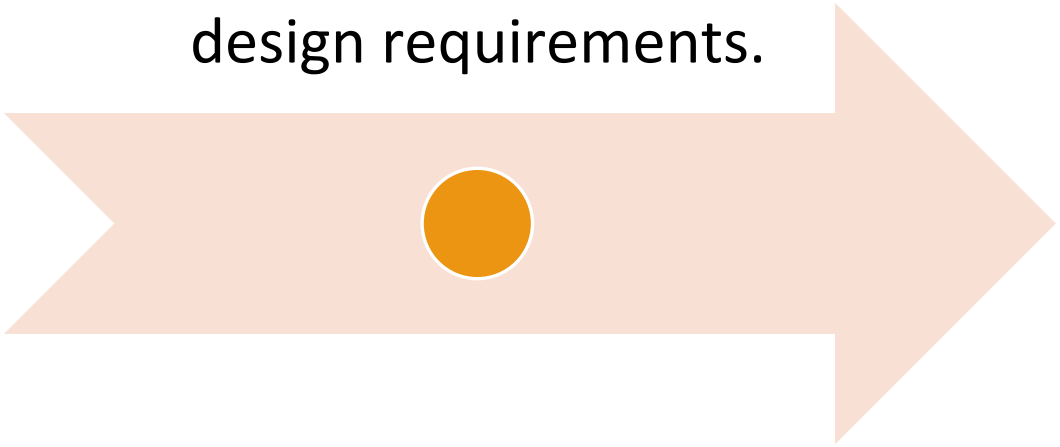
Implementing Site Search with Umbraco Examine

Alex Lindgren
alindgren@flightpath.com
twitter.com/alexlindgren

Why we love Umbraco

We find that it is often the fastest way we know to develop great sites that meet the functional, technical and design requirements.



Why Search?

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi Site visitors are in a hurry… ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu nulla pariatur. Ut enim minima veniam nisi ut aliquid, Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit aniSed ut perspiciatis sit unde omnis totam The want to find natus voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa relevant content quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim quickly. ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur. . Ut enim ad minima veniam, quis nostrum

Umbraco Search

One (of many) reasons to love Umbraco is that it comes with a great search and indexing service.

Umbraco provides a powerful API for building custom search for your site.

Umbraco search is built on top of the popular Lucene library.

Lucene

Lucene is an open source Java library for indexing and searching.

Lucene.NET is a .NET port of the Lucene library.

Lucene and Lucene.NET are projects of the Apache Foundation.

http://lucene.apache.org

http://lucenenet.apache.org

Lucene

Key Features

- Full-text indexing
- Relevancy searching
- Sorting search results (by field value or relevance)
- Boosting documents and fields
- Stemming analysis
- Synonyms

Lucene

How Lucene computes relevance

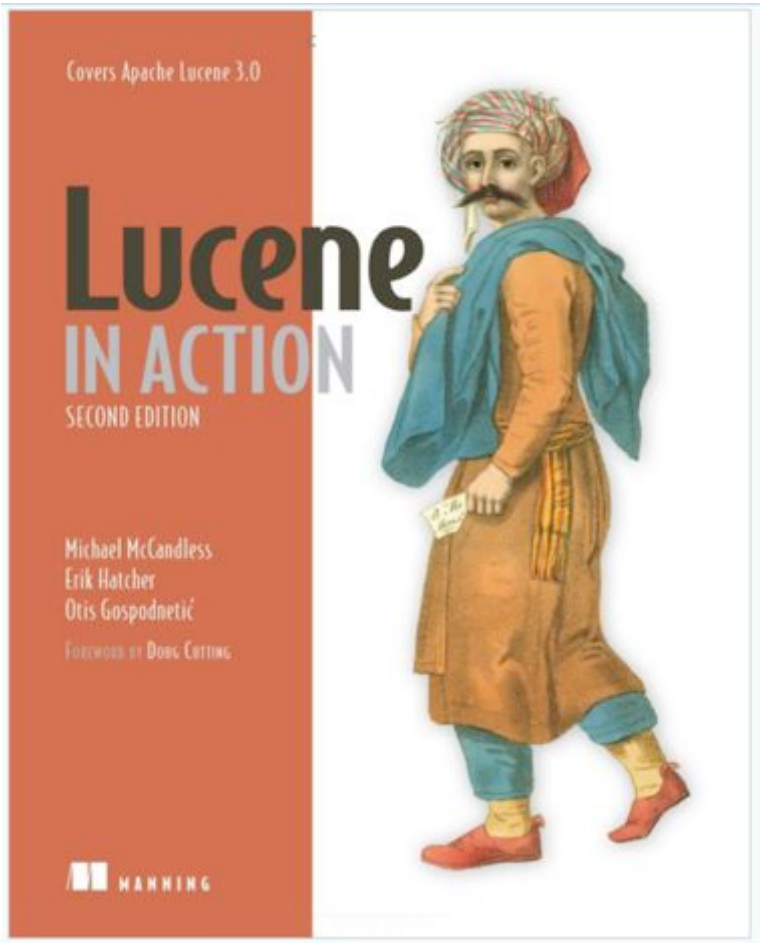
Lucene computes a score of how similar a document in the index matches the query.

$$\sum_{t \text{ in } q} (tf(t \text{ in } d) \times idf(t^2) \times boost(t.field \text{ in } d) \times lengthNorm(t.field \text{ in } d)) \times coord(q,d) \times queryNorm(q)$$

Some Factors:

- How many times the term occurs in the document.
- How 'unique' the term is.
- Fields and documents can be boosted.
- Shorter fields get a bigger boost.

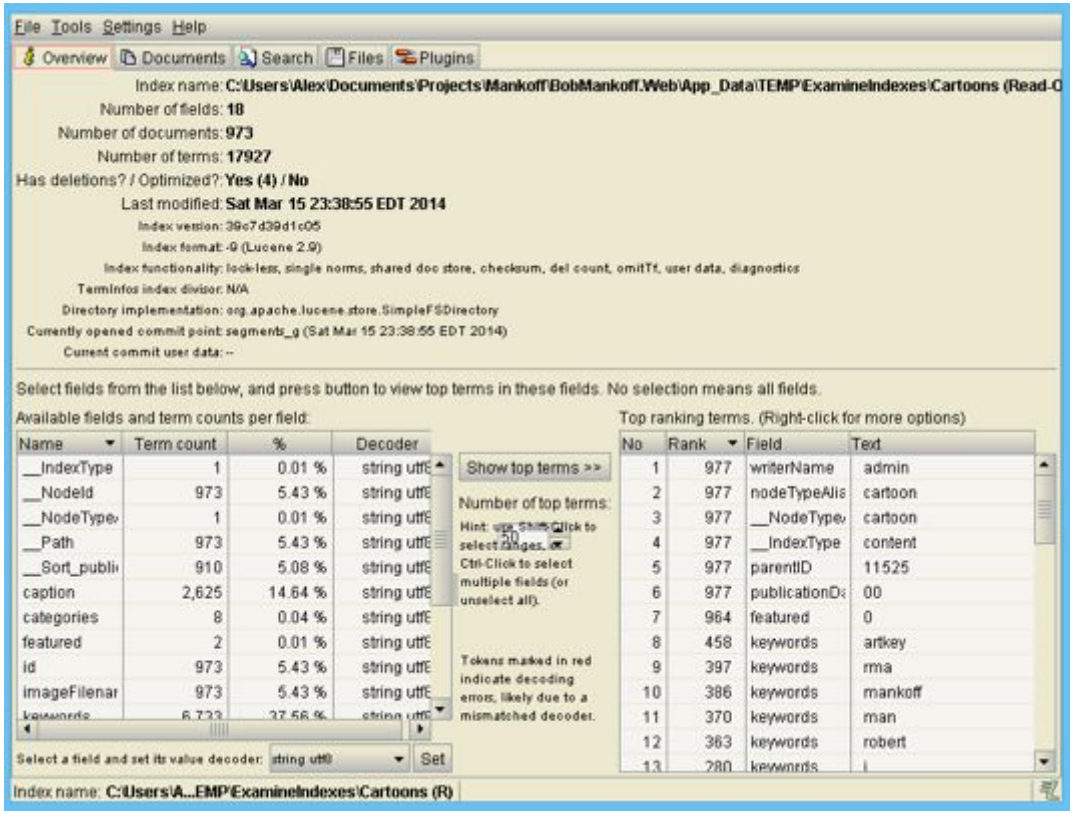
Lucene



Lucene in Action, 2nd edition
by Michael McCandless, Erik Hatcher and Otis Gospodnitic (Manning 2010)

Luke

Lucene diagnostic tool



http://code.google.com/p/luke/

Umbraco Examine

Umbraco Examine is an API for searching that uses **Lucene.NET**.

Umbraco Examine is an implementation of an **Examine** provider.

Examine is a separate project that provides a **Fluent API** for using **Lucene.NET**.

https://github.com/Shandem/Examine

Note: having Examine and Lucene.NET built into Umbraco makes it much easier to manage compared to Solr or other search index services that are commonly used for search applications.

Use Cases

Site Search

Custom Data Search

Site Search

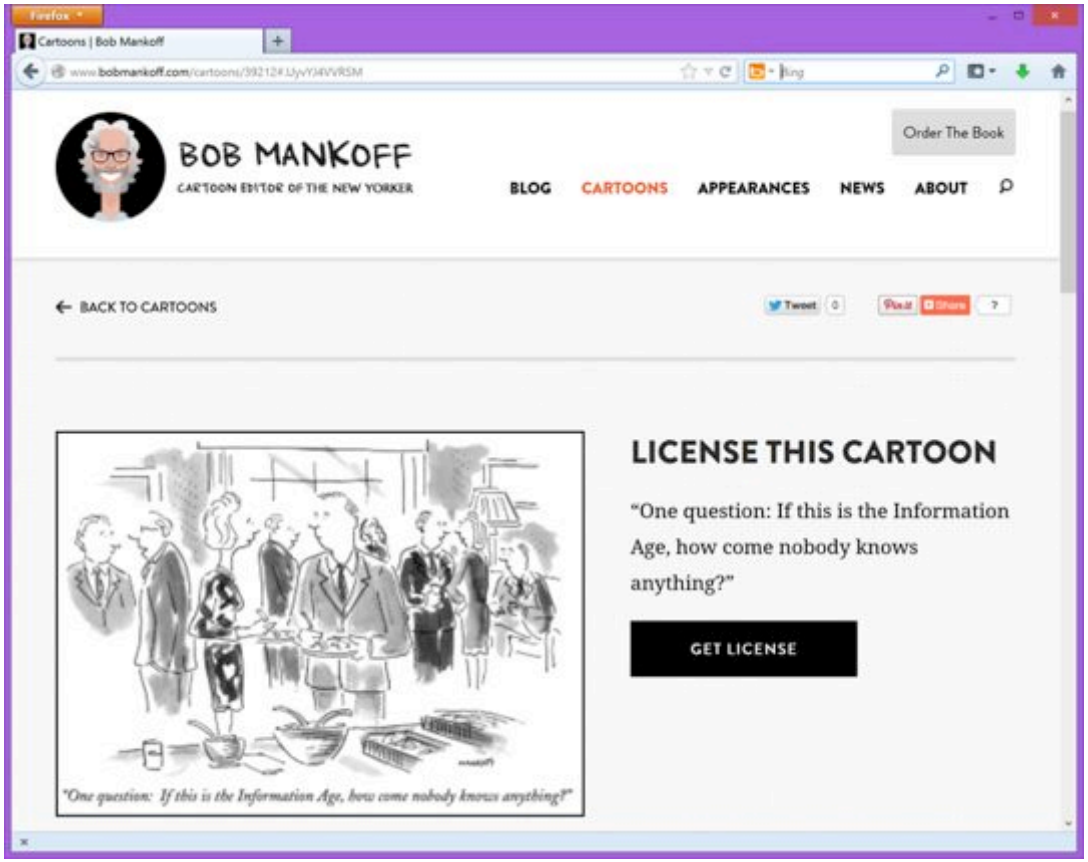
Challenge: Our sites that have complex pages built with blocks of content that are pulled from various nodes.

Custom Data Search

Our sites often have the requirement to search specific sets of data.
Examples: blog search, news and events search.

Cartoon Search Example

BobMankoff.com



Cartoon Search Example

BobMankoff.com

Examine Settings

ExamineIndex.config

```
<?xml version="1.0"?>
<ExamineLuceneIndexSets>
  <!-- The internal index set used by Umbraco back-office - DO NOT REMOVE -->
  <IndexSet SetName="InternalIndexSet" IndexPath="~/App_Data/TEMP/ExamineIndexes/Internal/"/>

  <!-- The internal index set used by Umbraco back-office for indexing members - DO NOT REMOVE -->
  <IndexSet SetName="InternalMemberIndexSet" IndexPath="~/App_Data/TEMP/ExamineIndexes/InternalMember/">
    <IndexAttributeFields>
      <add Name="id" />
      <add Name="nodeName"/>
      <add Name="updateDate" />
      <add Name="writerName" />
      <add Name="loginName" />
      <add Name="email" />
      <add Name="nodeTypeAlias" />
    </IndexAttributeFields>
  </IndexSet>

  <!-- Default Indexset for external searches, this indexes all fields on all types of nodes-->
  <IndexSet SetName="ExternalIndexSet" IndexPath="~/App_Data/TEMP/ExamineIndexes/External/" />
</ExamineLuceneIndexSets>
```

Examine Settings

ExamineSettings.config

```
<?xml version="1.0"?>
<Examine>
  <ExamineIndexProviders>

  ...

  </ExamineIndexProviders>

  <ExamineSearchProviders>

  ...

  </ExamineSearchProviders>
</Examine>
```

Examine Settings

`ExamineSettings.config` (Index Providers)

```
<ExamineIndexProviders>
  <providers>
    <add name="InternalIndexer" type="UmbracoExamine.UmbracoContentIndexer, UmbracoExamine"
         supportUnpublished="true"
         supportProtected="true"
         analyzer="Lucene.Net.Analysis.WhitespaceAnalyzer, Lucene.Net"/>

    <add name="InternalMemberIndexer" type="UmbracoExamine.UmbracoMemberIndexer, UmbracoExamine"
         supportUnpublished="true"
         supportProtected="true"
         analyzer="Lucene.Net.Analysis.Standard.StandardAnalyzer, Lucene.Net"/>

      <!-- default external indexer, which excludes protected and unpublished pages-->
      <add name="ExternalIndexer" type="UmbracoExamine.UmbracoContentIndexer, UmbracoExamine"/>

  </providers>
</ExamineIndexProviders>
```

Examine Settings

`ExamineSettings.config` (Search Providers)

```
<ExamineSearchProviders defaultProvider="ExternalSearcher">
  <providers>
    <add name="InternalSearcher" type="UmbracoExamine.UmbracoExamineSearcher, UmbracoExamine"
         analyzer="Lucene.Net.Analysis.WhitespaceAnalyzer, Lucene.Net"/>

    <add name="ExternalSearcher" type="UmbracoExamine.UmbracoExamineSearcher, UmbracoExamine" />

    <add name="InternalMemberSearcher" type="UmbracoExamine.UmbracoExamineSearcher, UmbracoExamine"
         analyzer="Lucene.Net.Analysis.Standard.StandardAnalyzer, Lucene.Net" enableLeadingWildcards="true"/>

  </providers>
</ExamineSearchProviders>
```

Examine Settings

Documentation
http://our.umbraco.org/documentation/Reference/Searching/Examine/full-configuration

Examine Settings

Analyzers

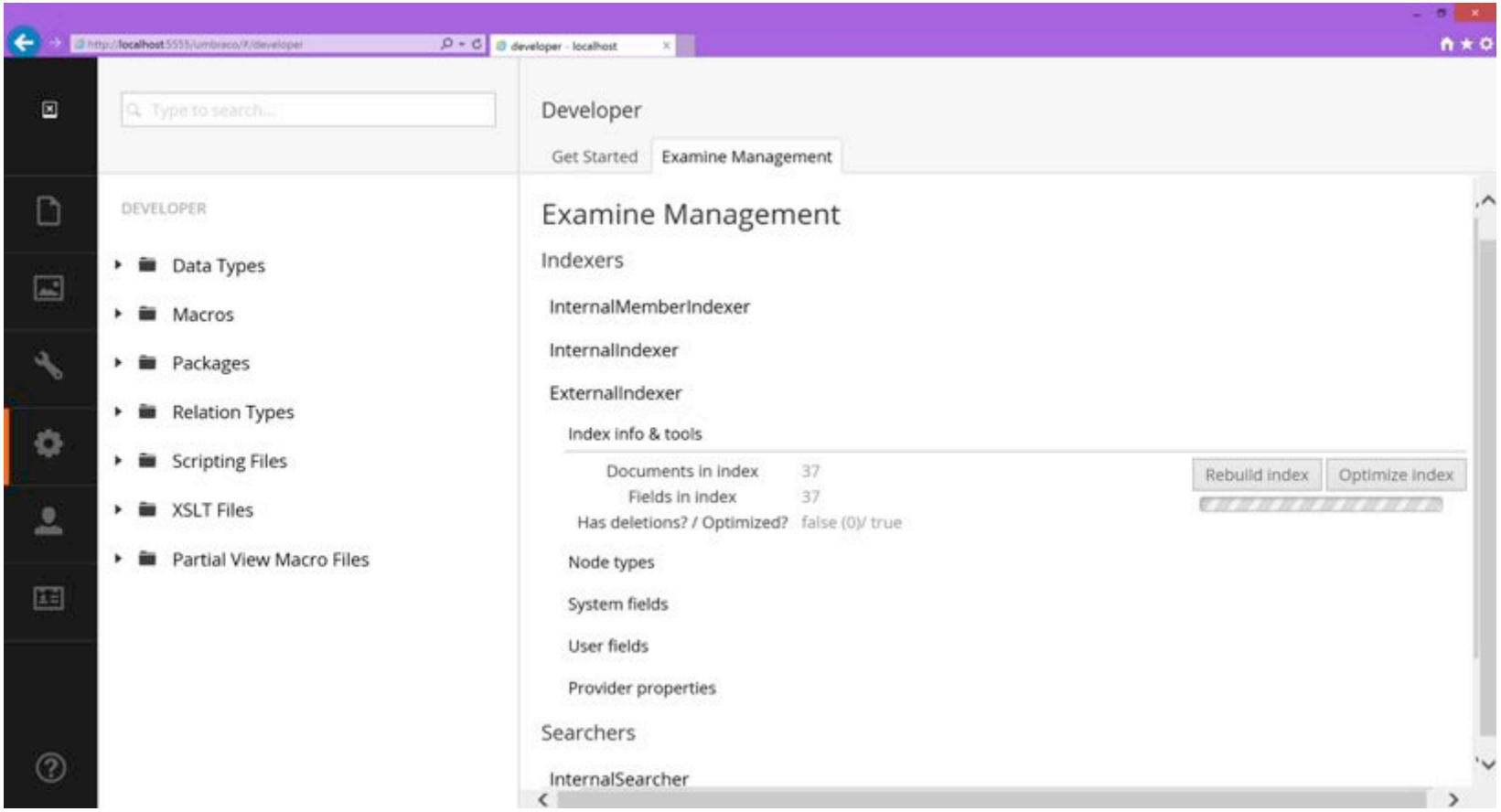
"An analyzer tokenizes text by performing any number of operations on it, which could include extracting words, discarding punctuation, removing common words, reducing words to a root form (stemming), or changing words into the basic form" (from chapter 4 of *Lucene In Action, 2nd edition*).

Common Analyzers:

- Lucene.Net.Analysis.WhitespaceAnalyzer
- Lucene.Net.Analysis.Standard.StandardAnalyzer

You must use the same analyzer for both indexing and searching.

Managing Indexes



Cartoon Search Example

Queries using Examine API

Cartoon Search Requirements

- Keyword search on both captions and keywords fields.
- Filter by category
- ‘all’ category returns all cartoons
- Sortable by relevance and publication date

Cartoon Search Example

ExamineIndex.config (IndexSet)

```
<IndexSet SetName="CartoonIndexSet" IndexPath="~/App_Data/TEMP/ExamineIndexes/Cartoons/">
  <IndexAttributeFields>
    <add Name="id" />
    <add Name="nodeName" />
    <add Name="updateDate" />
    <add Name="writerName" />
    <add Name="path" />
    <add Name="nodeTypeAlias" />
    <add Name="parentID" />
  </IndexAttributeFields>
  <IndexUserFields>
    <add Name="caption"/>
    <add Name="publicationDate" EnableSorting="true"/>
    <add Name="keywords"/>
    <add Name="categories"/>
    <add Name="featured"/>
  </IndexUserFields>
  <IncludeNodeTypes>
    <add Name="Cartoon"/>
  </IncludeNodeTypes>
  <ExcludeNodeTypes>
  </ExcludeNodeTypes>
</IndexSet>
```

Cartoon Search Example

`ExamineSettings.config` (Index Providers)

```
<add name="CartoonIndexer" type="UmbracoExamine.UmbracoContentIndexer, UmbracoExamine"
     indexSet="CartoonIndexSet"
     supportUnpublished="false"
     supportProtected="false"
     analyzer="Lucene.Net.Analysis.Standard.StandardAnalyzer, Lucene.Net" />
```

`ExamineSettings.config` (Search Providers)

```
<add name="CartoonSearcher" type="UmbracoExamine.UmbracoExamineSearcher, UmbracoExamine"
     analyzer="Lucene.Net.Analysis.Standard.StandardAnalyzer, Lucene.Net" enableLeadingWildcards="true"/>
```

Note: For comma-separated fields (such as Checkbox List), use the StandardAnalyzer – not the WhitespaceAnalyzer.

Cartoon Search Example

```
var searcher = ExamineManager
               .Instance
               .SearchProviderCollection["CartoonSearcher"];

var searchCriteria = searcher.CreateSearchCriteria(
               UmbracoExamine.IndexTypes.Content
);

var query = searchCriteria.RawQuery("nodeTypeAlias:cartoon");

var results = searcher.Search(query);
```

Cartoon Search Example

```
var results = searcher.Search(query);
```

Note: results are of type `Examine.ISearchResults`

`results.Count()` to get number of results

For paging:

```
var searchResults = results
                        .Skip(((pageNumber - 1) * pageSize))
                        .Take(pageSize);
```

Lucene Query (for debugging):

```
string luceneQuery = searchCriteria.ToString();
```

Cartoon Search Example

```
if (category != "All")
{
    query = criteria.Field("categories", category).Compile();
}

query = criteria.Field("keywords", keywords).Compile();

query.OrderBy(new string[] { "publicationDate" });
```

Note: searchResults is of type IEnumerable<SearchResult>

```
foreach (var item in searchResults)
{
     string cats = item.Fields["categories"];
}
```

Cartoon Search Example

Custom Index Field for Keyword search

```
public class ExamineEventHandlers : ApplicationEventHandler
{
    protected override void ApplicationStarted(UmbracoApplicationBase app,
        ApplicationContext applicationContext)
    {
        ExamineManager.Instance.IndexProviderCollection["Cartoons"].GatheringNodeData
            += ExamineEvent_GatheringNodeData;
    }

    void ExamineEvent_GatheringNodeData(object sender, IndexingNodeDataEventArgs e)
    {
        e.Fields.Add("combined", e.Fields["keywords"] +
                " " + e.Fields["caption"]);
    }
}
```

TypedSearch()

Instead of using the Examine API directly, UmbracoHelper has a method that returns Umbraco typed items.

```
UmbracoHelper helper = new UmbracoHelper(UmbracoContext);

IEnumerable<IPublishedContent> results
                                = helper.TypedSearch(query);
```

Site Search Example

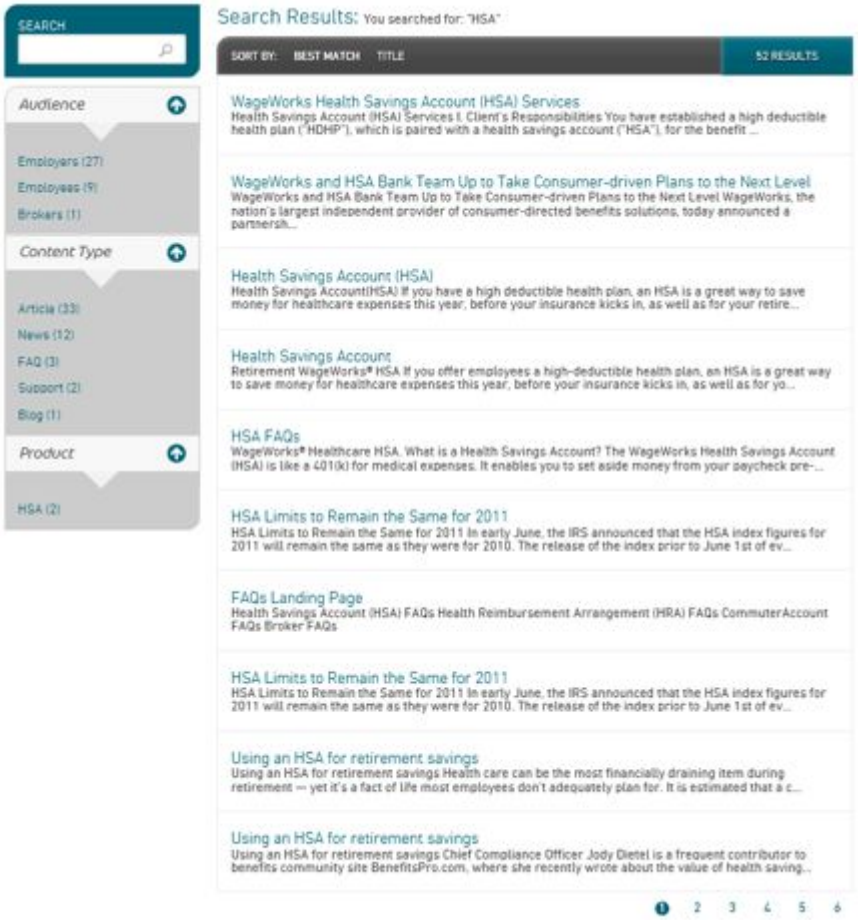
WageWorks.com

Site Search Example

WageWorks.com

Challenging Requirements:

- Composite pages (composed of node components)
- Faceted Search results



Site Search Example

Custom Index Field for Composite pages search

```
public class ExamineEventHandlers : ApplicationEventHandler
{
    protected override void ApplicationStarted(UmbracoApplicationBase app,
        ApplicationContext applicationContext)
    {
        ExamineManager.Instance.IndexProviderCollection[“SiteSearch"].GatheringNodeData
            += ExamineEvent_GatheringNodeData;
    }

    void ExamineEvent_GatheringNodeData(object sender, IndexingNodeDataEventArgs e)
    {
          // Create a StringBuilder to store indexed content
          // For each Document Type, add the doc types fields to be indexed
          // Get children nodes (page components) and add searchable fields
          // Set index field to StringBuilder string
    }
}
```

Site Search Example

Custom Index Field for Composite pages search

```
public class ExamineEventHandlers : ApplicationEventHandler
{
    protected override void ApplicationStarted(UmbracoApplicationBase app,
        ApplicationContext applicationContext)
    {
        ExamineManager.Instance.IndexProviderCollection[“SiteSearch"].GatheringNodeData
            += ExamineEvent_GatheringNodeData;
    }

    void ExamineEvent_GatheringNodeData(object sender, IndexingNodeDataEventArgs e)
    {
          // A better way... ?
          //  RenderTemplate()
          //  Load page from web request
    }
}
```

Faceted Search

Faceted search gives the user immediate feedback so they can make good filtering decisions.

Faceted Search

Bobo-Browse

Original (Java) project powers LinkedIn search

http://code.google.com/p/bobo-browse/

BoboBrowse.Net is a .Net port

http://bobo.codeplex.com/

Cogworks blog post on using BoboBrowse.Net for faceted search in Umbraco: http://thecogworks.co.uk/blog/posts/2013/january/examiness-hints-and-tips-from-the-trenches-part-6/

There is experimental support for facets in the 'Facet' branch of Examine on Github.

ezSearch

ezSearch is the easiest way to add search to an Umbraco site.

Available from the Umbraco package repository.

https://github.com/mattbrailsford/ezsearch

Basic steps:

- Install the package
- Create a search results page
- Add the macro to the search results page
- Wire up the global search form

Works well for simple sites but only searches all the nodes.

Resources

- Umbraco Examine documentation
  http://our.umbraco.org/documentation/Reference/Searching/Examine/
- Examine documentation
  https://github.com/Shandem/Examine/wiki
- Examining Examine blog post
  http://umbraco.com/follow-us/blog-archive/2011/9/16/examining-examine.aspx