عنوان: **استفاده از لوسین برای برجسته سازی عبارت جستجو شده در نتایج حاصل**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۷ ۱۲:۳۸
آدرس: www.dotnettips.info
گروه‌ها: Lucene.NET

قسمت جستجوی سایت جاری رو با استفاده از لوسین بازنویسی کردم. خلاصه‌ای از نحوه انجام این‌کار رو در ادامه ملاحظه خواهید کرد:

1) دریافت کتابخانه‌های لازم

نیاز به کتابخانه‌های Lucene.NET و همچنین Lucene.Net Contrib است که هر دو مورد را به سادگی توسط NuGet می‌توانید دریافت و نصب کنید.
Highlighter استفاده شده، در کتابخانه Lucene.Net Contrib قرار دارد. به همین جهت این مورد را نیز باید جداگانه دریافت کرد.

2) تهیه منبع داده

در اینجا جهت سادگی کار فرض کنید که لیستی از مطالب را به فرمت زیر دراختیار داریم:

```
public class Post
{
    public int Id { set; get; }
    public string Title { set; get; }
    public string Body { set; get; }
}
```

تفاوتی نمی‌کند که از چه منبع داده‌ای استفاده می‌کنید. آیا قرار است یک سری فایل متنی ساده موجود در یک پوشه را ایندکس کنید یا تعدادی رکورد بانک اطلاعاتی؛ از NHibernate استفاده می‌کنید یا از Entity framework و یا از ADO.NET. کتابخانه Lucene مستقل است از منبع داده مورد استفاده و تنها اطلاعاتی با فرمت شیء Document معرفی شده به آن‌را می‌شناسد.

3) تبدیل اطلاعات به فرمت Lucene.NET

همانطور که عنوان شد نیاز است هر رکورد از اطلاعات خود را به شیء Document نگاشت کنیم. نمونه‌ای از اینکار را در متد ذیل مشاهده می‌نمائید:

```
static Document MapPostToDocument(Post post)
{
    var postDocument = new Document();
    postDocument.Add(new Field("Id", post.Id.ToString(), Field.Store.YES, Field.Index.NOT_ANALYZED));
    postDocument.Add(new Field("Title", post.Title, Field.Store.YES, Field.Index.ANALYZED,
Field.TermVector.WITH_POSITIONS_OFFSETS));
    postDocument.Add(new Field("Body", post.Body, Field.Store.YES, Field.Index.ANALYZED,
Field.TermVector.WITH_POSITIONS_OFFSETS));
    return postDocument;
}
```

این متد وهله‌ای از شیء Post را دریافت کرده و آن‌را تبدیل به یک سند Lucene می‌کند.
کار با ایجاد یک وهله از شیء Document شروع شده و سپس اطلاعات به صوت فیلدهایی به این سند اضافه می‌شوند.

توضیحات آرگومان‌های مختلف سازنده کلاس Field:
- در ابتدا نام فیلد مورد نظر ذکر می‌گردد.
- سپس مقدار متناظر با آن فیلد، به صورت رشته باید معرفی شود.
- آرگومان سوم آن مشخص می‌کند که اصل اطلاعات نیز علاوه بر ایندکس شدن باید در فایل‌های Lucene ذخیره شوند یا خیر. توسط Field.Store.YES مشخص می‌کنیم که بله؛ علاقمندیم تا اصل اطلاعات نیز از طریق Lucene قابل بازیابی باشند. این مورد جهت نمایش سریع نتایج جستجوها می‌تواند مفید باشد. اگر قرار نیست اطلاعاتی را از این فیلد خاص به کاربر نمایش دهید می‌توانید از گزینه Field.Store.NO استفاده کنید. همچنین امکان فشرده سازی اطلاعات ذخیره شده با انتخاب گزینه

Field.Store.COMPRESS نیز میسر است.
- توسط آرگومان چهارم آن تعیین خواهیم کرد که اطلاعات فیلد مورد نظر ایندکس شوند یا خیر. مقدار Field.Index.NOT_ANALYZED سبب عدم ایندکس شدن فیلد Id می‌شوند (چون قرار نیست روی id در قسمت جستجوی عمومی سایت، جستجویی صورت گیرد). به کمک مقدار Field.Index.ANALYZED، مقدار معرفی شده، ایندکس خواهد شد.
- پارامتر پنجم آن‌را جهت سرعت عمل در نمایان سازی/برجسته کردن و highlighting عبارات جستجو شده در متن‌های یافت شده معرفی کرده‌ایم. الگوریتم‌های متناظر با این روش در فایل‌های Lucene.Net Contrib قرار دارند.

یک نکته

اگر اطلاعاتی را که قرار است ایندکس کنید از نوع HTML می‌باشند، بهتر است تمام تگ‌های آن‌را پیش از افزودن به لوسین حذف کنید. به این ترتیب نتایج جستجوی دقیق‌تری را می‌توان شاهد بود. برای این منظور می‌توان از متد ذیل کمک گرفت:

```
public static string RemoveHtmlTags(string text)
{
    return string.IsNullOrEmpty(text) ? string.Empty : Regex.Replace(text, @"<(.|\n)*?>",
string.Empty);
}
```

4) تهیه Full text index به کمک Lucene.NET

تا اینجا توانستیم اطلاعات خود را به فرمت اسناد لوسین تبدیل کنیم. اکنون ثبت و تبدیل آن‌ها به فایل‌های Full text search لوسین به سادگی زیر است:

```
static readonly Lucene.Net.Util.Version _version = Lucene.Net.Util.Version.LUCENE_29;
public static void CreateIdx(IEnumerable<Post> dataList)
{
    var directory = FSDirectory.Open(new DirectoryInfo(Environment.CurrentDirectory +
"\\LuceneIndex"));
    var analyzer = new StandardAnalyzer(_version);
    using (var writer = new IndexWriter(directory, analyzer, create: true, mfl:
IndexWriter.MaxFieldLength.UNLIMITED))
    {
         foreach (var post in dataList)
        {
            writer.AddDocument(MapPostToDocument(post));
        }

        writer.Optimize();
        writer.Commit();
        writer.Close();
        directory.Close();
    }
}
```

ابتدا محل ذخیره سازی فایل‌های full text search مشخص می‌شوند. سپس آنالیز کننده اطلاعات باید معرفی شود. در ادامه به کمک این اطلاعات، شیء IndexWriter ایجاد و مستندات لوسین به آن اضافه می‌شوند. در آخر، این اطلاعات بهینه سازی شده و ثبت نهایی صورت خواهد گرفت.
ذکر version در اینجا ضروری است؛ از این جهت که اگر ایندکسی با فرمت مثلا LUCENE_29 تهیه شود ممکن است با نگارش بعدی این کتابخانه سازگار نباشد و در صورت ارتقاء، نتایج جستجوی انجام شده، کاملا بی‌ربط نمایش داده شوند. با ذکر صریح نگارش، دیگر این اتفاق رخ نخواهد داد.

نکته

StandardAnalyzer توکار لوسین، امکان دریافت لیستی از واژه‌هایی که نباید ایندکس شوند را نیز دارا است. اطلاعات بیشتر در اینجا .

5) به روز رسانی ایندکس‌ها

به کمک سه متد ذیل می‌توان اطلاعات ایندکس‌های موجود را به روز یا حذف کرد:

```
public static void UpdateIndex(Post post)
{
       var directory = FSDirectory.Open(new DirectoryInfo(Environment.CurrentDirectory +
"\\LuceneIndex"));
       var analyzer = new StandardAnalyzer(_version);
       using (var indexWriter = new IndexWriter(directory, analyzer, create: false, mfl:
IndexWriter.MaxFieldLength.UNLIMITED))
       {
           var newDoc = MapPostToDocument(post);

            indexWriter.UpdateDocument(new Term("Id", post.Id.ToString()), newDoc);
            indexWriter.Commit();
            indexWriter.Close();
            directory.Close();
        }
}

public static void DeleteIndex(Post post)
{
        var directory = FSDirectory.Open(new DirectoryInfo(Environment.CurrentDirectory +
"\\LuceneIndex"));
        var analyzer = new StandardAnalyzer(_version);
        using (var indexWriter = new IndexWriter(directory, analyzer, create: false, mfl:
IndexWriter.MaxFieldLength.UNLIMITED))
        {
            indexWriter.DeleteDocuments(new Term("Id", post.Id.ToString()));
            indexWriter.Commit();
            indexWriter.Close();
            directory.Close();
         }
}

public static void AddIndex(Post post)
{
     var directory = FSDirectory.Open(new DirectoryInfo(Environment.CurrentDirectory +
"\\LuceneIndex"));
     var analyzer = new StandardAnalyzer(_version, getStopWords());
     using (var indexWriter = new IndexWriter(directory, analyzer, create: false, mfl:
IndexWriter.MaxFieldLength.UNLIMITED))
     {
          var searchQuery = new TermQuery(new Term("Id", post.Id.ToString()));
          indexWriter.DeleteDocuments(searchQuery);

           var newDoc = MapPostToDocument(post);
           indexWriter.AddDocument(newDoc);
           indexWriter.Commit();
           indexWriter.Close();
           directory.Close();
       }
}
```

تنها نکته مهم این متدها، استفاده از متد IndexWriter با پارامتر create مساوی false است. به این ترتیب فایل‌های موجود بجای از نو ساخته شدن، به روز خواهند شد.
محل فراخوانی این متدها هم می‌تواند در کنار متدهای به روز رسانی اطلاعات اصلی در بانک اطلاعاتی برنامه باشند. اگر رکوردی اضافه یا حذف شده، ایندکس متناظر نیز باید به روز شود.

6) جستجو در اطلاعات ایندکس شده و نمایش آن‌ها به همراه نمایان/برجسته سازی عبارات جستجو شده

قسمت نهایی کار با لوسین و اطلاعات ایندکس‌های تهیه شده، کوئری گرفتن از آن‌ها است. متدهای کامل مورد نیاز را در ذیل مشاهده می‌کنید:

```
public static void Query(string term)
{
     var directory = FSDirectory.Open(new DirectoryInfo(Environment.CurrentDirectory +
"\\LuceneIndex"));
     using (var searcher = new IndexSearcher(directory, readOnly: true))
     {
          var analyzer = new StandardAnalyzer(_version);
          var parser = new MultiFieldQueryParser(_version, new[] { "Body", "Title" }, analyzer);
          var query = parseQuery(term, parser);
          var hits = searcher.Search(query, 10).ScoreDocs;
```

```
        if (hits.Length == 0)
        {
             term = searchByPartialWords(term);
             query = parseQuery(term, parser);
             hits = searcher.Search(query, 10).ScoreDocs;
         }

         FastVectorHighlighter fvHighlighter = new FastVectorHighlighter(true, true);
         foreach (var scoreDoc in hits)
         {
             var doc = searcher.Doc(scoreDoc.doc);
             string bestfragment = fvHighlighter.GetBestFragment(
                             fvHighlighter.GetFieldQuery(query),
                             searcher.GetIndexReader(),
                             docId: scoreDoc.doc,
                             fieldName: "Body",
                             fragCharSize: 400);
              var id = doc.Get("Id");
              var title = doc.Get("Title");
              var score = scoreDoc.score;
              Console.WriteLine(bestfragment);
          }

          searcher.Close();
          directory.Close();
    }
}

private static Query parseQuery(string searchQuery, QueryParser parser)
{
    Query query;
     try
     {
         query = parser.Parse(searchQuery.Trim());
     }
     catch (ParseException)
     {
         query = parser.Parse(QueryParser.Escape(searchQuery.Trim()));
     }
     return query;
}

private static string searchByPartialWords(string bodyTerm)
{
    bodyTerm = bodyTerm.Replace("*", "").Replace("?", "");
    var terms = bodyTerm.Trim().Replace("-", " ").Split(' ')
                            .Where(x => !string.IsNullOrEmpty(x))
                            .Select(x => x.Trim() + "*");
    bodyTerm = string.Join(" ", terms);
    return bodyTerm;
}
```

توضیحات:

اکثر سایت‌ها را که بررسی کنید، جستجوی بر روی یک فیلد را توضیح داده‌اند. در اینجا نحوه جستجو بر روی چند فیلد را به کمک MultiFieldQueryParser ملاحظه می‌کنید.

نکته‌ی مهمی را هم که در اینجا باید به آن دقت داشت، حساس بودن لوسین به کوچکی و بزرگی نام فیلدهای معرفی شده است و در صورت عدم رعایت این مساله، جستجوی شما نتیجه‌ای را دربر نخواهد داشت.

در ادامه برای parse اطلاعات، از متد کمکی parseQuery استفاده شده است. ممکن است به ParseException بخاطر یک سری حروف خاص بکارگرفته شده در عبارات مورد جستجو برسیم. در اینجا می‌توان توسط متد QueryParser.Escape، اطلاعات دریافتی را اصلاح کرد.

سپس نحوه استفاده از کوئری تهیه شده و متد Search را ملاحظه می‌کنید. در اینجا بهتر است تعداد رکوردهای بازگشت داده شده را تعیین کرد (به کمک آرگومان دوم متد جستجو) تا بی‌جهت سرعت عملیات را پایین نیاورده و همچنین مصرف حافظه سیستم را نیز بالا نبریم.

ممکن است تعداد hits یا نتایج حاصل صفر باشد؛ بنابراین بد نیست خودمان دست به کار شده و به کمک متد searchByPartialWords، ورودی کاربر را بر اساس زبان جستجوی ویژه لوسین اندکی بهینه کنیم تا بتوان به نتایج بهتری دست یافت.

در آخر نحوه کار با ScoreDocs یافت شده را ملاحظه می‌کنید. اگر محتوای فیلد را در حین ایندکس سازی ذخیره کرده باشیم، به کمک متد doc.Get می‌توان به اطلاعات کامل آن نیز دست یافت.

همچنین نکته دیگری را که در اینجا می‌توان ملاحظه کرد استفاده از FastVectorHighlighter می‌باشد. به کمک این Highlighter ویژه می‌توان نتایج جستجو را شبیه به نتایج نمایش داده شده توسط موتور جستجوی گوگل درآورد. برای مثال اگر شخصی ef code first را جستجو کرد، توسط متد GetBestFragment، بهترین جزئی که شامل بیشترین تعداد حروف جستجو شده است، یافت گردیده و همچنین به کمک تگ‌های B، ضخیم نمایش داده خواهند شد.

نظرات خوانندگان

نویسنده: Humid
تاریخ: ۱۳۹۱/۰۵/۰۵ ۱:۵۰

سلام و خسته نباشید. من از طریق سایت شما با این کتابخونه آشنا شدم. دارم باهاش کار می‌کنم اما یه مشکلی باهاش دارم. و اون اینه که قبلا توی سرچ من اگر کلمه "ما" رو سرچ می‌کردم 2600 تا نتیجه برمیگردوند اما الان 20 تا. چرا؟
و سوال بعد من اینه که چطور می‌تونم ایندکس کلمه پیدا شده توی متن رو پیدا کنم؟ چون نرم افزار خواسته شده رو می‌خوان مثه نرم افزار نور باشه.
ممنون میشم راهنماییم کنید.
تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۵ ۸:۲۳

- احتمالا به عنوان stopword معرفی شده. این نوع کلمات ایندکس نخواهند شد. کلمه «ما» ارزش جستجو ندارد مانند «از»، «و»، «به»، «تا» و امثال آن.
- در مطلب فوق به قسمت ذیل دقت کنید. این Id همان Id واقعی یک رکورد در دیتابیس است که به عنوان یک سند لوسین ثبت شده:

```
var id = doc.Get("Id");
```

نویسنده: Humid
تاریخ: ۱۳۹۱/۰۵/۰۵ ۱۱:۱۷

ممنون آقای نصیری اما در زبان عربی "ما" یک کلمه تقریبا مهم هست. چطور می‌تونم اون رو از توی stopword خارج کنم.
سوال دیگه اینکه من با سرچ قبلیم که اول یه select میزدم با لایک و رکوردهایی که اون کلمه توش بود و پیدا میکردم و با استفاده از سرچ حرف به حرف در میاوردم کلمه رو. اما الان مثلا اگر "حسین" رو سرچ کنم با لوسین 10 نتیجه و با سرچ قبلی 31 نتیجه میده. چطور می‌تونم نزدیک کنم به نتیجه واقعی؟ البته در سرچ با لوسین از کاراکتر * استفاده هم کردم فرقی نکرد.

چطور می‌تونم در مبحث اِعراب‌های کلمات عربی از لوسین استفاده کنم؟ آیا از زبان عربی و اِعراب گذاری‌ها پشتیبانی می‌شه در این کتابخانه؟
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۵ ۱۱:۳۴

- 10 نتیجه احتمالا به تنظیم زیر مرتبط است:

```
searcher.Search(query, 10)
```

در اینجا فقط 10 نتیجه بازگشت داده می‌شود (پارامتر دوم ذکر شده).

- در مورد اعراب زبان عربی به صورت پیش فرض خیر. اما اگر به کدهای فوق دقت کرده باشید از یک StandardAnalyzer توکار استفاده شده. این مورد یک سری تنظیمات ابتدایی را به همراه دارد. اگر کارکرد آن مورد قبول شما نیست می‌تونید خودتون یک Analyzer سفارشی رو توسعه بدید.
برای مثال یک نمونه سورس باز رو اینجا می‌تونید پیدا کنید که مباحث اعراب گذاری، ی و ک فارسی و عربی، یک سری stopword

فارسی و مسایل دیگر را هم لحاظ کرده.

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۲۶ ۲۱:۵۸

سلام آقای نصیری،
دو تا سوال داشتم ازتون:
-1 از این توابع مثلاً باید موقع درج مقاله(پست یا ..) در بانک اطلاعاتی برای ایندکس جدید استفاده کرد و موقع ویرایش و حذف مقاله(پست یا ..) هم از توابع معرفی شده متناسب استفاده کرد؟

-2 جستجوی پیشرفته به چه صورت هستش؟مثلاً تاریخ درج مقاله از ... تا ... ،نام نویسنده ،کلمه کلیدی و ... (که هر کدام از این موارد میتونه به صورت اختیاری و یا اجباری باشه).

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۶ ۲۲:۲

- بله. نیاز است مدام این ایندکس را به روز نگه داشت.
- برای این موارد متداول از تاریخ تا تاریخ، از همان SQL معمولی استفاده کنید. هر جایی که امکان تعریف ایندکس و کوئری‌های SQL ایی که از ایندکس استفاده می‌کنند، وجود دارد، روش‌های متداول SQLایی بهینه‌ترین روش‌ها هستند. هدف در اینجا، full text search است بر روی انبوهی text. جستجوی بسیار سریع روی فیلدهای ایندکس نشده حجیم متنی با کیفیتی بالا. این هدف full text search است. چیزی مثل جستجوی گوگل.
در غیر اینصورت نیاز خواهید داشت از عبارات sql به همراه like استفاده کنید که ... بسیار کند هستند؛ چون باید کل جداول و بانک اطلاعاتی را هربار اسکن کنند و در حالت استفاده از like از ایندکس استفاده نمی‌شود.

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۲۶ ۲۲:۴۱

-منظورتون از "روش‌های متداول SQLایی بهینه‌ترین روش‌ها هستند" چیه؟

-پس در جستجوهای پیشرفته باید از روش معمول(like بدون استفاده از full text search) استفاده کرد و در جستجوهای تک کلمه ای مثل همین سایت باید از lucene(یا full text search) استفاده کرد،درسته؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۶ ۲۳:۹

- اگر کوئری SQL شما از ایندکس استفاده می‌کند نیازی به روش‌های full text search ندارید و موتورهای بانک اطلاعاتی به اندازه کافی برای مدیریت این نوع موارد سریع و بهینه هستند.
- جستجوی این سایت و یا full text search تک کلمه‌ای نیست. می‌تونید جمله هم بنویسید. کلا برای بهبود سرعت، کاهش مصرف CPU و حافظه کوئری‌های SQL ایی که از like استفاده می‌کنند، روش full text search پیشنهاد می‌شود.
استفاده از like در عبارات SQL روش بهینه‌ای نیست چون هربار full table scan صورت می‌گیرد (تصور کنید 100 نفر در حال جستجوی مطالبی در سایت هستند. در این حالت مصرف CPU، استهلاک هارد و مصرف بالای حافظه را درحین اسکن کامل جداول بانک اطلاعاتی درنظر بگیرید)

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۲۷ ۰:۰

"برای این موارد متداول از تاریخ تا تاریخ، از همان SQL معمولی استفاده کنید "
منظورتون اینه که تو جستجوهای پیشرفته باید از روش معمول و like(یعنی بدون استفاده از full text search و lucene)استفاده بشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۷ ۰:۵

برای جستجوی تاریخ از like استفاده نمی‌شود. like معادل متد الحاقی Contains در LINQ to EF است + توضیح دادم چرا like مناسب نیست.

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۷/۱۲ ۱۵:۲۳

البته با تلفیقی از جستجوی لوسین و کوئری روی داده‌های رنج دار مثل **از تاریخ تا تاریخ** میتوان نتیجه ای سریعتری بدست آورد چون میتوان شناسه آبجکت را در لوسین ذخیره کرد و در زمانی که کوئری دریافت شد بقیه کارها را انجام داد یا تاریخ را هم لوسین ذخیره کرد و روی آبجکت‌های گرفته شده کار کرد یا که با توجه به شناسه‌های بدست آمده از بانک کوئری گرفت

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۹/۰۷ ۸:۴۵

البته من تست نکردم ولی شاید بتوان با توجه به این که میتوان تاریخ را بصورت عدد ذخیره کرد و نیز با استفاده از عبارات با قاعده تمام وظایف جستجو را به لوسین سپرد.

نویسنده: محسن عباس آبادعربی
تاریخ: ۱۳۹۱/۰۹/۲۰ ۱۷:۴۳

سلام؛
من تازه mvc رو شروع کردم در پروژه ای نیاز به استفاده از لوسین دارم خواهش میکنم یک نمونه برنامه که با استفاده از لوسین می‌باشد را در سایت قرار دهید
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۲۰ ۱۷:۴۶

این دو برچسب را در سایت دنبال کنید:
برچسب مطالب مرتبط با لوسین
برچسب اشتراک‌گذاری‌های مرتبط با لوسین

نویسنده: حسین غلامی
تاریخ: ۱۳۹۱/۱۰/۱۱ ۰:۱۸

سلام؛
در قسمت FastVectorHighlighter متدی با نام () GetIndexReader شناخته شده نیست.
آیا از ورژن لوسین است؟

نویسنده: حسین غلامی
تاریخ: ۱۳۹۱/۱۰/۱۱ ۰:۲۲

در قسمت متد Query اگر ما خواسته باشیم اطلاعات رو در قالب همون شی (مثلا post) برگردونیم ، چطور باید این کار رو انجام داد؟
مثلا قسمت جستجوی همین سایت.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۱ ۰:۲۵

بله. در نگارش جدید یک سری متدهای Get دار به خاصیت تبدیل شدن. مثلا متد GetIndexReader تبدیل شده به خاصیتی به نام IndexReader و مواردی از این دست.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۱ ۰:۲۸

سورس قسمتی از جستجوی سایت در دسترس است . ولی در کل مقادیری رو که در ایندکس ذخیره کردید به صورت زیر قابل بازیابی است. پس از آن نگاشت نهایی را خودتان باید انجام دهید.

```
var prop = doc.Get("prop_name");
```

نویسنده: M.Q
تاریخ: ۱۳۹۲/۰۲/۲۱ ۱۶:۴۲

با سلام

من میخوام توی lucene از عبارات منطقی and و or استفاده کنم ولی هرچی تست می‌کنم جواب دقیق و کاملی نمیگیرم (جستجو انجام میشه ولی مثلا از 3 آیتمی که حتما باید برگردونه 2 تاشو بر میگردونه).

آیا عبارات منطقی رو میشه در lucene استفاده کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۲۱ ۱۷:۰

بله. بهترین مرجع برای این مسایل « Apache Lucene - Query Parser Syntax » است و همچنین اینجا برای مثال‌های بیشتر. به علاوه کلاس BooleanQuery هم برای اینکار وجود دارد.

نویسنده: آیمو
تاریخ: ۱۳۹۲/۰۸/۱۵ ۱۸:۱۵

سلام؛ من پروژه لوسین رو که شما ضمیمه کرده بودین توبخش استفاده از AutoComplete Jquery هم دانلود کردم و عین همونا رو پیاده کردم و همه چیز داره خوب کار میکنه. منتها شما اونجا چند تا post رو یک جا به لوسین دادین تا ایندکس کنه و لوسین هم برای همشون یه فایل میسازه . اما من که هر چند وقت یه بار تو سایت یه مطلبو ایندکس میکنم برای هر کدوم یه فایل ساخته و خب اگه تعداد مطلبام زیاد باشه این همینجور برای همشون تو دایرکتوری خودش فایل‌های یک کیلو بایتی میسازه .ِایا این درسته؟ نمیدونم مشکل از کجاست! اگه میشه راهنمایی کنین....

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۱۵ ۲۰:۴۱

Too many open files

نویسنده: آیمو
تاریخ: ۱۳۹۲/۰۸/۱۶ ۱۲:۱

ممنون به خاطر پاسختون . اون لینکو نگاه کردم و ایتم هایی که گفته بود رو من رعایت کرده بودم . اما همین طور فایل اضاف می‌کرد . بعد اومدم ببینم تا چند تا این همینجوری میخواد فایل بسازه ... یه برنامه ساده نوشتم و توش همون درخواست رو با httpClient بهش میدادم و و اونم می‌ساخت . جالب اینجاست که تا مثلا 30 تا ایتم فایل با پسوندها مختلف می‌ساخت بعد یه فایل Write Lock می‌ساخت .همشونو با هم یکی میکرد با یه اسمه جدید و بعد باز شروع به ایندکس کردن میکرد .من با این برنامه چیزی حدود 10 هزار درخواست رو براش فرستادم و اونم ایندکس کرد مثله برق بدونه اینکه تعداد فایل‌ها از ماکزیمم 50 تا ایتم بیشتر بشه . من بی خود نگران بود . دستتون درد نکنه

نویسنده: جهش
تاریخ: ۱۳۹۳/۰۴/۰۳ ۱۰:۵۰

سلام
از تابع **CreateIdx** کجا باید استفاده کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۰۳ ۱۱:۰

چون یک لیست را قبول می‌کند، نیاز است یکبار برای ایندکس کردن لیستی از اسناد موجود ایندکس نشده اجرا شود (در پنل مدیریتی برنامه مثلا). در سایر موارد (مانند افزوده شدن، به روز رسانی یا حذف یک رکورد) از مورد 5 استفاده کنید.