Adding Product Search

Search is an essential part of any ecommerce application. If you don’t believe me, got to [https://amazon.com](https://amazon.com/), how hard is it to find the search box, pretty easy right.

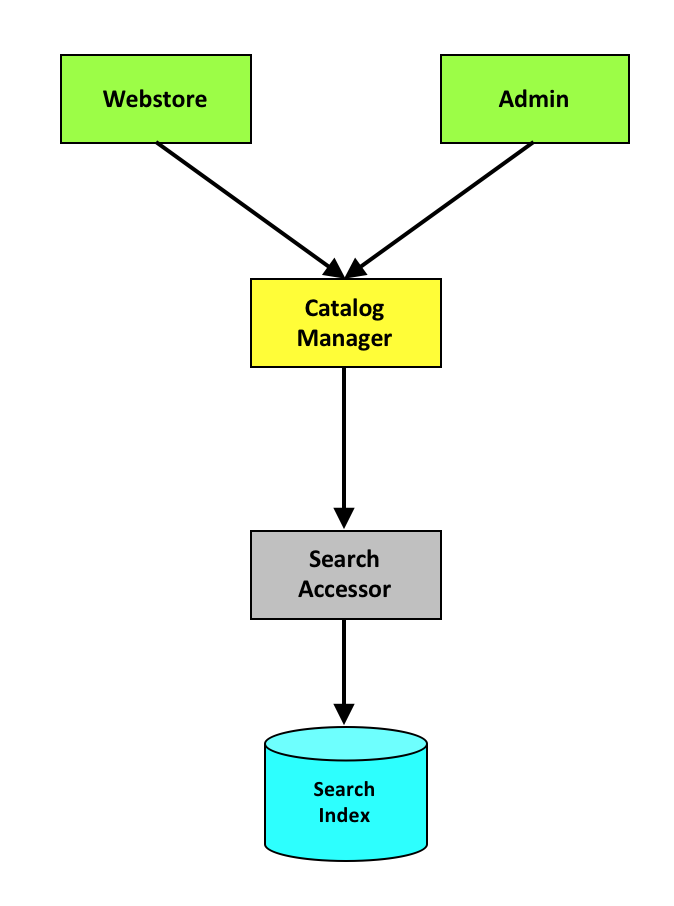
Well if search is essential, we probably need to make a pretty good search right? Well our current ecommerce application supports search, but not a very good search. The current reference implementation does a very simple like query against the database to find products that match the query text. While this might look like a working search, it won’t do even the basic things we would expect from a search, like finding misspelled words.

How should we go about adding search to our application? We could use something like full text search in the database. That might work for very basic use cases, but there is better options out there. For this activity we are going to use Lucene.net to do the searches. Lucene.net is an open source search library based off of a java version called Lucene.

At DPL we have used Lucene on a few projects. The great thing about Lucene is that it is pretty light weight for a search solution. No services to setup. No web server to setup. No Java to install.

But before we jump into code lets discuss the change we are going to make to the system. We want to add search to our system. Our current architecture doesn't have search, we will need to extend our architecture to support search.

Next it would make sense to diagram out what we would expect the new architecture to be.



In this activity we are not going to just use database like queries to do search. We are going to use lucene. Lucene is an awesome library for doing searches of information. We have often used it for searching documents, but you can use it to search random data too.

Where do we start? At the beginning of course.

We will add a nuget package reference to Lucene.net to our accessors project. Then we will rewrite our search accessor to use Lucene.net. (code snippets below assume Lucene 4.8)

But this might leave us wondering? How do we use lucene? Well luckily Lucene is pretty easy to use.

First we will add some configuration as to where this index should be stored. We will configure this as an environment variable, just like the database connection string. This code is already in place in our reference application, just use the Config.IndexPath variable.

<https://gist.github.com/chadmichel/46e4e540f0aae621668775e7e0c52a8b>

After we have some configuration out of the way we can build our search index. The code to build a search index is pretty straight forward.

<https://gist.github.com/chadmichel/c425820b8479d7bf222f775a31cdebc9>

Next we will have to write the actual code to do the searching. This code should be pretty straight forward too.

<https://gist.github.com/chadmichel/230594873a2948bdd5d42359a2e1c91c>

Once we have those in place we just need to add some unit tests for using Lucene.net, and we should be good to go from a lucene standpoint.

Next we would need to update our manager level to support calling our new search accessor. We will need to add a new method to our CatalogManager to perform this search. We should also add a few unit tests for this code.

Last we need to update our webstore UI to support they lucene based searched.

This activity is a great example of how our architecture could be extended with minimal effort. We were able to extend the system by only changing one existing architecture piece (CatalogManager). And the change to CatalogManger was only an extension.