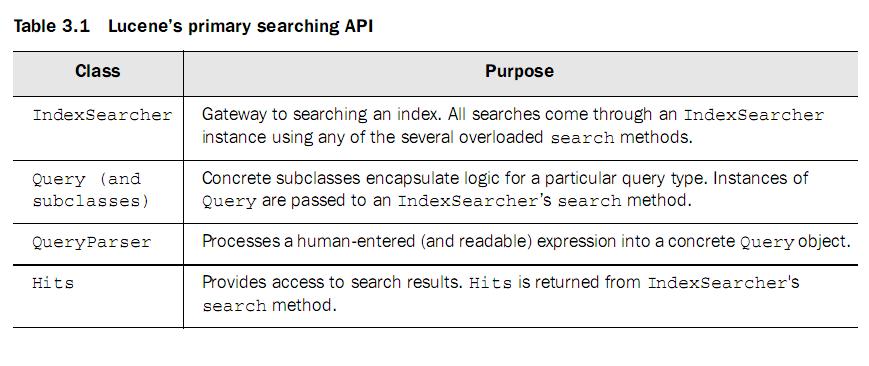
**LUCENE IN ACTION**

# CHAPTER 3: SEARCH -CHỨC NĂNG TÌM KIẾM TRONG ỨNG DỤNG

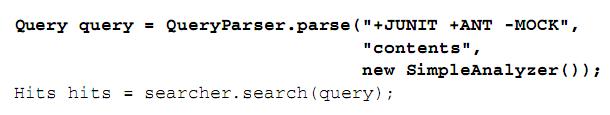
Chương này sẽ giới thiệu về việc thêm chức năng search vào ứng dụng một khi đã có dữ liệu đã được đánh chỉ mục(index).Chúng tôi sẽ trình bày một số lớp cơ bản trong Lucene hỗ trợ bạn việc tìm kiếm mà nhóm đã ứng dụng,với những chức năng cao cấp hơn sẽ được thảo luận ở các chương sau.

Bảng sau mô tả các lớp chính dùng để search trong Lucene API :



## Bộ chuyển đổi câu truy vấn của người dùng : QueryParser

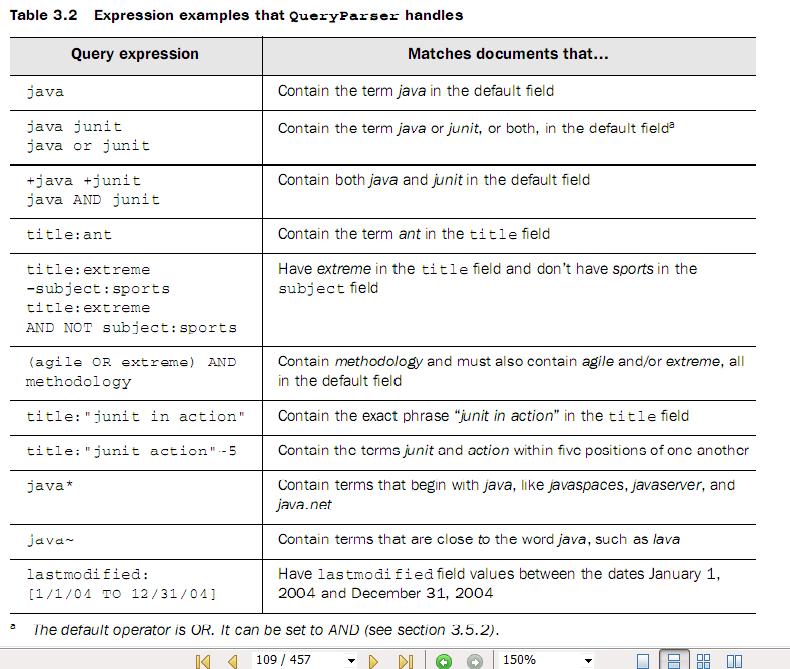
Hai yêu cầu quan trọng trong ứng dụng tìm kiếm đòi hỏi là : chuyển đổi câu truy vấn và truy xuất thông tin trả về. Hầu hết các phương thức Lucene đòi hỏi đối tượng Query.Việc chuyển đổi câu truy vấn là việc biểu diễn câu truy vấn của người dùng thành đối tượng Query phù hợp để sau đó truyền vào hàm tìm kiếm của lucene.Lucene có thể tìm ra kết quả chỉ khi câu truy vấn truyền vào là đúng định dạng của nó. Chẳng hạn như ví dụ sau:



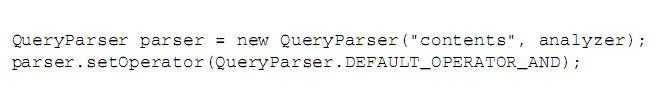
Để thực hiện được việc chuyển đổi câu truy vấn của người dùng,QueryParser cần thêm một đối tượng khác gọi là bộ phân tích Analyzer-sẽ đề cập trong chương sau.Ở đây ta chỉ cần biết là tùy vào việc chọn lựa bộ Analyzer để phân tích chuỗi truyền vào thì kết quả sẽ khác nhau.Điều ta cần nhớ là index dùng bộ Analyzer nào thì khi tìm kiếm ta nên dùng lại bộ Analyzer đó.

Như ví dụ trên,người dùng truy vấn bằng chữ viết hoa,song trong dữ liệu đánh chỉ mục đã lưu trữ thì lại là chữ thường.Do đó dùng SimpleAnalyzer –đã sử dụng trong việc index, sẽ chuyển đổi câu truy vấn thành chữ thường và vì thế sẽ tìm ra kết quả trong quá trình tìm kiếm.

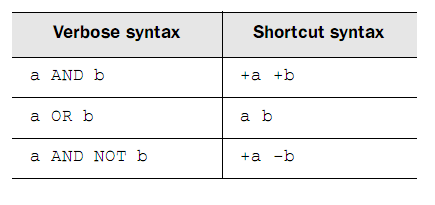
### Các biểu thức truy vấn của QueryParser



## Các toán tử luận lí

QueryParser sử dụng nhiều toán tử luận lí để thực hiện việc chuyển đổi câu truy vấn như : OR ,AND,NOT.Mặc định là OR.Chẳng hạn câu truy vấn sau : abc xyz thì sẽ được phân tích thành là abc or xyz or (abc and xyz).Để thay đổi tham số mặc định này,ta cần đặt lại toán tử cho đối tượng QueryParser.

Ngoài ra ta có thể sử dụng các từ viết tắt thay thế cho các toán tử này ví dụ như bảng dưới đây:



Tất nhiên là ta hoàn toàn có thể gom nhóm câu truy vấn lại ví dụ như là (a and b) or c…bằng ngoặc đơn.Khi đó sẽ hiểu là tìm tài liệu có chứa cả a và b hoặc là chỉ c mà thôi

Việc phân tích câu truy vấn nhằm mục đích tìm kiếm ra tài liệu có chứa từ cần tìm.Tuy nhiên việc đánh chỉ mục diễn ra trên nhiều trường khác nhau (Fields).Do đó muốn tìm tài liệu ứng với trường nào đã đánh chỉ mục thì bạn cần xác định rõ ràng.Như ví dụ trên là tìm trong trường “contents” của chỉ mục.

## Cụm truy vấn(phrase queries)

Những câu truy vấn được bao đóng trong dấu nháy kép sẽ tạo ra một cụm truy vấn.Các từ trong dấu nháy sẽ được phân tích,vì thế kết quả có thể sẽ không chính xác như cụm từ ban đầu.(chưa hoàn thành)

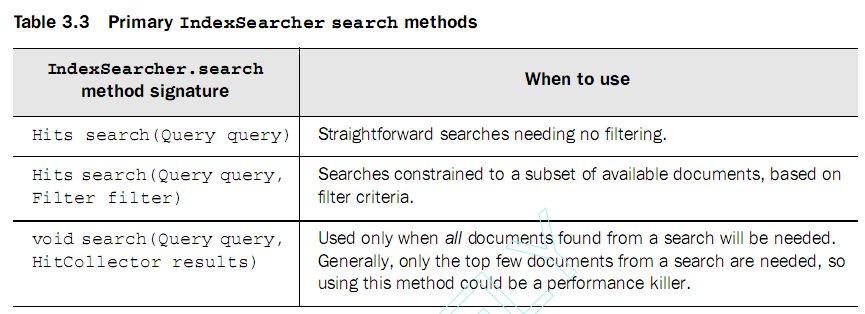
## Sử dụng lớp IndexSearcher

Tiếp theo chúng ta sẽ xem xét tới lớp IndexSearcher của Lecene.Giống như hầu hết các hàm API khác,nó rất dễ sử dụng.Việc khởi tạo thể hiện IndexSearcher thực hiện bằng các tham số khác nhau

* Bằng Directory chứa dữ liệu đã index
* Bằng đường dẫn tới file hệ thống



Sau khi tạo ra đối tượng IndexSearcher,ta sẽ gọi phương thức search để thực hiện việc tìm kiếm.Có ba phương thức chính để tìm kiếm.Song ta chủ yếu sử dụng phương thức search(Query),tức tham số là câu truy vấn Query

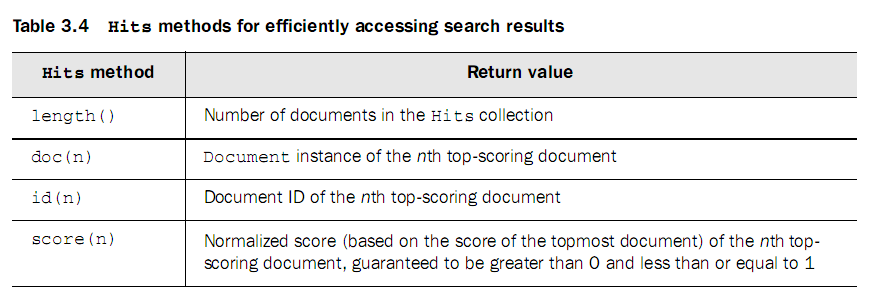


Như ta đã thấy,các phương thức tìm kiếm này đều trả về là các Hits –chứa các thông tin đã tìm kiếm được,Sau đây ta sẽ tìm hiểu thêm về Hits

### Tìm hiểu về Hits

Kết quả trả về là đối tượng Hits và được sắp xếp theo thứ tự độ chính xác.Thông qua đối tượng này ta có thể truy xuất thêm nhiều thông tin về kết quả tìm kiếm.

Sau đây sẽ điểm qua bốn phương thức chính của đối tượng này,được liệt kê dưới bảng đây.



Mặc định 100 Documents sẽ tự động được khởi tạo ban đầu và sẽ được xử lí.bộ Hits sẽ tự nó thêm vào khi người dùng truy vấn tới những tài liệu ở mức trên.Tuy nhiên điển hình vẫn là việc xem xét các tại liệu có độ ưu tiên cao hơn mà thôi.

### Phân trang thông qua Hits

Phân trang kết quả tìm kiếm là điều hết sức cần thiết trong việc trình bày kết quả trả về.Có hai hướng cài đặt chính:

* Giữ đối tượng Hits và IndexSearcher trong khi người dùng chuyển kết quả tìm kiếm
* Thực hiện truy vấn lại mỗi khi người dùng chuyển đến trang mới

Truy vấn lại thường được dùng hơn và là giải pháp tối ưu hơn.Việc này đòi hỏi phải lưu trữ trạng thái người dùng.Trong ứng dụng web,nơi người dùng gõ truy vấn ta cần lưu lại chuỗi truy vấn ban đầu.ta có thể lưu giữ trong các hidden field hoặc là cookie.và sau mỗi lần truy vấn lại thì phỉa cập nhật lại câu truy vấn của người dùng.

Một điểm cần lưu ý là mặc dù việc tìm kiếm diễn ra trên thư mục chứa dữ liệu index,song để tăng tốc độ tìm kiếm,ta nên nạp dữ liệu đọc được từ index và đẩy lên RAMDirectory.phần này đã đề cập tới trong chương trước.

## Tìm hiểu về các loại Query

Như ta đã thấy,phương thức search của đối tượng IndexSearcher đòi hỏi tham số Query.Tùy vào ứng dụng ,mục đích của câu truy vấn là tìm kiếm gì mà ta sẽ cung cấp dạng Query tương ứng hoặc là kết hợp tất cả. Trong hấu hết các ứng dụng ta sử dụng đối tượng QueryParser để chuyển đổi câu truy vấn theo từng loại thích hợp.Lucene cung cấp bốn loại Query :QueryParse,BooleanQuery,RangeQuery và TermQuery.Sau đây.ta sẽ tìm hiểu từng loại Query và lúc nào QueryParse sẽ chuyển đổi câu truy vấn thành dạng nào.

### Tìm kiếm theo giới hạn : TermQuery

Cách tìm kiếm chính trong chỉ mục là tìm trong giới hạn xác đinh nào đó gọi là Term.Term là một phần nhỏ trong chỉ mục,bao gồm tên trường(Field name) và giá trị kèm theo chẳng hạn như sau:



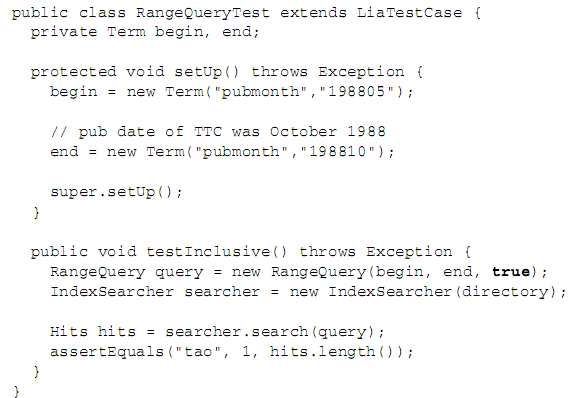
Như vậy là tất cả các kết quả nào có chứa từ khóa “java” trong phần “contents” sẽ được trả về.Chú ý rằng kiểu chữ ở đây là viết thường sẽ trùng hợp với bộ phân tích đã dùng để index dữ liệu nếu không kết quả sẽ không chính xác

Nói thêm là việc sử dụng TermQuerys rất hữu dụng khi bạn cần tìm kiếm tài liệu theo từ khóa,nếu tài liệu đã được index theo dạng là Field.Keyword().

* Nếu câu truy vấn chỉ chứa một từ thì QueryParser sẽ tạo ra TermQuery để truy vấn

### Tìm kiếm theo dãy :RangeQuery

Trong trường hợp mà dữ liệu được index đã được sắp xếp theo thứ tự giống kiểu từ điển,thì việc tìm kiếm theo dãy tỏ ra hiệu quả hơn cả.RangeQuery của Lucene cho phép ta tìm kiếm những mục nằm trong giới hạn đầu và cuối của dãy cần tìm.

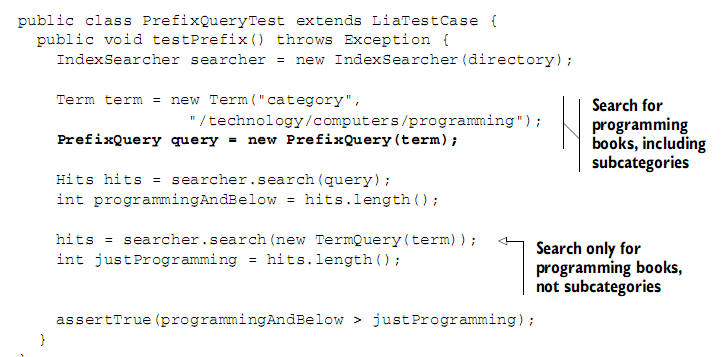


* Nếu biểu thức truy vấn có dạng :
  + [begin to end] : lấy các tài liệu nằm trong dãy này
  + {begin to end} : lấy các tài liệu nằm ngoài khoảng này

Thì QueryParse sẽ khởi tạo RangeQuery .

### Tìm kiếm dựa trên chuỗi :PrefixQuery

Việc tìm kiếm theo kiểu này sẽ cho những kết quả chứa các tài liệu mà bắt đầu bằng chuỗi xác định.cách này thi thoảng phát huy tác dụng trong ít trường hợp chẳng hạn như là bạn có chỉ mục về danh mục sách và ta muốn tìm kiếm những cuốn sách thuộc danh mục nào đó.như ví dụ sau:



* Nếu câu truy vấn kết thúc là dấu “ \* ” thì term dạng PrefixQuery sẽ đươc tạo ra

### Kết hợp các truy vấn : BooleanQuery

Những loại truy vấn trên có thể được kết hợp lại với nha bằng cách sư dụng đối tượng BooleanQuery với phương thức add.Các truy vấn này có thể kết hợp bằng NOT,AND hoặc là OR.

Nếu required=prohibited=true: kết hợp các truy vấn bằng toán tử AND

Nếu required=prohibited=false: kết hợp các truy vấn bằng toán tử OR

Nếu required=true và prohibited=false: kết hợp các truy vấn bằng toán tử NOT

* Nếu câu truy vấn là kết hợp của nhiều mục bao đóng bởi dấu ngoặc đơn truy vấn dạng BooleanQuery sẽ đươc tạo ra

### Tìm kiếm theo cụm: PhraseQuery

Index chứa các thông tin về vị trí giữa các từ với nhau.PhraseQuery sử dụng các thông tin này để xác định các tài liệu chứa các từ có quan hệ với các từ khác trong cùng tài liệu.chẳng hạn, bạn đang tìm chuỗi sau:”the quick brown fox can jumped over the lazy dog”.Nếu không biết chính xác cụm truy vấn này ,ta sẽ tìm các tài liệu mà có chứa các từ “quick” và “fox” gần nhau.nếu chỉ sử dụng TermQuery-truy vấn theo từ đơn thì ta không thể tìm được tài liệu nào có hai từ kể trên gần nhau

Ta gọi khoảng cách tối đa giữa hai từ là slop và Distance là số vị trí cần di chuyển của từ đó trong cụm từ cần tìm so với cụm từ trong tài liệu.Khi đó PhraseQuery sẽ so sánh giữa các slop này để cho ra kết quả gần giống nhất

### Tìm kiếm theo wildcard: WildcardQuery

Đây là cách tìm kiếm cho những loại truy vấn mà thiếu đi một vài từ trong đó,chẳng hạn như là tìm từ khóa “?oo\*” (đúng ra là book).

Dấu “?” có nghĩa là không có từ nào hoặc là một từ bất kì

Dấu “\*” có nghĩa là không có từ nào hoặc là nhiều từ bất kì

### Tìm kiếm theo những từ giống nhau: FuzzyQuery

Loại truy vấn cuối cùng được đề cập tới là FuzzyQuery.Cách truy vấn này sẽ xem những từ tương tự nhau là một từ.Lucene sử dụng thuật toán về khoảng cách từ gọi là Levenshtein distance ,chi tiết có thể tham khảo tại đây (<http://www.merriampark.com/ld.htm>).

Ta chỉ cần hiểu rằng cách tính khoảng cách ở đây khác hơn so với khoảng cách trong loại truy vấn PhraseQuery,đó là số từ cần thêm.xóa để biến một từ thành từ khác.Do đó các từ tương tự nhau ở đây được xét là các từ mà xóa bỏ hày thêm một số từ nào đó để trở thành từ khác tuy nhiên mức độ ưu tiên cho các từ có số từ cần thêm vào hay bớt đi là ít hơn.

* Nếu câu truy vấn kết thúc là dấu “~” thì truy vấn dạng FuzzyQuery sẽ đươc tạo ra