivamente viene effettuato il matching con il regex a disposizione, mentre altri file in cui gli indirizzi mail sono sparsi nel testo e per i quali non è servita la fase di splitting.

Operazione di salvataggio su db

Le operazioni essenziali effettuate sul database non relazionale sono state quelle di ricerca, salvataggio e cancellazioni dei dati. Infatti, quando si cerca di memorizzare degli indirizzi mail all'interno del database, viene effettuato un controllo per capire se l'indirizzo non è già stato memorizzato precedentemente, in caso negativo si può procedere con il salvataggio dello stesso. (i metodi in questione sono **find** e **save**).

Operazione di ping

Per quanto riguarda questa operazione è stato necessaria l'installazione del pacchetto 'ping' come descritto precedentemente. Per ogni indirizzo mail contenuto all'interno del database viene effettuata una ricerca del dominio associato e successivamente viene effettuato il ping in modo asincrono.

Conclusioni

Il software è ottimizzato in modo da gestire grandi quantità di mail (proprio perché utilizza un database non relazionale che permette di accedere ai dati in modo molto veloce). In più grazie al fatto che sia stato implementato lato server è possibile installarlo in una macchina server, mandarlo in esecuzione ed accedervi dall'esterno.

Il software è disponibile su **github** all'indirizzo:

https://github.com/Jacitano87/mailParser-nodejs

Per qualsiasi informazione contattare:

Antonio Fischetti : fischetti[dot]antonio[at]gmail[dot]com