Searcher.cfc 1.0 Beta

Posted At : June 14, 2007 2:21 PM | Posted By : Steve
Related Categories: ColdFusion, com.sebtools

When I first started developing public web sites, I noticed that I often needed the same functionality on several sites. It was fairly easy to package those up as programs that I could copy from site to site. What I discovered, however, was that each one required a bit of set up.

I didn't like that I had to create (or copy) the required tables and set up other prerequisites. At the time, I was using Fusebox 3, so I also had to create circuits. All told, it always took me more time that I wanted to set up a program - and that was boring work.

I decided that I wanted my programs to be somewhat self-installing. **DataMgr** was one step toward that vision (which you can see actualized to a large degree in **CodeCop**). Now I can use DataMgr to install the tables that I need.

Searcher.cfc expands on that vision by using the same principal for verity collections. It does use cfcollection, so if that isn't available to you it won't do much good.

With Searcher.cfc, you can use the indexPath or indexQuery methods to index a query. If the collection doesn't exist, Searcher.cfc will automatically create it for you.

Additionally, Searcher.cfc will keep track of searches performed on your site.

In order to use Searcher.cfc, you must first initialize it with the init() method, which has the following arguments:

- **CollectionPath** (required): The full file path of the folder in which you want Searcher.cfc to create collections.
- **DataMgr** (required): An instantiated DataMgr component.
- **sendpage** (required): The browser path to the page that will redirect the user to the results of their search (more on this later).
- **excludedirs** (optional): a list of folders to exclude from search results.
- **excludefiles** (optional): a list of files to exclude from search results.

Searcher.cfc uses two tables to track searches: srchSearches and srchSelections, which it creates via DataMgr.

The "sendpage" argument must be the browser path to a file with the following code (assuming you have Searcher.cfc instantiated into Application.Searcher):

```
<cfif isDefined("url.searchid") AND isDefined("url.to")>
  <cfset Application.Searcher.send(url.searchid,url.to)>
</cfif>
```

This is what allows Searcher.cfc to see what pages a users selects from a search.

In order to index a search against a file path, use the indexPath() method, with the following arguments:

- **CollectionName** (required): The name of the collection that you are indexing.
- **Key** (required): The full file path to the folder that you want to index.

In order to index a recordset, use the indexQuery() method, with the following arguments:

- **CollectionName** (required): The name of the collection that you are indexing.
- **query** (required): The query you are indexing (pass in the actual recordset, not just the name of the query) .
- **Key** (required): The name of the key field from the recordset.
- **Title** (required): The name of the field holding the title you want to display in your search results.
- **Body** (required): The name (or list of names) of the field(s) holding the content you want to search.

- **URLPath** (optional): The URL Path of the page that will result from the search (just like in verity, this will have the value of the key appended to it).
- **Custom1** (optional): The field for a Custom1 value, if you want one.
- **Custom2** (optional): The field for a Custom2 value, if you want one.

If you are calling the indexing methods from within a CFC and want to make sure that they are indexed regularly, I would suggest using **Scheduler.cfc**.

In order to run a search and get back a recordset of results, use the run() method with the following arguments:

- **searchterm** (required): The word for which the user is searching.
- **collections** (optional): The collections to search (defaults to searching all collections that Searcher.cfc is aware of).

This will log the search and return a recordset of the results. If you want to run the same search again without logging the search (for subsequent pages of a search, for example), use the reRun() method with the following argument:

- **searchid** (required): The id of search you want to rerun ("SearchID" will be a column in the query returned from run() and reRun().

If you want to get search data, just call the getSearchData() method with the following arguments:

- **startdate** (optional): The date from which the report should start.
- **enddate** (optional): The date at which the report should end.

To get details about the landing pages chosen for a search phrase, you the getLandingPages() method with the followig arguments:

- **searchterm** (required): The phrase for which you want details.
- **startdate** (optional): The date from which the report should start.
- **enddate** (optional): The date at which the report should end.

I have been using Searcher.cfc for a few years and I have found that it has made the creation and deployment for Verity searches relatively quick and easy.

Searcher.cfc is open source and free for any use.