Item Buckets for Sitecore CMS 6.4-6.6

Administrator’s and Developer’s Guide

A Quick Start Guide and Configuration Reference for Administrators and Developers

Table of Contents

[Chapter 1 Introduction 4](#_Toc331495009)

[1.1 Fundamental Concepts 5](#_Toc331495010)

[1.1.1 Bucket 5](#_Toc331495011)

[1.1.2 Hidden Items (Setup) 6](#_Toc331495012)

[1.1.3 Why Bucket? 6](#_Toc331495013)

[Chapter 2 Installation and Configuration Guide 7](#_Toc331495014)

[2.1 Installation 8](#_Toc331495015)

[2.1.1 Requirement 8](#_Toc331495016)

[2.1.2 Installing the Package 8](#_Toc331495017)

[2.1.3 Installation Guide 8](#_Toc331495018)

[2.2 Setup 10](#_Toc331495019)

[2.2.1 Creating a Bucket 10](#_Toc331495020)

[2.2.2 Defining a Bucket Template 11](#_Toc331495021)

[2.2.3 Defining a Facet (Category) 11](#_Toc331495022)

[2.2.4 Defining a Search Result Image and Search Result Text 12](#_Toc331495023)

[2.2.5 Display Media Library Images in Search Results. 13](#_Toc331495024)

[2.2.6 Creating an Item in a Bucket 13](#_Toc331495025)

[2.2.7 Re-Syncing a Bucket 14](#_Toc331495026)

[2.2.8 Locking Child Relationship 14](#_Toc331495028)

[2.2.9 Managing your buckets through the Control Panel 14](#_Toc331495029)

[2.2.10 Item Bucket Search Settings 14](#_Toc331495030)

[2.2.11 Link Database 15](#_Toc331495031)

[Chapter 3 Searching 16](#_Toc331495032)

[3.1 Introduction to the new Search Interface 17](#_Toc331495033)

[3.1.1 Simple Search 17](#_Toc331495034)

[3.1.2 Enabling new Views 18](#_Toc331495035)

[3.1.3 Using the Facets (Category Filters) 19](#_Toc331495036)

[3.1.4 Language Search 20](#_Toc331495037)

[3.1.5 Complex Searches 20](#_Toc331495038)

[3.1.6 Range Searches 20](#_Toc331495039)

[3.1.7 Combined, AND, OR, NOT 21](#_Toc331495040)

[3.1.8 Using the Tips 21](#_Toc331495041)

[3.1.9 Launching Search Results 21](#_Toc331495042)

[3.1.10 Using Search Filters 23](#_Toc331495043)

[3.1.11 Auto-Organising 26](#_Toc331495044)

[3.1.12 Paging Results 26](#_Toc331495045)

[3.1.13 Search Box Clear 27](#_Toc331495046)

[3.1.14 Search Box Helpers 27](#_Toc331495047)

[3.1.15 Locking Check 29](#_Toc331495048)

[3.1.16 Securing Bucket and UnBucket Rights 29](#_Toc331495049)

[3.1.17 Restoring a Bucket back to a structure before it was bucketed. 29](#_Toc331495050)

[3.1.18 IA Modifications 30](#_Toc331495051)

[3.1.19 Keyboard Shortcuts 30](#_Toc331495052)

[3.1.20 Context Menu 30](#_Toc331495053)

[3.1.21 Query the index from any inbuilt field supporting Lists 31](#_Toc331495054)

[3.1.22 Using the new Field Types 31](#_Toc331495055)

[3.1.23 Using Item Buckets with the datasource of a Control 33](#_Toc331495056)

[3.2 Page Editor and DMS Support 37](#_Toc331495057)

[3.2.1 Personalisation and MV Tests 37](#_Toc331495058)

[3.2.2 Setting the Datasource 37](#_Toc331495059)

[3.3 Inserting Links and Link Management 39](#_Toc331495060)

[3.3.1 Inserting Links 39](#_Toc331495061)

[3.3.2 Link Providers 41](#_Toc331495062)

[3.3.3 Item Resolvers 41](#_Toc331495063)

[3.3.4 Support tags across any item 41](#_Toc331495064)

[Chapter 4 Developing with Item Buckets 43](#_Toc331495065)

[4.1 Buckets API 44](#_Toc331495066)

[4.2 Working with new code 48](#_Toc331495067)

[4.3 Working with Extension Points 49](#_Toc331495068)

[4.3.1 Working with Existing Code 50](#_Toc331495069)

[4.3.2 Common Requirements 53](#_Toc331495070)

[4.3.3 Creating a Tag Repository 53](#_Toc331495071)

[4.3.4 Creating a new Facet 55](#_Toc331495072)

[4.3.5 Configuration Files 56](#_Toc331495073)

[4.4 Multiple Index Support 59](#_Toc331495074)

[4.4.1 Adding a new View 60](#_Toc331495075)

[Chapter 5 Sitecore Items with Big Data 62](#_Toc331495076)

[5.1 Explanation of the configuration files 63](#_Toc331495077)

[5.1.1 In Memory Index 63](#_Toc331495078)

[5.1.2 Remote Index 63](#_Toc331495079)

[5.1.3 New Crawlers 64](#_Toc331495080)

[5.1.4 Query Server 65](#_Toc331495081)

[5.1.5 Extending Support with SOLR 66](#_Toc331495082)

[5.1.6 Installing SOLR on Windows 66](#_Toc331495083)

[5.1.7 Replication of Index Across Servers 70](#_Toc331495084)

[5.1.8 Master Server Configuration 70](#_Toc331495085)

[5.1.9 Slave Server Configuration 70](#_Toc331495086)

[5.1.10 Repeater Server Configuration 70](#_Toc331495087)

[5.1.11 Extending Support with Hadoop Clusters 71](#_Toc331495088)

[Chapter 6 Appendix 72](#_Toc331495089)

[6.1 Tips and Tricks 73](#_Toc331495090)

[6.2 Known Issues 76](#_Toc331495091)

# Introduction

This document is designed for Sitecore administrators and developers. It contains information about how to set up, configure and tune the module.

The document contains the following chapters:

* Chapter 1 — Introduction

The introduction and the fundamental concepts of the module.

* Chapter 2 — Installation and Configuration Guide.

The steps required for the quick setup of Item Buckets. The quick setup is the minimal amount of configuration that would let you use the module.

* Chapter 3 — Searching.

Practical advice on configuring and using the module from an administrator’s perspective.

* Chapter 4 — Developing with Item Buckets.

The descriptions of the module’s configuration settings and how to use the API

* Chapter 5 — Scaling with Item Buckets

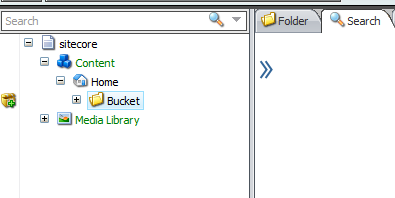
How to use scaling features with Item Buckets

## Fundamental Concepts

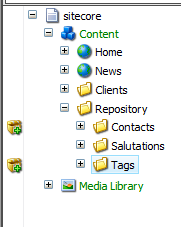
This section contains a list of definitions for the terms and concepts used throughout the Item Bucket documentation.

### Bucket

A bucket is a container within the Content Tree that will store content items. What differs this from a normal container is that all items stored within the container will be hidden and have a new Content Search tab to be able to look up the items within the container instead of using the tree to navigate and select items. In addition to this, items in a bucket, will auto organise your content into folders and completely remove the parent to child relationship between items.



All buckets will be marked with a Gutter Warning to show authors which containers are buckets and which are normal containers. It is up to the individual user to enable this gutter warning by right-clicking to the left of the content tree and selecting “Bucket Container”



Having an item as a bucket brings many advantages including:

* You now can search for content under it (even “non-bucketable” items)
* You can now use the new Bucket API with those items
* All items are now auto-organised for you into logical format
* You can now have items living below other items that act as embedded items
* You have a repository that can store millions of items and not hinder the UI.

### Hidden Items (Setup)

The items contained within a Bucket will **NOT** be hidden in the content tree by default (They will however have their hidden checkbox selected). You will need to navigate to the **View Tab** and uncheck the Hidden Items Checkbox to not show the hidden items. This is recommended if you are using Item Buckets as it will free up a lot of the loading of items in the content tree that is now unnecessary. Developers can work with hidden items on if they need to.

### Why Bucket?

The Item Buckets addresses the management of large amounts of items within the content tree and being able to retrieve and work with them in a speedy and efficient manner. To decide if you should turn an item into a bucket, and in-turn, hiding all its descendants, is as simple as asking yourself if you care about the structure of the data that lives under the bucket. For example, if you had a Product Repository, Movie Repository or Tag Repository within the content tree, you would most likely want to just dump them all into a folder and when you want to work with a particular product, movie or tag, you would simply search for it and open it.

Item Buckets allows for connections to be made through semantics, not hierarchy. For example, traditionally for creating products you would put categories into the content tree and then place product items under those categories. With item buckets, you can place all products in the one repository but tag each product with what category they belong to.

# Installation and Configuration Guide

This chapter describes how to quickly set up the Item Buckets. The quick setup, is the minimal amount of configuration that would let you use the module.

This chapter contains the following sections:

* Installation
* Setup
* Configuration

## Installation

This section describes the installation process. It is worth noting that with the Item Buckets Module installed, it doesn’t mean that one must use the module. Only when a user starts creating buckets, will the buckets system be in place. You can also run buckets alongside normally structured items. A hybrid of buckets and hierarchically structured items can be used. This may be useful if you want to define some sub-structure within a bucket.

### Requirement

\*\*\*Ivan told that RendererURL will probably be configured automatically\*\*\*

The following are required to install this module.

* You must be running an application pool in the .net 4.0 runtime
* You must have at least 4 GB of ram but 8 Gb is recommended
* An SSD Hard Drive is recommended as Lucene.net is very I/O intensive

### Installing the Package

\*\*\*Ivan told that RendererURL will probably be configured automatically\*\*\*

To install Item Buckets you must first determine if this module is the right solution for you. Once installed, you will be giving content authors the ability to:

* Alter the parent to child relationships for items (possible causing your code to not work as desired if the code is already existent – see Developer Section for explanation)
* Hide Items from the content author’s and offer a new way to retrieve them
* Allow authors to set values of fields via search

### Installation Guide

There are two types of installation for the Sitecore Item Buckets. This is to address the different server environments that this module will be installed into.

1. **Sitecore.ItemBuckets as a Nuget Package**

Either through Visual Studio 2010 or from Nuget.Org, search for Sitecore Item Buckets. It will provide you with 3 links. 1 is for the Kernel and the other is for the UI. You will want to attach the Kernel to all projects that will be using the Item Buckets Search API. You will want to attach the UI package to any of your projects that require the Item Buckets Search UI e.g. your Sitecore Website Project. There is a 3rd optional package with is Sitecore Item Buckets Big Data. This can be attached to your projects if you plan on having millions of content items in the content tree. It also brings with it other features such as “in-memory” indexes and “remote-indexes”.

By installing with the Nuget Package, you will be notified through Visual Studio when there is an update. You will have the choice to accept the updates or continue on with the version you have.

1. **Sitecore ItemBuckets as a Sitecore Package**

You will get 3 packages for the Sitecore Item Buckets. The first is the Kernel. This needs to be installed in all environments. The second is the Sitecore Item Buckets UI. This will only need to be installed in environments that need the UI e.g. Authoring, Development. Once you have downloaded the modules as a Sitecore Package, simply install through the Installation Wizard within the Desktop. It will ask you to override some content items. Please select “Override” or “Merge” if you have already made customisations to the template that is being modified.

A 3rd optional package can be installed which is the Sitecore Item Buckets Big Data functionality. This can be installed alongside the Sitecore Item Buckets Kernel.

The installation will fire some post installation steps and will run a Smart Publish as well as rebuilding the new index that will be installed for you.

You will need to restart the client to see your changes.

**Warning:** After you have installed the module, it will need to rebuild/build indexes and hence it may take some time for your authoring environment to start before you can start using the module. If you for example run a search straight away and don’t see all your facets or results, this is simply because the index has not finished building. You can check your log files to see when the index rebuild has finished.

## Setup

\*\*\*will Eng Plans differentiate between target audiences?\*\*\*

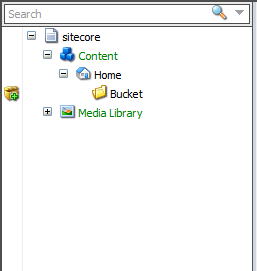
This section describes the how to setup Item Buckets

Note

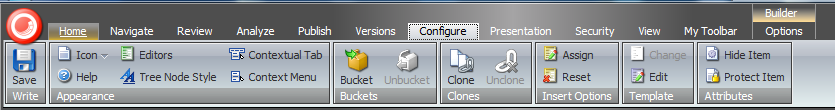
Before starting to work with the module, we recommend that you look through the appendix of tips and tricks as well as known issues at the end of this document.

### Creating a Bucket

To create a bucket you simply need to create any item within the content tree (e.g. a Folder) and give it a name.

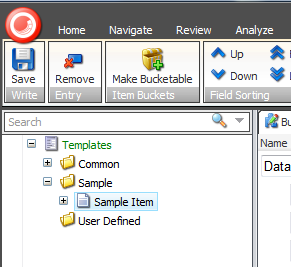


In the Ribbon, take a lock on the item in question and navigate to the Configure Tab and Click “**Bucket**”. You can create a bucket from a new item or an already existent item. Be aware that if you create a bucket from an already existent item, that it will organise and hide all the items below it. If you are turning a parent item into a bucket that has 1000’s or even millions of items, then it will take some time to organise them. A progress bar will show and show a running tally of the items being processed.

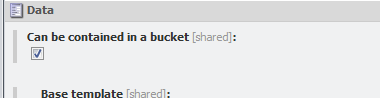


### Defining a Bucket Template

Another important part of setting up the system is that items need to be based off a template that is “allowed” to be hidden inside a bucket. If your item is based off a template that is not “allowed”, then the Bucket system will not automatically structure it for you. Instead, it will be act like a normal item within the content tree. To make your items “bucketable”, Open the **Template Manager** and select the types of templates that you would like to be able to go into Buckets (i.e. types of items that when created in this new folder will be hidden and searchable).

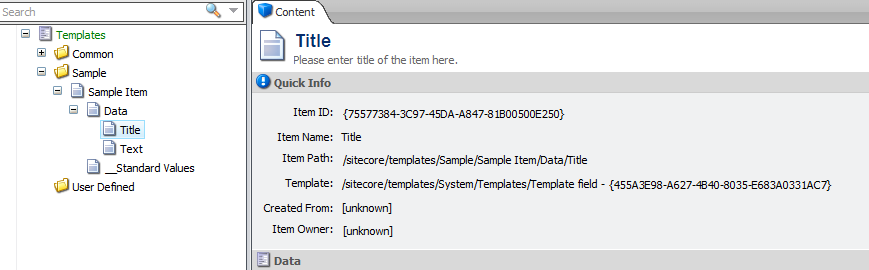


Select a Template and check the “Can be contained in a bucket” checkbox.



### Defining a Facet (Category)

If you would like to filter your search results based off categories, then you can mark your Template Fields as a Tag. To do this, expand the Field Section of your Template and click on the individual Template Field Items.



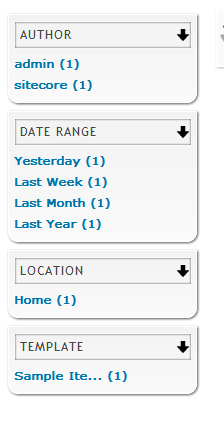
There is a checkbox field on this item called “Is Facet” (Make sure that Standard Fields is enabled). Simply check this if you would like to be able to filter your search results on this field. The following field types are supported for Facets:

**Single-Line-Text**

**Multi-Link-Text**

**Rich Text**

**Bucket Rich Text**

Once you have set this, your filter should show up in the facets when you run a search. 

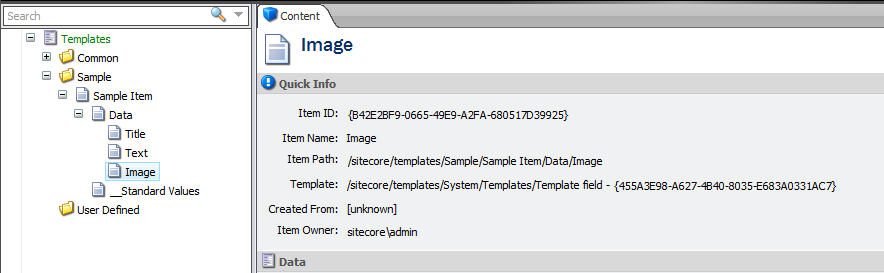
**NOTE:** If you find that the results that show in the facets differ from the results that show in the search results when you click the facet this could be for the following reasons

* Lucene has retrieved an item which you do not have read access to and therefore will not be shown in the results.
* If your search results returns e.g. 10 items and the Template facet shows 9 results then this is simply because there are other types of items in the results that you have not tracked as bucketable. To increase performance, Item Buckets will only facet on templates that are marked as bucketable.

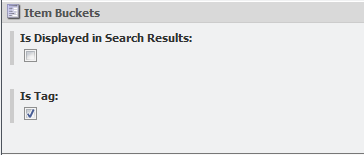
### Defining a Search Result Image and Search Result Text

When searching items, you are given the options of seeing a list view or a grid view of results. In both of these options Sitecore will return an image associated with the content. If you would like one our your image fields on your templates to be the image that is shown, this will guide you through how to set this up.

Navigate to your Image Field that you would like to make the search result image.



There is a checkbox field on this item called **“Is Displayed in Search Results”.** Simply check this if you would like to see this image shown in your results. You can also apply this to any other field that you want. For example, if you wanted your search results to show the **Title**, **Description** and **Body** field for the article template then you just need to mark those fields as “Is Displayed in Search Results” as well.



### Display Media Library Images in Search Results.

The entire content tree can work with buckets, including the Media Library. If you need to search for media items and want the images to show up in the search results, you will need to mark the “**Is Displayed in Search Results”** as “Checked” on the blob field that is part of the following 2 fields:

***/sitecore/templates/System/Media/Versioned/File/Media/Blob***

***/sitecore/templates/System/Media/Unversioned/File/Media/Blob***

### Creating an Item in a Bucket

In the same way that you would create an item in the content tree, simply right click on your bucket and insert an item. (You will want to make sure that you have marked in the template that this type of item can be “**Bucketable**”)

Your item will now be created however will be invisible within the content tree (if not, then you must turn Hidden Items off in the View Tab). If you created an item which is based off a template that is NOT bucketable then it will not be hidden. This allows you to run a hybrid of hierarchical and non-hierarchical items.

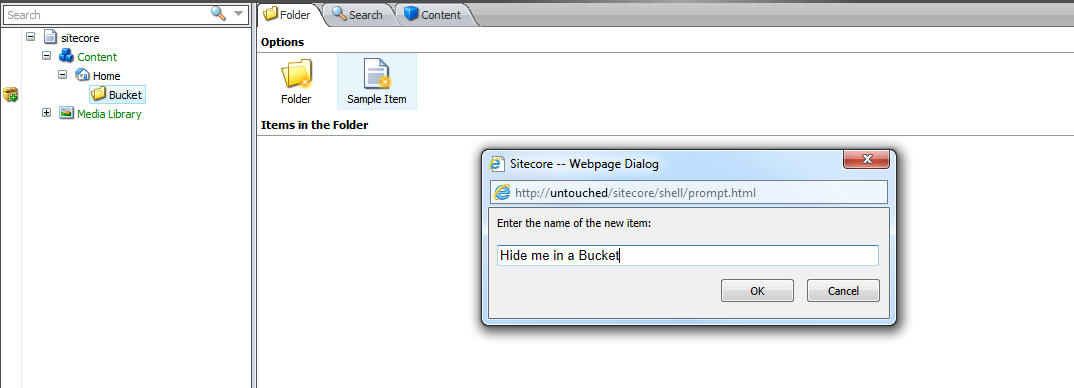
### Re-Syncing a Bucket

You won’t necessarily mark all your templates as “bucketable” straight away. If you want to make a template bucketable at a later stage, you can go back to your bucket items and run “Sync”. This will re-analyse which items should be hidden and automatically organized.

### 

### Locking Child Relationship

In some cases you will want to lock the relationship between parent and child. An example may be that you want to lock the parent to child relationship between articles and comments. If this is the case then you can mark on the standard values of a template to lock child relationships. In the example above, you would make “Lock child relationship” as checked on the Article standard values. These would mean that any item that is created below an item based off the article template would not auto categorize.



### Managing your buckets through the Control Panel

A new menu option is available in the Control Panel for you to be able to manage your buckets in the following ways:

* You can rebuild the bucket indexes
* You can rebuild the indexes on a remote machine
* You can see how many items are in your buckets and how many buckets exist

### Item Bucket Search Settings

You will have an item in the content tree that is for configuring the Item Buckets Settings

* You can change the keyboard shortcut for searching
* You can specify which Field in your content is the tag field
* You can specify the way that items open when selected from search results
* You can specify the number of items returned per page in a search

To set the field that is used for your Tags search, please set the “Tag Parent” field in the settings item to the item in the content tree that contains all your tags. You can then create a field on ANY template called “tags” and set the type to either multi-list or bucket-list.

### Link Database

The module will override your built in LinkDatabase so that it can scale to large amounts of items. If you have existing code that uses the LinkDatabase, then there is a new overload for the GetRefferes and GetReferences methods for passing in the page number. This is useful for when an item has many connections e.g. an Article template to all items based off it. This way, when trying to lookup links between items, this will only load 20 items into memory at a time, rather than every link.

\*\*\*is there anything that you must configure about the engagement plan?\*\*\*

# Searching

This chapter describes how to Search with Item Buckets

\*\*\*consider changing the title of the chapter or breaking it into three chapters\*\*\*

This chapter contains the following sections:

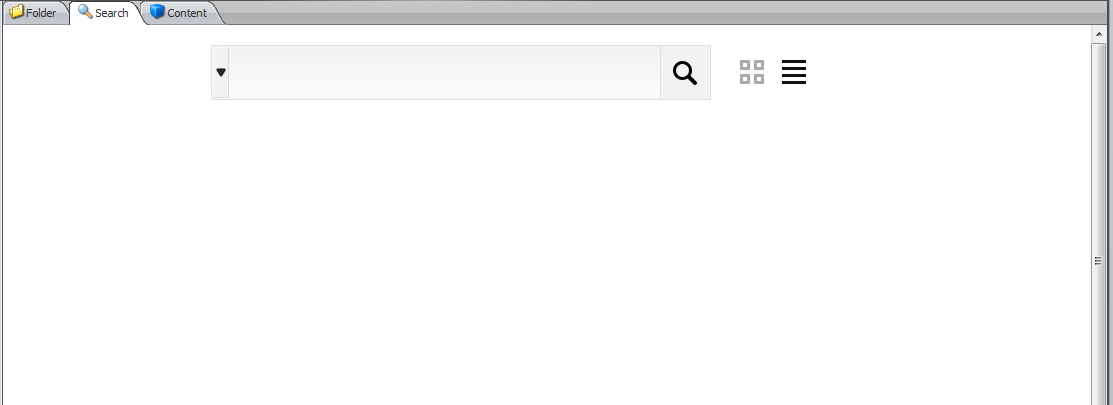
* Introducing the new Search Interface
* Search Terms
* Inserting Links and Link Management

## Introduction to the new Search Interface

This section contains an introduction on how to use the new Search Interface.

### Simple Search

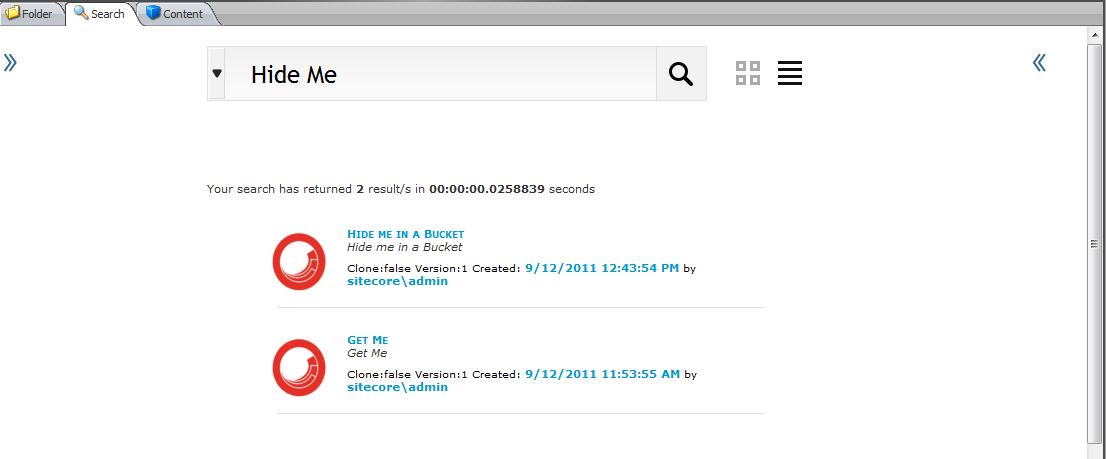
When you turn an item into a Bucket, it will automatically inherit the new search interface shown below.



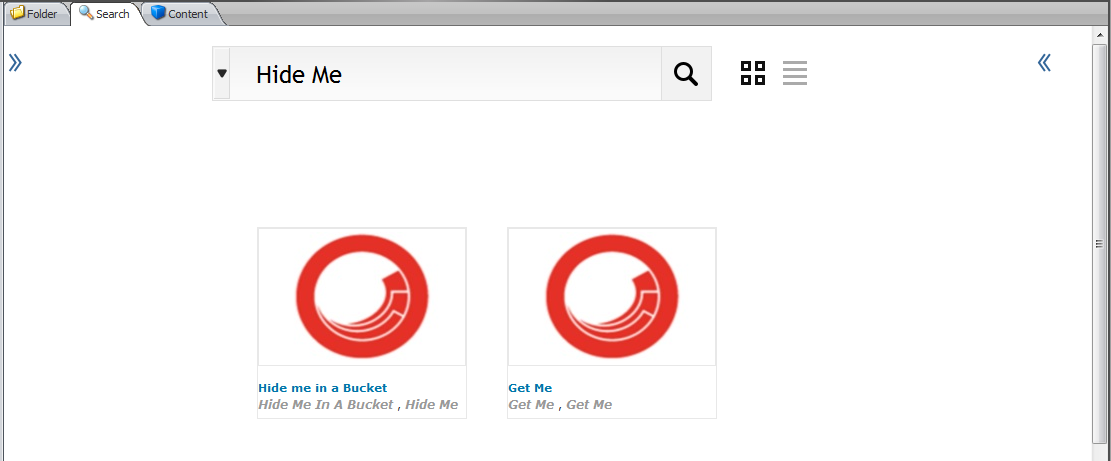
You can start searching by text e.g. “Hide Me” and either press the Enter Key or click the Search button.

You can also click the List or Grid View button to show the results in two different formats. List view is better for quick searches and Grid view is better for finding items where the image easily identifies what the item is e.g. the media library results.

**List View Results**



**Grid View Results**

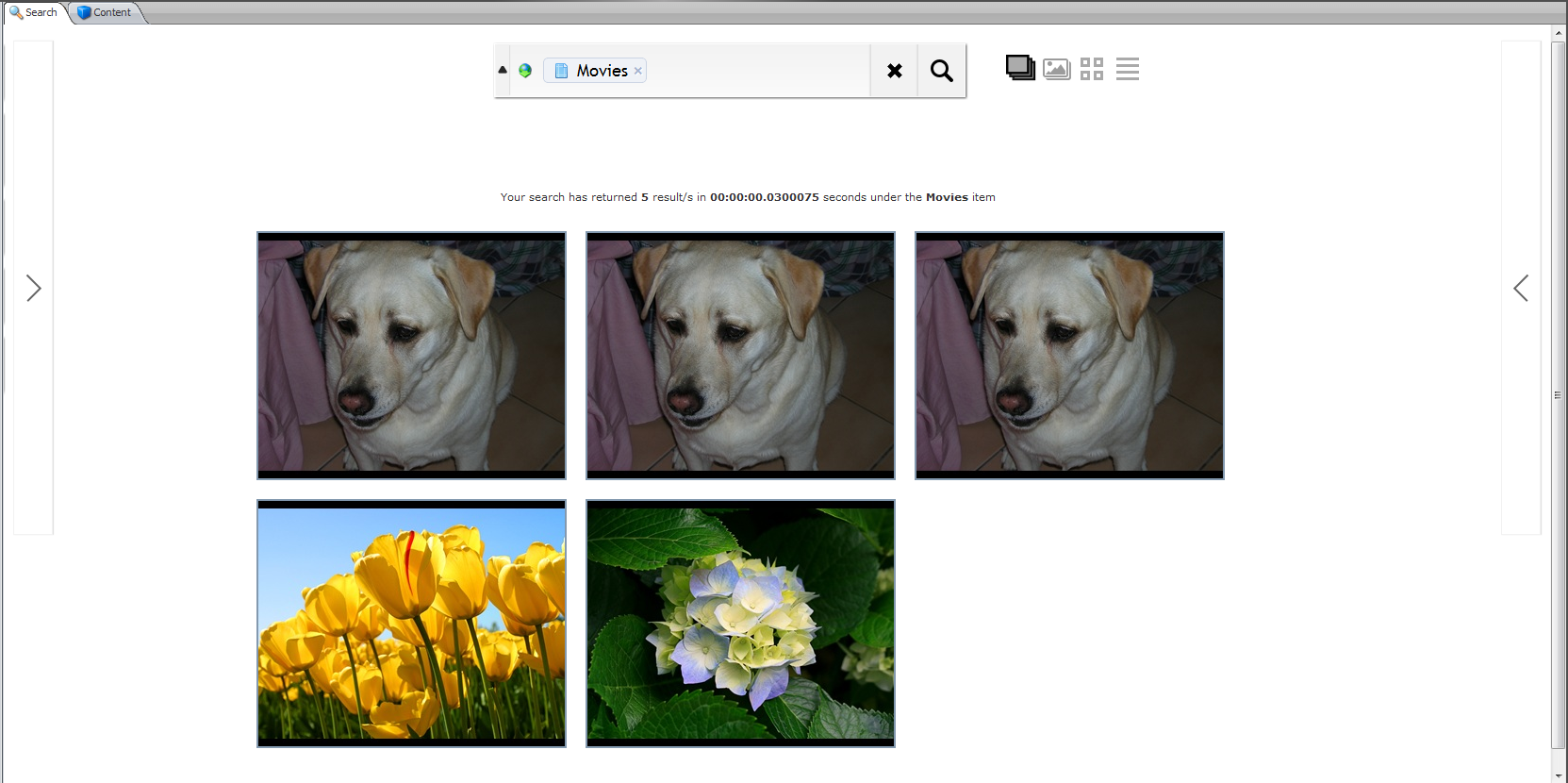


### Enabling new Views

Although not turned on by default, Item Buckets comes with other views built in. To enable them, check the “enabled” checkbox on the items under the following part in the content tree.

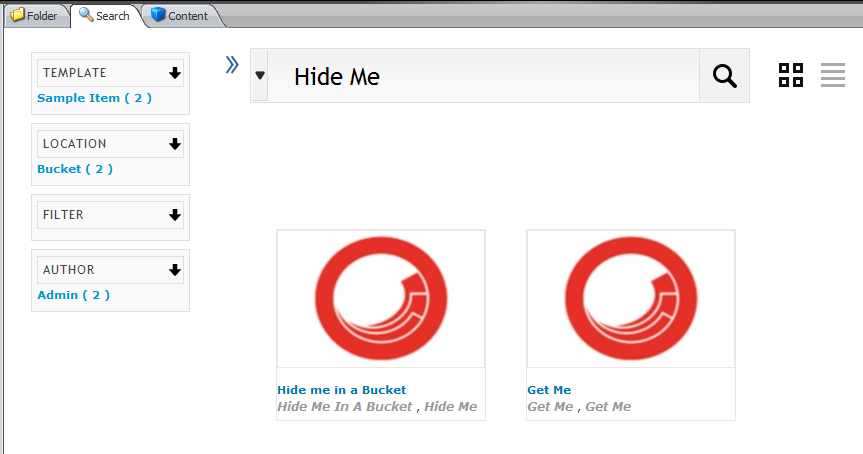
***/sitecore/system/Modules/Item Buckets/Views***

The other views returns results like this:



### Using the Facets (Category Filters)

When you run any search query, it will run Facet searches as well for you to be able to simply filter your results even more-so once you have run a query within a bucket.



By clicking on the Blue Facet links on the left, you will be able to filter your query even more. For example, the above facets show that of the 2 results that came back from the “Hide Me” search, 2 are based off the Sample Item Template, 2 are located under the “Bucket” item and 2 are written by the Admin User.

The facets included in the module include:

**Date Created**

This will group results up into 3 main groups including, created within a day, week or month.

**Author**

This will group all results based off only authors that have created an article within the system.

**Location**

This will search all bucket locations to see which bucket the results are contained within. If one of the results is not in a bucket, then it will not show up in the location filter.

**Field**

This will filter the results on if the result of a text search showed was is one of the tracked fields. You must mark fields as “Is Facet” in the template. If you have templates that have duplicate field names then this will skew the results. Please do not use fields with the same name as this will result in the lucene index concatenating these results as if they were the same field.

**Template**

This will show you the results, filtered by templates that are bucketable.

### Language Search

Item Buckets support multiple languages as well as Chinese, Arabic and non-UTF based characters.



### Complex Searches

The bucket system support complex searches such as wildcard, replacement and exact text searches.

Examples that are supported:

* Tim\*
* \*Tim
* \*Tim\*
* T\*m
* T?m
* ?im
* Ti?e
* Ti\*e
* “Tim Tim”

To run one of these searches, simply type it into the text box. Use the filters described below if you want to clean up your search.

### Range Searches

The bucket system support complex searches such as range searches.

Examples that are supported:

* price:[00000400 TO 00000500]
* price:{00000400 TO 00000500}
* title:[Algeria TO Bahrain]

To run one of these searches, simply type it into the text box. Use the filters described below if you want to clean up your search.

**NOTE:** For numeric searches, you will need to pad the values to 8 places e.g. 400 will become 00000400

### Combined, AND, OR, NOT

The bucket system support complex text searches

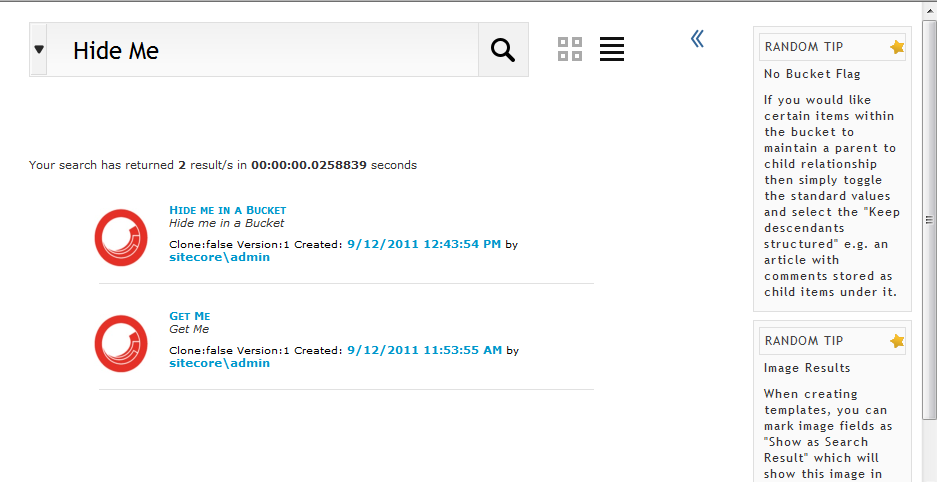
Examples that are supported:

* +Tim +Chao (results MUST have Tim AND Chao)
* +Tim Chao (results MUST have Tim AND can have Chao)
* Tim Chao (results CAN have Tim OR Chao)
* -Tim Chao (results CAN’T have Tim and can have Chao)
* +(Tim Chao) (results MUST have Tim OR Chao)
* +(Tim Chao) Shanee (results MUST have Tim OR Chao and can have Shanee)
* “Tim Chao” ~ 10 will search for results that contain the words Tim and Chao within 10 words of each other. The number must be larger than 1 and must have quotes around the text.

To run one of these searches, simply type it into the text box. Use the filters described below if you want to clean up your search.

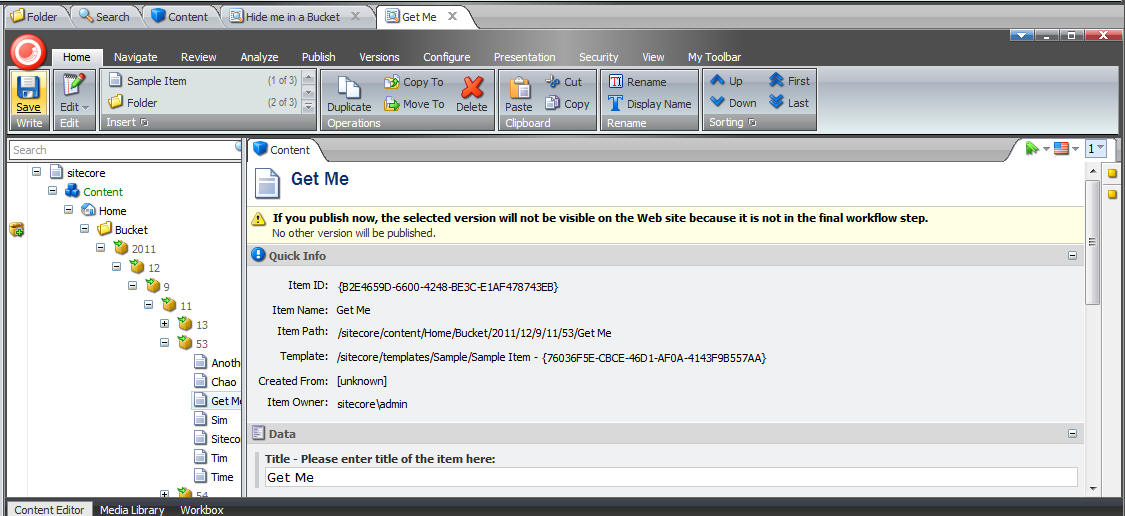
### Using the Tips

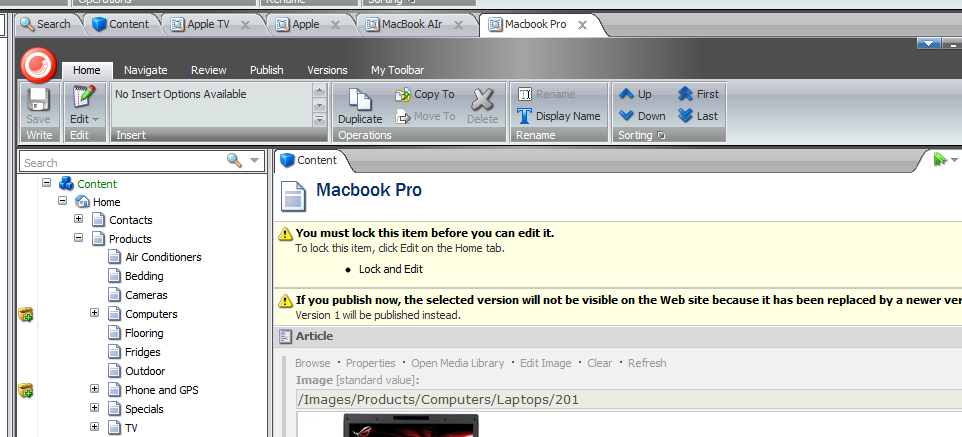
Whenever a new search is run, Sitecore will randomly choose from a list of predefined tips on how to use the advanced search options of the Item Buckets. These tips are defined in ***/sitecore/system/Modules/Item Buckets/Tips***. You can create new tips here if you see it necessary.



### Launching Search Results

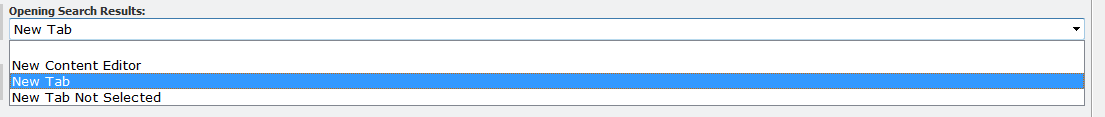
To open a result, simply click on the result image, title or any of the other blue links and it will open the item in a new content editor tab.

You can open as many of these as you want. This is an extremely handy tool for not only having multiple content items open at the same time, but to span searches and keep open multiple search results from multiple different searches e.g. I search for “**Apple**” and opened **2 results**, then searched for “**MAC**” and opened up **2 more**. If I visit other items, these tabs persist.



The tab is context sensitive and will get its own Ribbon where all the normal operations of an item will be able to be carried out e.g. Save, Preview etc.

By default, when clicking on Search Results, it will open the items up in a new tab within the content editor. This behavior can be configured within the Item Bucket Settings Item located here ***/sitecore/system/Modules/Item Buckets/Item Buckets Settings***



Currently, when you create a new item in the content tree in a bucket, your tabs will disappear.

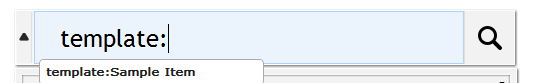
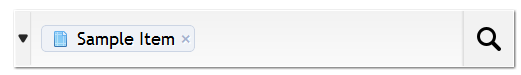
### Using Search Filters

The Item Buckets module allows users to insert filters into their search string to narrow down the possible results. The following filters ship with the module:

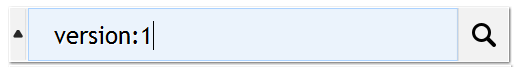
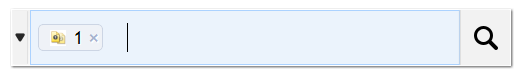
|  |  |
| --- | --- |
| * Template * Version * Language * Creation Start and End Date * File Type * Author * Tag * Site * Advanced Text * ID * Custom * Links * Sort * Location | * template: * version: * language: * end: or start: * filetype: * author: * tag: * site: * text: (click the icon to negate the text) * id: * custome:fieldname|fieldvalue * ref: * sort:fieldname (click on the icon A->Z) to change the sort order of the search). * location: |

To use a search filter, simply type in the reserved filter keyword and Item Buckets will either auto-suggest a filter or prompt for a date or prompt for text to be entered.

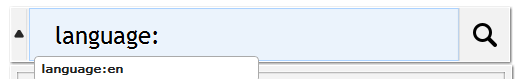
**Template Filter:**

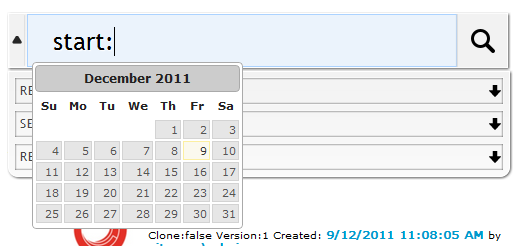
**Version Filter:**

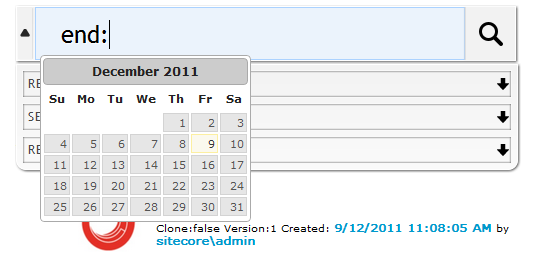
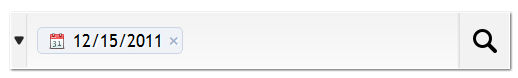
 

**Language Filter:**

**Start and End Date Filter:**

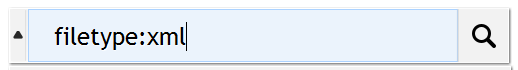
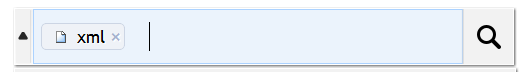
 

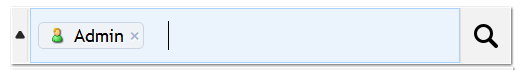
To drive the calendar from the keyboard, simple use the following keyboard shortcuts.

* page up/down - previous/next month
* ctrl+page up/down - previous/next year
* ctrl+home - current month or open when closed
* ctrl+left/right - previous/next day
* ctrl+up/down - previous/next week
* enter - accept the selected date
* ctrl+end - close and erase the date
* escape - close the datepicker without selection

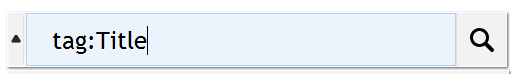
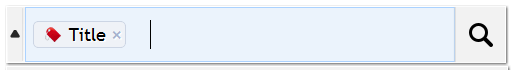
**File Type Filter:**

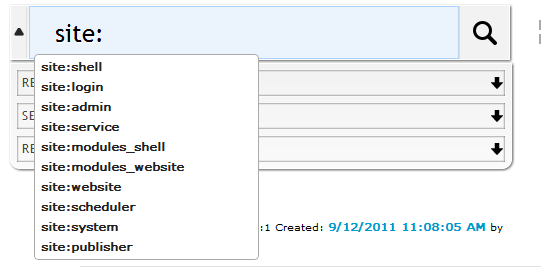
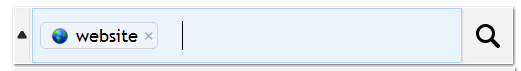
**Author Filter (You will have to wait until you have at least 2 characters typed before seeing suggested results)**

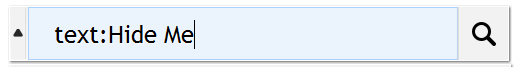
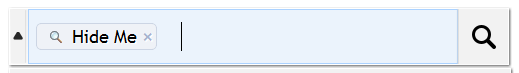
**Tag Filter**

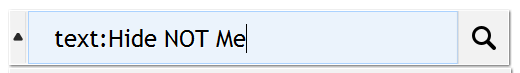
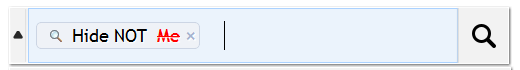
 

**Site Filter**

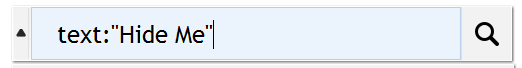
 

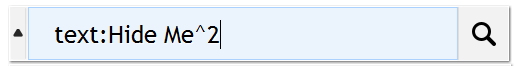
**Advanced Text:**

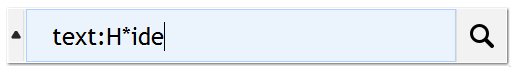
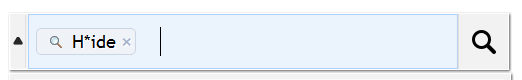
 

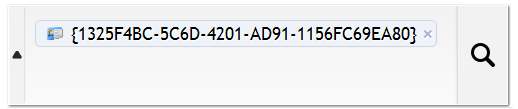
 

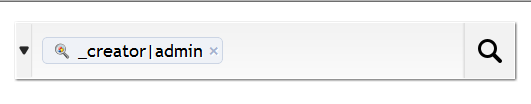


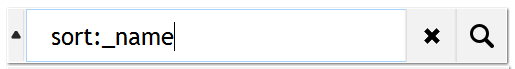
 

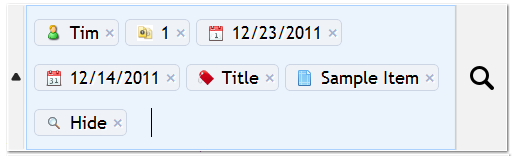
 



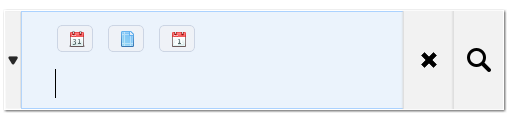
**Combining Filters:**



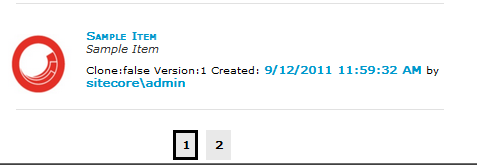
**NOTE:** All filters are currently lowercase sensitive.

### Auto-Organising

Item buckets will autoshrink searches once too many terms are in the drop box. They can be re-opened by clicking on the search.



### Paging Results

Results will be paged in results of 20 items each

. Once more than 10 pages are shown, it will ask the user to skip to the next 10.



### Search Box Clear

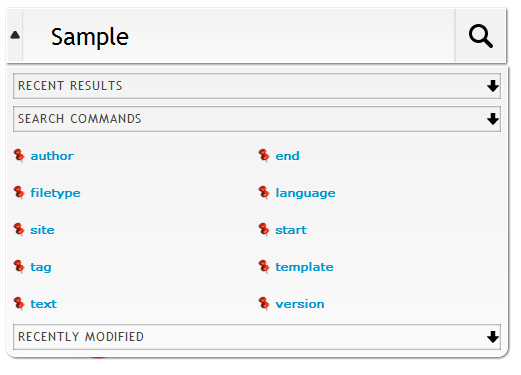
When you type in text into the search box, you will see a cross show up. This allows you to clear the text within the search box. Ctrl + X will also clear this box when it has focus.



This will also clear any of the filters you are using as well.

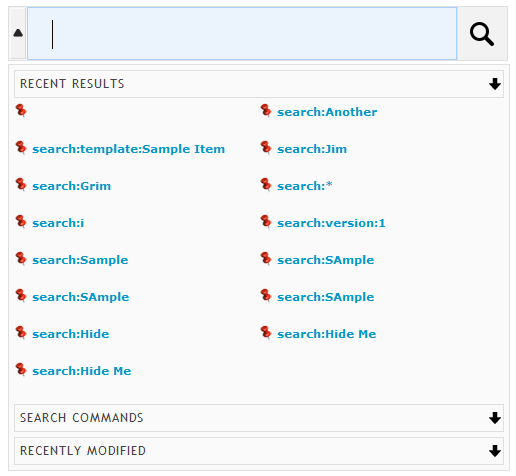
### Search Box Helpers

When clicking in the search box to start a search, you will see a drop down of Recent Results, Search Commands and Recently Modified Items. This will populare the lists on clicking the text box. This will reload it with new results on subsequent clicks of the text box. One main reason for this is to warm up the caches for subsequent searches.

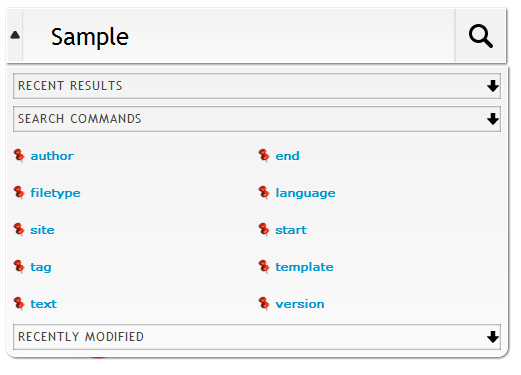


To add new Search Helpers, simply add new Helper items to the ***/sitecore/system/Modules/Item Buckets/Settings/Search Box Dropdown*** Folder.

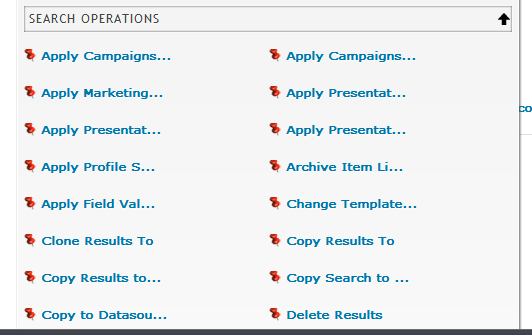
* **My Recent Searches** – This will show you the recent search queries that have been run so that you can simply call them again by clicking them instead of typing in the query again.



* **Search Commands** – This will show you all the possible Search Filter commands that are available. To add new commands add to this folder ***/sitecore/system/Modules/Item Buckets/Search Types***



* **Recently Modified** – This will show the recently edited or modified.
* **Recently Created** – This will show the recently created items.
* **Search Operations –** This is a powerful feature for being able to run any operation on the results that come back from the search. There are some operations that are shipped with Item Buckets, but mainly there is a framework for you to extend and make your own. Included Operations include applying presentation to all items, profile scores, campaigns, copying, deleting and many more.



Tip

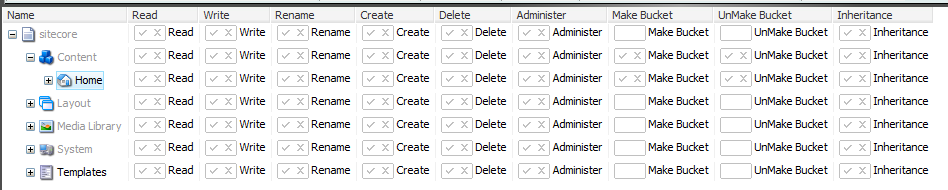
Use Security to deny certain operations to certain users. You would not want someone accidentally deleting lots of items.

### Locking Check

To minimize the possibility of accidentally bucketing an item, even an Administrator **MUST** have a lock on the item for it to be bucketed.

### Securing Bucket and UnBucket Rights

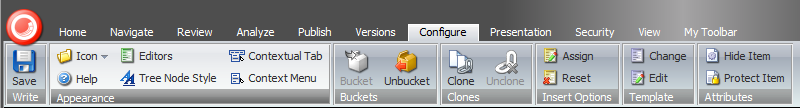
If you need to restrict permission to a user or role to be able to turn an item into a bucket or taking an item and restoring it as a normal item then you can do this through the Security Editor.



### Restoring a Bucket back to a structure before it was bucketed.

If you have turned an item into a bucket and would like to restore it back to an unbucketed item, you can do this from the Configure tab.

When a bucketing process takes place, the system will save a reference to its original parent so that you can restructure a bucketed item at any time. Items that do not have a parent reference will return as a child of the bucket itself.



Warning

This may be a long running process if you have lots of items under the one bucket. This runs on a separate thread, so if you need to continue on with other work, simply minimize the pop-up box.

### IA Modifications

The Item Buckets Module drastically changes the way that the IA of a site is managed. Items that are in a bucket no longer maintain a child to parent relationship and are simply a pool of available items that are searched with a custom search index. The normal operations of IA are still supported (but modified) with Item Buckets including:

* Copy To Bucket
* Copy From Bucket
* Move To Bucket
* Move From Bucket
* Clone To Bucket
* Clone From Bucket
* Delete Bucket – This will prompt a message saying that by deleting the bucket you could potentially be deleting hidden items under it. (These can be restored from the recycling bin).
* Drag copy into Bucket
* Drag copy out of Bucket
* Drag move into Bucket
* Drag move out of Bucket

### Keyboard Shortcuts

The following section describes the available keyboard shortcuts for using Item Buckets.

**Ctrl + Shift + B** – This will turn an item into a Bucket

**Ctrl + Shift + D** – This will unbucket an item

**Ctrl + X** – This will clear the text box.

**Ctrl + Shift + S** – This will add a search tab over any item, but will not bucket or unbucket the item.

**Ctrl + Shift + A** – This will find the closest ancestor with a search interface and select it

**Space Bar** – When the text box is not given focus, this will scroll the results.

**Ctrl + Space Bar** – This will show the dropdown options when the text box is given focus

**Esc** – This will hide the dropdown if it is shown.

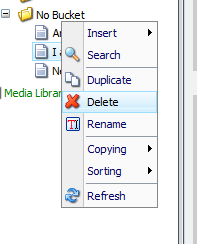
**Ctrl + B** – Focus on the text box if focus is outside the textbox.

**Shift + Number** –This will run the search in a particular view. 1 will run the first, 2, the second, and so on.

**1-9** – Pressing the numeric characters will move to that specific page of results.

### Context Menu

The following section describes the available Context Menu options for using Item Buckets.



To add a search tab to any item, you can select “Search” from the Context Menu. Adding a search tab will not turn the item into a bucket. This is perfect for items that won’t contain hidden items under it but you still want to search within descendants. You will need to refresh the item after you have added the search tab.

### Query the index from any inbuilt field supporting Lists

Now that you could potentially be working with many thousands of items, it also means that you could have thousands of items showing up as items within a multilist field, treelist etc. To be able to query them quickly you can use the new **lucene** syntax of setting a datasource. Simply start your query with “lucene:” followed by the name of a field and the value you would like to search for e.g.

**lucene:\_name:Sitecore;location:<item guid>;**

### Query the index using a custom class

Now that you could potentially be working with many thousands of items, it also means that you could have thousands of items showing up as items within a multilist field, treelist etc. To be able to query them quickly you can use a complied class that returns an Item[] to act as the source of your fields. Simple start your query with “code:” and then place the namespace.class, assembly in the source field. You can do this by implementing the IDataSource interface.

**code:Sitecore.Namespace.Class, Assembly**

### Using the new Field Types

There are new field types available for Rich Text, MultiList, Search and Links. All of these have been extended so that they can support large amounts of content without a performance hit.

**Bucket List Field** – This will allow you to attach a search query to a multilist field to only display the search results as selectable items e.g. if you wanted a multilist of all Product Items you can set the Source field of the Field to TemplateFilter=”Product ID” and it would return the items to the list.

Here are all the list of available filters (all are optional)

IndexName=bucket

Language=en

TemplateFilter=<Template ID> query:// (replace “=” in your query with “->”)

StartSearchLocation=<Parent ID> or query:// (replace “=” in your query with “->”)

FullTextQuery=Tim

FieldsFilter=Title

SortField=<Insert Lucene Term Name>

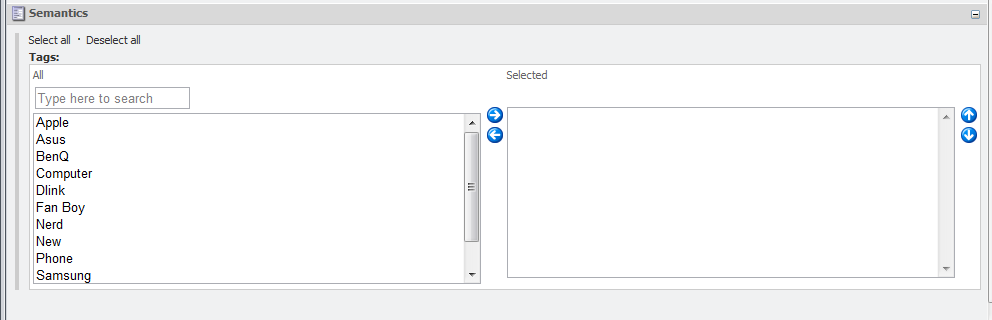
PageSize=20000

PageNumber=1 (anything lower than 0, this will not bring back results)

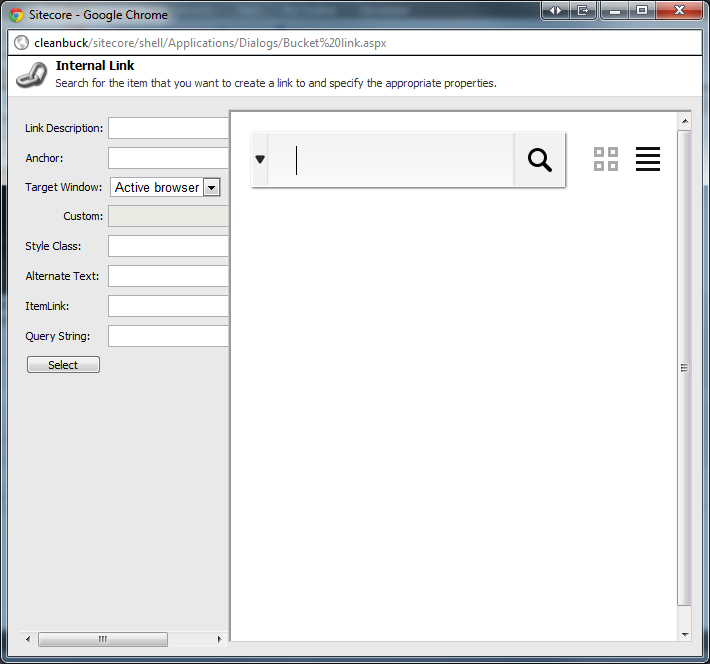
**Example 1:** TemplateFilter=<Product Template ID>|<Another Template ID>

**Example 2:** TemplateFilter=<Product Template ID>&FullTextQuery=Nicam

These queries will populate the list that you can select from and you can use the search filter to filter the list even more.



**Bucket Link Field** – This will allows you to link to items that are contained within a bucket.



**Bucket IFrameField –** If you would like to simply be able to search for items and add and remove them from a list, then you can setup a field like so.

Name = <anything>

FieldType = IFrame

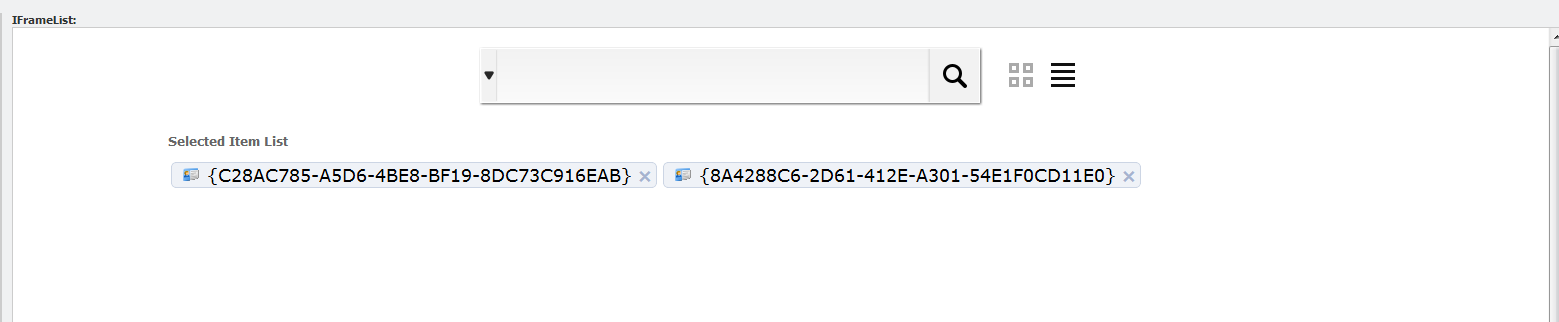
Style = height:800px;

Field Source = http://<sitename>/sitecore%20modules/Shell/Sitecore/ItemBuckets/IFrameField.aspx



This is not a NEW control, but utilises an existing field type within Sitecore to show a search interface.

This will give the following control in your content item.



To remove items from the field, simply click the cross next to the items under the “Selected Item List”. To add items to the list, simply click on the search result links.

### Using Item Buckets with the datasource of a Control

Because everything contained within a bucket will be hidden and non-selectable by users, a user can now choose the datasource for a control by specifying a search query. The syntax for the setting the datasource to run a Bucket Query is detailed below.

Types of queries:

|  |  |
| --- | --- |
| * Template * Version * Language * Creation Start and End Date * File Type * Author * Tag (Facet) * Site * Advanced Text * ID * Custom | * template: * version: * language: * end: or start: * filetype: * author: * tag: * site: * text: * id: * custom:fieldname|fieldvalue |

You must separate all queries by a “;” e.g. to search for all “Nicam Products” you could specify text:Nicam;template:<Product Template ID>;

This would be passed through to your control as a string in which you can use the BucketManager to resolve this to a list of items i.e.

var items = BucketManager.ParseDataSourceQueryForItems(((Sublayout)this.Parent).DataSource, Sitecore.Context.Item, 1, 200);

**Tips:** Sitecore is going to run the query in context to its location i.e. if you run the query on a control it will query from the Context Item down to all its descendants. If you need to do a global search or a search in another part of the content tree then you can by passing through the location parameter into the query for the datasource e.g. the following query will look for all items tagged with Nicam and that exist under the /sitecore/content/home node (the id of the home item is shown below).

**tag:{TagId};location:{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9};**

If you do not want your users to have to enter these queries in manually then you have many choices. Firstly, change the inbuilt type of the Datasource field located here -> /sitecore/templates/System/Layout/Rendering Parameters/Standard Rendering Parameters/General/Data Source.

You will need to change this from “Internal Link” to the type of “Bucket Data Source”.

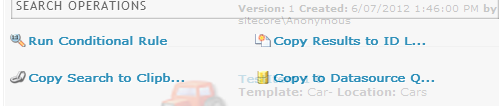
This will allow users to build these queries using a search interface. Simply click the “Build Query” button and this will pop up a search interface for you to build up your query. You can either build up a search or by clicking on one of the results, it will just return that one item as the result.

Tips

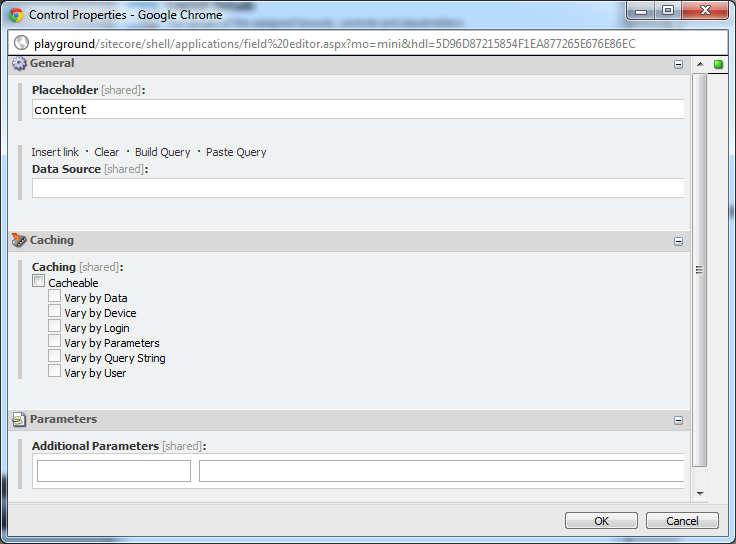
If you would like to lock the results so that any new items created after you set the datasource will not show up, then put a start and end date filter in the search. You will need both the start and end date for this too work.

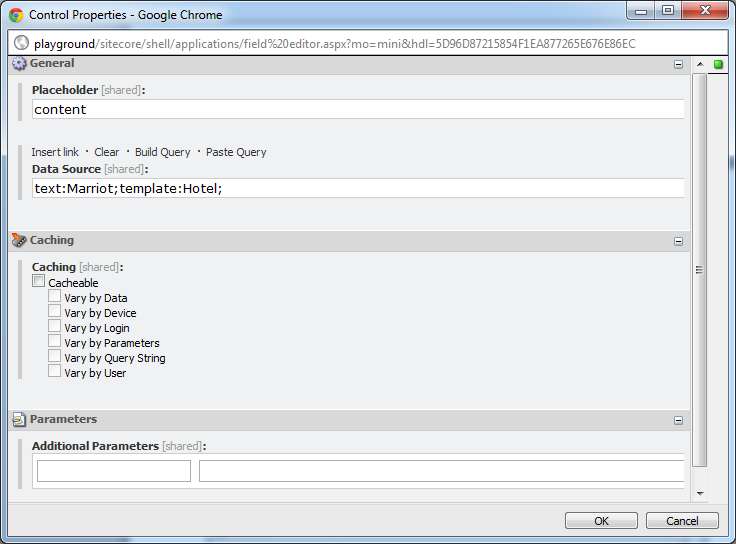
It will also allow you to paste queries from a search that you have done on an item.

To paste queries, simply search for content as you normally would. Once you have your filter, click the drop down menu and under the Search Operations, you can click on Copy to Datasource.



When you configure your presentation datasource, you will be able to paste this query into the source field.

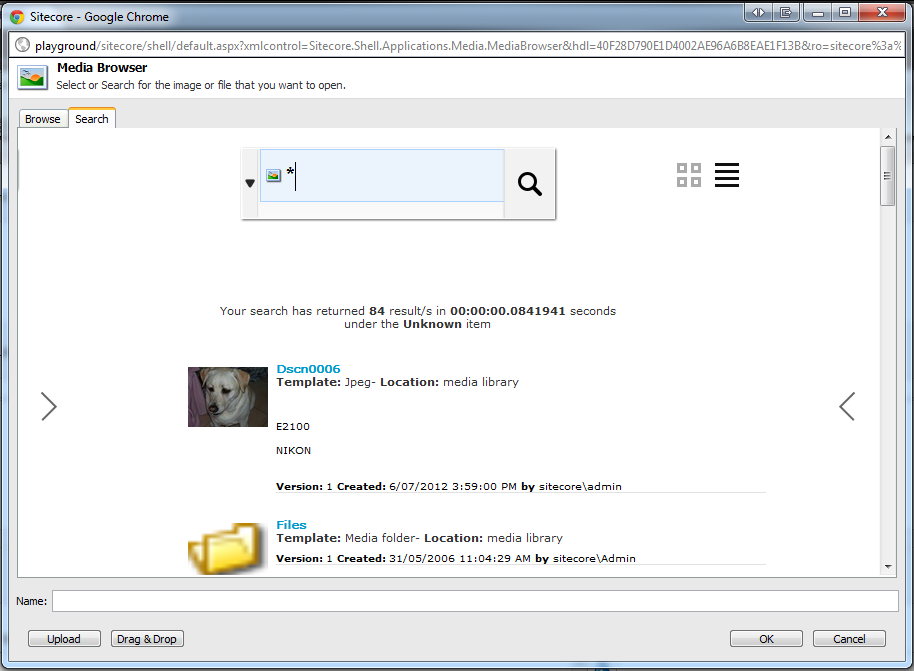




**Image Search**

Item Buckets will replace the inbuilt image fieldtype with an extended one that allows you to search for images.

Currently, this does not support having the Source field set on the template for this field.

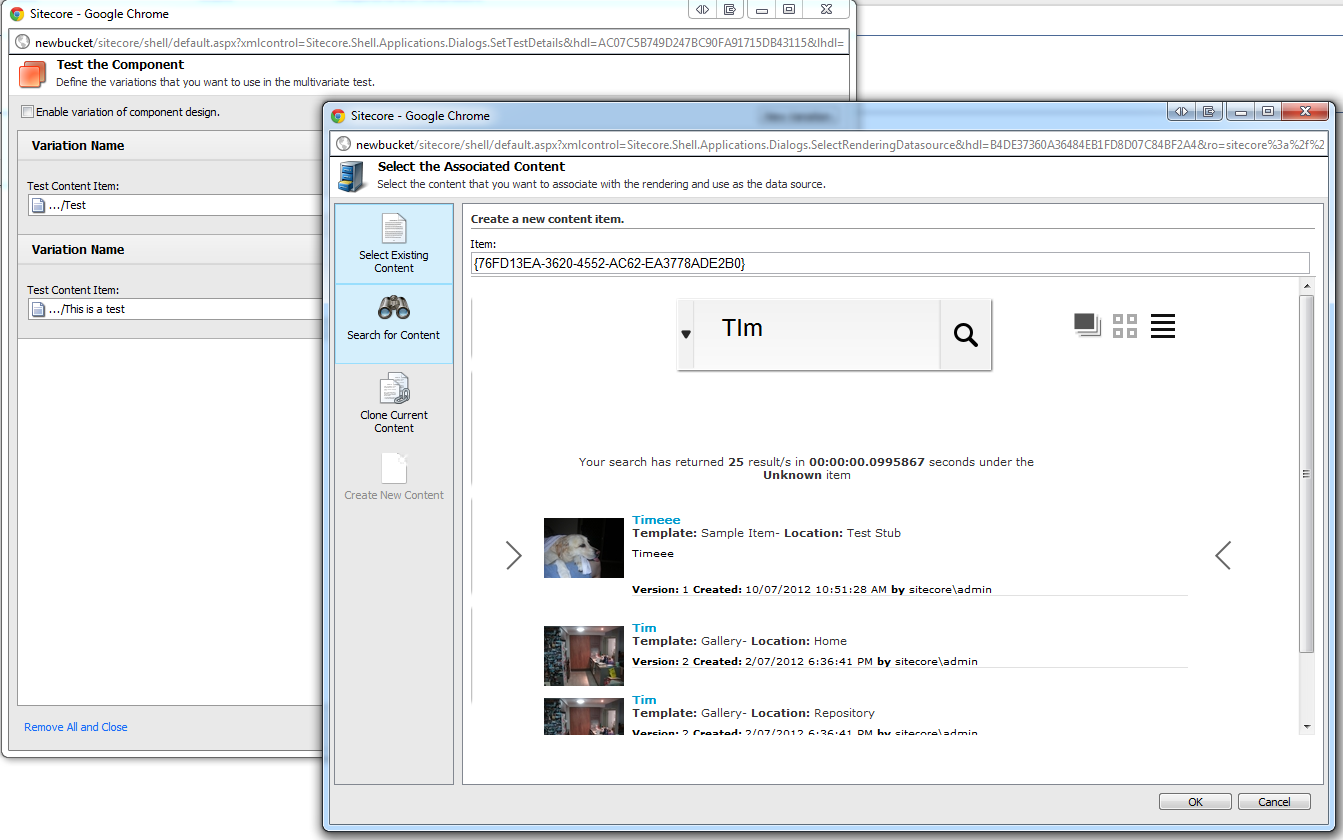
. 

## Page Editor and DMS Support

This section describes the support for Item Buckets with the Page Editor for selecting content and for personalization.

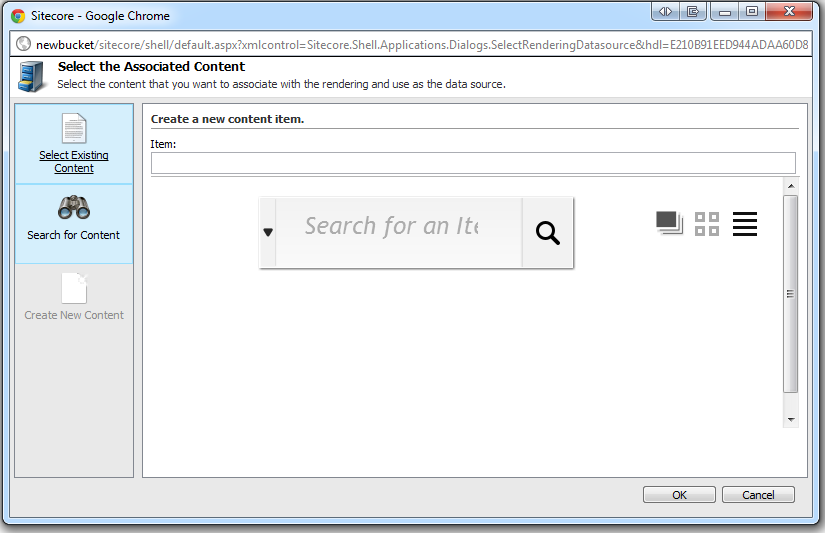
### Personalisation and MV Tests

When setting the datasource item for a test variant or a personalization rule, you now have a search tab which allows you to determine the from search for the content first.



### Setting the Datasource

When setting the datasource item for a control, you now have a new search interface to either set an individual item or a list of items.



## Inserting Links and Link Management

This section describes how to manage links to items that are contained within Buckets.

### Inserting Links

To link to items that are contained within Buckets you will need to use the new tabs that have been introduced. These are available in

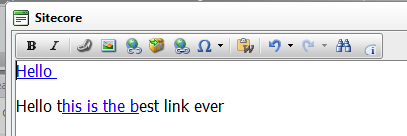
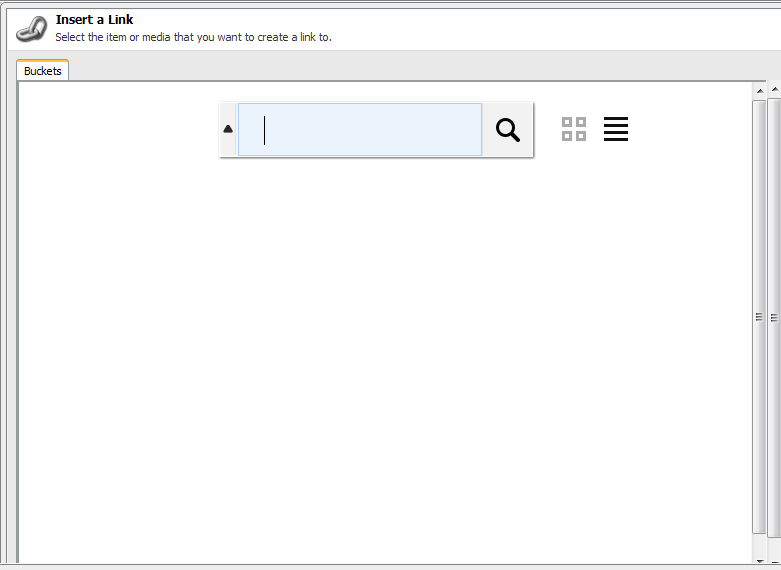
* Bucket Rich Text Editor Insert Link
* New Bucket Link Field

**Bucket Rich Text Editor**

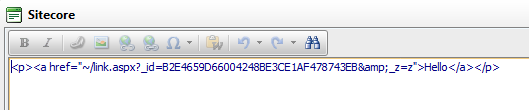
To insert a link with a Rich Text Editor field, click Show Editor in the RTE field of an item and click the Insert Sitecore Link button.

You can use the filters in your template field source to restrict the items that users can search for in the RTE, however if you do, you MUST also specify a Rich Text Editor Profile to use e.g.

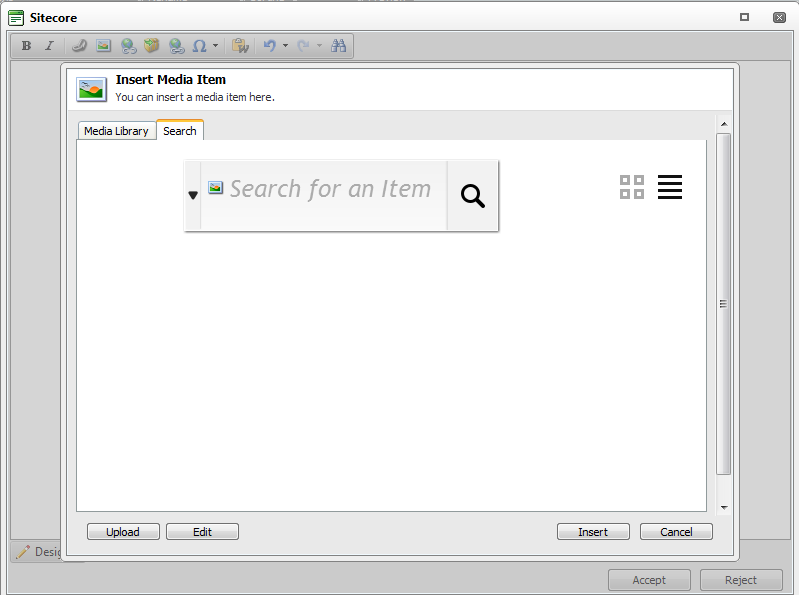
***/sitecore/system/Settings/Html Editor Profiles/Rich Text Full&StartSearchLocation={26F03C1A-778B-48E4-B3A7-70833C382C97}***

Simply run a query in this text box and click on the item to return it as the selected link.

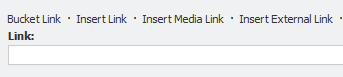


You can also insert images into the Rich Text Editor this way.



**Bucket Link**

To insert a link within the new Bucket Link field type, simply click the Bucket Link Menu Option next to the original Internal Link Menu option. (See ***working with the new field types***)



### Link Providers

The Item Bucket module ships with a link provider that will give bucket items a friendly URL. To build your own Link Providers use the standard Sitecore LinkManager implementation.

The built in Link Provider will convert a bucketed item e.g. **/sitecore/content/home/products/computers/2011/12/22/10/21/Macbook Pro**

into:

[**http://site/products/computers/apple/macbook%20pro-201112221021**](http://site/products/computers/apple/macbook%20pro-201112221021)

The logic is to format the link so that it is SEO Friendly as well as embedding any tags that are associated to the content item into the URL as well.

The inbuilt link provider is disabled by default and requires you to uncomment it in the Sitecore.ItemBuckets.config file to work.

<!-- Enable to use the default buckets Link Provider

Example: /sitecore/content/Home/Product/2011/11/11/11/11/ItemName

Output: http://<sitename>/Product/ItemName-201111111111

-->

<!--<linkManager>

<patch:attribute name="defaultProvider">buckets</patch:attribute>

<providers>

<clear />

<add name="buckets" type="Sitecore.ItemBucket.Kernel.Common.Providers.NewsLinkProvider, Sitecore.ItemBucket.Kernel" addAspxExtension="false" alwaysIncludeServerUrl="false" encodeNames="true" languageEmbedding="never" languageLocation="filePath" shortenUrls="true" useDisplayName="false" />

</providers>

</linkManager>-->

### Item Resolvers

The Item Bucket module ships with an Item Resolver to manage requests to items that site within a Bucket Folder. You will only need to extend this if you include your own Link Provider to override items contained within Buckets.

[**http://site/products/computers/apple/macbook%20pro-201112221021**](http://site/products/computers/apple/macbook%20pro-201112221021)

into:

**/sitecore/content/home/products/computers/2011/12/22/10/21/Macbook Pro**

### Support tags across any item

Due to the removal of the parent to child relationship, items contained within a bucket will require some way of being able to connect to other items. Item buckets supports a semantic tagging system which allows you to tag associations on all items. To support tagging of items, add a Bucket List field to any template and make sure it is called “tags”. For best practices, you should add this to a base template. If you think that you will tag every item in the content tree, then add this field to the standard template.

For example, if you want to tag Media Items, set the “tags” field on the File template and set the source of the field type to:

***StartSearchLocation={Tag Repository ID}&IndexName=itembuckets\_sitecore***

The reason for setting the IndexName is due to the context switch that Item Buckets does dependent upon which part of the content tree that you are querying from.

This is where the Semantics comes into play. You can add any tag to any item. For example, if you wanted to tag items with “Work in Progress”, you can now search at a later point for all items that are “Work in Progress”.

# Developing with Item Buckets

This chapter describes working with Bucket Items in the Sitecore API.

This chapter contains the following sections:

* Buckets API
* Working with new code
* Working with existing code
* Using the Search namespace
* Configuration Files

## Buckets API

This section details how to work with the Bucket API.

The Item Buckets concept is designed to address the management of millions of items of content within the content tree. This is not only from a performance perspective but all from an ease of management perspective.

Therefore, when developers are writing code, there is a new way that they will use the API to reference items.

The following standard Sitecore API methods/Properties will not work with items that are contained within a bucket:

* GetChildren() and Children

The following methods will work but are not recommend due to performance

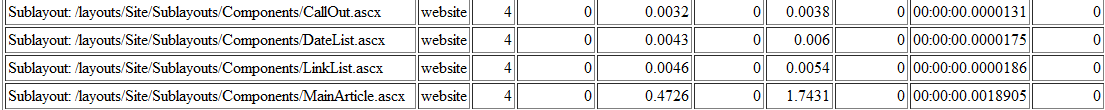
* Axes Methods and Properties

Instead, developers will talk directly to the Bucket Index via a new API to run queries on items.

|  |  |  |  |
| --- | --- | --- | --- |
| **Original Query** | **Lookup(ms)** | **Item Bucket Query** | **Lookup(ms)** |
| item.GetChildren() | 12.3 | item.Search(text:””) | 0.043 |
| item.GetChildren().Where(i => i.TemplateName == “Sample Item”); | 40.8 | item.Search(templates: ”Sample Item Template ID”) | 0.012 |
| item.GetChildren().Where(i => i.TemplateName == “Sample Item” || i.TemplateName == “Folder”); | 47.1 | item.Search(templates: ”Sample Item Template ID|Folder Template ID”) | 0.028 |
| item.Axes.GetDescendants() | 102 | item.Search(text:””) | 0.043 |
| item.Axes.GetDescendants().Where(i => i.Fields[“Related Articles”].Contains(“<GUID>”); | 114 | item.Search(RelatedIds: “<GUID>”) | 0.091 |
| item.Children.ToList().Where(i => i.Name == “Sample”); | 89 | item.Search(text: ”Sample”) | 0.018 |

Using this table above, you will need to migrate your own code to work with the new bucketing system. As you can see, the performance gains are immense. This is with only 1000 items in the content section of the content tree.

You can see the gains of performance in your presentation components using the **stats.aspx** page



**NOTE:** Notice that the average time to load is extremely low and that because the presentation components are going straight to the index and then filling the objects in the Bucket data layer.

**Example Queries:**

**NOTE:** For simplicity, the methods below have exluded the hitCount attribute that gets returns through an “out” parameter so that you can page results if necessary.

**Using the BucketManager**

//Test 1 - Get All Items Under the Home Item of Template "Sample Item"

var HomeDescendantsOfTypeSampleItem =

BucketManager.Search(Sitecore.Context.ContentDatabase.GetItem(

"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}"), templates: "{76036F5E-CBCE-46D1-AF0A-4143F9B557AA}");

//Test 2 - Get All Items Under Tim Folder that have the Title Field Starting with the Word Tim

var TimDescendantsWithTitleOfTim =

BucketManager.Search(Sitecore.Context.ContentDatabase.GetItem(

"{27812FBA-F5E6-41B9-9DDE-4A82AE81496C}"), new SafeDictionary<string> { { "title", "Tim" } });

//Test 3 – All Items Under the Repository Item that have an Item Name of Tim

var RepositoryFolderWithNameOfTim =

BucketManager.Search(Sitecore.Context.ContentDatabase.GetItem(

"{346C12DF-8FC7-4B97-8570-9D26F78240F2}"), new SafeDictionary<string> { { "\_name", "Tim" } });

//Test 4 - Get All Items Under Home of Template "Sample Item"

var HomeDescendantsOfTypeArticleWithTimContainedWithinIt =

BucketManager.Search(Sitecore.Context.ContentDatabase.GetItem(

"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}"),

templates: "{14633DB7-360E-447F-808B-B71128628009}", text: "Tim");

//Test 5 - Items under Home that contain the word Tim, sort by Name

var TimItemsSortedByName =

BucketManager.Search(Sitecore.Context.ContentDatabase.GetItem(

"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}"),

text: "Tim",

sortField: "\_name");

//Test 6 - Items under Home that contain the word Tim, sort by Name

var TimItemsOfTypeSampleItemSortedByName =

BucketManager.Search(Sitecore.Context.ContentDatabase.GetItem(

"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}"),

text: "Tim",

sortField: "\_name",

templates: "{76036F5E-CBCE-46D1-AF0A-4143F9B557AA}");

//Test 7 - Items under Home that contain the word Tim, sort by Name

var ItemsUnderHomeContainingBrisbane =

BucketManager.Search(Sitecore.Context.ContentDatabase.GetItem(

"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}"),

text: "Brisbane");

//Test 8 – Get by Id

var GetVersion3OfItem =

BucketManager.Search(Sitecore.Context.ContentDatabase.GetItem(

"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}"),

id: "{344E1BED-B68C-4E13-9689-97BB7797D844}");

//Test 9 – Complex Search

var ComplexSearch =

BucketManager.Search(Sitecore.Context.ContentDatabase.GetItem(

"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}"),

text: "Tim", startDate: "03/12/2012", endDate: "03/26/2012", numberOfItemsToReturn: 60, language: "en", sortField: "title");

**Using the Item Extension Methods**

//Test 1 - Get All Items Under the Home Item of Template "Sample Item"

var HomeDescendantsOfTypeSampleItem =

Sitecore.Context.ContentDatabase.GetItem.Search(

templates: "{76036F5E-CBCE-46D1-AF0A-4143F9B557AA}");

//Test 2 - Get All Items Under Tim Folder that have the Title Field Starting with the Word Tim

var TimDescendantsWithTitleOfTim =

Sitecore.Context.ContentDatabase.GetItem(

"{27812FBA-F5E6-41B9-9DDE-4A82AE81496C}").Search(

new SafeDictionary<string> { { "title", "Tim" } });

//Test 3 – All Items Under the Repository Item that have an Item Name of Tim

var RepositoryFolderWithNameOfTim =

Sitecore.Context.ContentDatabase.GetItem(

"{346C12DF-8FC7-4B97-8570-9D26F78240F2}").Search(

new SafeDictionary<string> { { "\_name", "Tim" } });

//Test 4 - Get All Items Under Home of Template "Sample Item"

var HomeDescendantsOfTypeArticleWithTimContainedWithinIt =

Sitecore.Context.ContentDatabase.GetItem(

"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}").Search(

templates: "{14633DB7-360E-447F-808B-B71128628009}",

text: "Tim");

//Test 5 - Items under Home that contain the word Tim, sort by Name

var TimItemsSortedByName =

Sitecore.Context.ContentDatabase.GetItem(

"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}").Search(

sortField: "\_name");

//Test 6 - Items under Home that contain the word Tim, sort by Name

var TimItemsOfTypeSampleItemSortedByName =

Sitecore.Context.ContentDatabase.GetItem(

"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}").Search(

text: "Tim",

sortField: "\_name",

templates: "{76036F5E-CBCE-46D1-AF0A-4143F9B557AA}");

//Test 7 - Items under Home that contain the word Tim, sort by Name

var ItemsUnderHomeContainingBrisbane =

Sitecore.Context.ContentDatabase.GetItem(

"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}").Search(

text: "Brisbane");

//Test 8 – Get by Id

var GetVersion3OfItem =

Sitecore.Context.ContentDatabase.GetItem(

"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}").Search(

id: "{344E1BED-B68C-4E13-9689-97BB7797D844}");

//Test 9 – Complex Search

var ComplexSearch =

Sitecore.Context.ContentDatabase.GetItem(

"{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}").Search(

text: "Tim", startDate: "03/12/2012", endDate: "03/26/2012", numberOfItemsToReturn: 60, language: "en", sortField: "title");

**Using Linq Notation with a BucketQuery Collection**

You can also use Lambda and Linq expressions to chain searches together.

//Test 1 – The first 200 results of Page 1 of results where items under the Context Item where the content contains the numbers 53 and is based off a Template with ID of {F9935D0B-D84D-46AF-B420-A67A6022193F}” where the name starts with Test and the Language is the context language only for content that is 2 days old and sorted by the \_name field.

var results = new BucketQuery().WhereContentContains("53")

.WhereTemplateIs("{F9935D0B-D84D-46AF-B420-A67A6022193F}")

.WhereItemNameIs("Test\*")

.WhereLanguageIs(Sitecore.Context.Language)

.Starting(DateTime.Now.AddDays(-2))

.Ending(DateTime.Now)

.SortBy("\_name")

.Page(1, 200, out hitCount);

var movies = new BucketQuery().WhereContentContains("Dark")

.WhereTemplateIs("{D3335D0B-D84D-46AF-C620-A67A6022AB3F}")

.WhereLanguageIs(Sitecore.Context.Language)

.WhereTaggedWith("Tag ID for Tim Burton")

.WhereTaggedWith("Tag ID for Johnny Depp")

.WhereTaggedWith("Tag ID for Helen Bohnam-Carter")

.Starting(DateTime.Now.AddYears(-12))

.Ending(DateTime.Now)

.SortBy("\_name")

.Page(1, 200, out hitCount);

## Working with new code

Designing your IA and code will work better if you are starting on a new implementation using the Item Buckets module. As long as you don’t use the methods mentioned above, then all your presentation components will not only work with bucket containers but will be considerably faster for looking up items as well.

Developers will be given new classes and namespaces to work with the Buckets API. Here is a detailed list of the available class and namespaces.

|  |  |  |
| --- | --- | --- |
| Namespace | Class | Description |
| Sitecore.ItemBucket.Kernel.ItemExtensions.Axes | BucketItemAxes | If you add this to your using statements in your CS file you will get extension method replacements for using GetChildren(), Children and Axes methods. |
| Sitecore.ItemBucket.Kernel.ItemExtensions.Axes | ItemExtensions | This will allow you to have new extension methods and properties on an item object to simply be able to run queries on an item like so item.Search(“”) or item.FullSearch(“”) |
| Sitecore.ItemBucket.Kernel.Managers | BucketManager | This is your main entry point for working with the Bucket containers. It contains methods such as IsBucket(), IsBucketItem(), GetParentBucket() etc. This also allows you to run Searches as well. |
| Sitecore.ItemBuckets.Kernel.Search.Query | BucketQuery | The entry point for running Lambda or Linq expressions that get converted into Lucene queries. |
| Sitecore.ItemBucket.Kernel.Util | SitecoreItem | To keep memory to a minimum, all searches will return a SitecoreItem which is a stripped-back representation of the Item class. |
| Sitecore.ItemBucket.Kernel.Kernel.Hooks | QueryWarmUp | An abstract class that allows you to specify warm-up queries that run when Sitecore is initializing. This is useful to run common queries so that they are cached when requested another time. This sacrifices startup time for operation performance. |
| Sitecore.ItemBucket.Kernel.Presentation | BucketPresentationExtensions | A helper class for converting a string datasource to a Bucket Query. |

## Working with Extension Points

If the functionality of the Item Buckets is not all that you need, there are many extension points for you to extend the features. The following is a list of Interfaces that are meant to be extended if necessary.

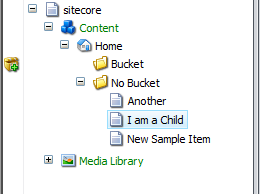
**Interfaces**

|  |  |  |
| --- | --- | --- |
| Namespace | Interface | Description |
| Sitecore.ItemBucket.Kernel.FieldTypes | IDataSource | Implement this interface if you would like to be able to have list field types within your Sitecore template take advantage of populating itself from a lucene query. |
| **Example of IDataSource Implementation** | BucketListQuery | public class BucketListQuery : IDataSource  {  public Item[] ListQuery(Item itm)  {  return itm.Children.ToArray();  }  } |
| Sitecore.ItemBucket.Kernel.Kernel.Interfaces | IBucketController | If you would like to build your own UI layer that can send the Bucket Handler a request and receive back a list of items, implement this interface. |
| Sitecore.ItemBucket.Kernel.Kernel.Interfaces | IBucketSearchQuery | There are many filters that come with the Item Buckets e.g. Author, Start Date, Text, Tags etc. If you would like to implement a new Filter then implement this interface. |
| Sitecore.ItemBucket.Kernel.Kernel.Interfaces | ITag | A Tag Repository works with the ITag Interface. If you have Tags that are pulled from external systems then you will need to implement this interface to be able to use these tags to tag items and then search by them as well. |
| Sitecore.ItemBucket.Kernel.Kernel.Interfaces | ITagRepository | Item Buckets comes with an implementation of a Tag Repository using Items within the content tree. If you have an existing Tag Repository and would like to use this to search for tagged content within Sitecore then you will need to implement a ITagRepository. |
| Sitecore.ItemBucket.Kernel.Kernel.Search.SearchOperations | ISearchOperation | If you would like to introduce new actions to do on a list of search results, then you will need to implement this interface. For example, if you want to search for all items in the content tree that had $name in any of the fields and replace them, then you would could implement a new ISearchOperation so that authors could do this. |
| Sitecore.ItemBucket.Kernel.Search | IFacet | Item Buckets ships with 5 different types of faceting.   1. Templates 2. Fields 3. Dates 4. Locations 5. Authors   If you would like to introduce your own faceting categories then you only need to implement the IFacet interface. |
| Sitecore.ItemBucket.Kernel.Search | ISearchDropDown | When running a search you will see a dropdown that is shown from the textbox where you enter your text. If you would like to add your own, firstly you will need to implement this interface. Secondly, you will need to add an item into the content tree (/**sitecore/system/Modules/Item Buckets/Settings/Search Box Dropdown**) to register this class so that it will show up in the drop down menu. |

### Working with Existing Code

If you are installing Item Buckets into an existing solution you need to be extra careful and should do this on a developer’s machine first. It is never recommended to install this directly onto an authoring, production or any other server until the functionality is proven to work with your existing code.

**Example 1:** If you have a control built that looks at an item and displays its children.



Your code behind would look something like this.

protected void Page\_Load(object sender, EventArgs e)

{

repeater.DataSource = Sitecore.Context.Item.GetChildren();

repeater.DataBind();

/\* We can assume that the markup is showing the name of the item in the datasource \*/

}

This would render

**Another**

**I am a Child**

**New Sample Item**

Once you turn the “No Bucket” folder into a bucket, and use the same code show above:

This would render

**2011**

Changing your code to us the Bucket API to

protected void Page\_Load(object sender, EventArgs e)

{

repeater.DataSource = Sitecore.Context.Item.Search(text: “\*”);

repeater.DataBind();

/\* We can assume that the markup is showing the name of the item in the datasource \*/

}

OR

protected void Page\_Load(object sender, EventArgs e)

{

repeater.DataSource = new BucketQuery().WhereContentContains(“\*).Run();

repeater.DataBind();

/\* We can assume that the markup is showing the name of the item in the datasource \*/

}

OR

protected void Page\_Load(object sender, EventArgs e)

{

repeater.DataSource = BucketManager.Search(text:”\*”)

repeater.DataBind();

/\* We can assume that the markup is showing the name of the item in the datasource \*/

}

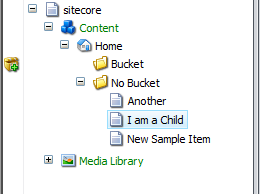
This would render

**Another**

**I am a Child**

**New Sample Item**

**Example 2:** If you have a control built that looks at an item and displays its descendants.



Your code behind would look something like this.

protected void Page\_Load(object sender, EventArgs e)

{

repeater.DataSource = Sitecore.Context.Item.Axes.GetDescendants();

repeater.DataBind();

/\* We can assume that the markup is showing the name of the item in the datasource \*/

}

This would render

**Another**

**I am a Child**

**New Sample Item**

Once you turn the “No Bucket” folder into a bucket, and use the same code show above:

This would render

**2011**

**12**

**9**

**15**

**1**

**Another**

**I am a Child**

**New Sample Item**

Changing your code to us the Bucket API to

protected void Page\_Load(object sender, EventArgs e)

{

repeater.DataSource = Sitecore.Context.Item.Search(text: “\*”);

repeater.DataBind();

/\* We can assume that the markup is showing the name of the item in the datasource \*/

}

This would render

**Another**

**I am a Child**

**New Sample Item**

### Common Requirements

A lot of websites are very similar and have similar requirements. Below is a list of example queries to retrieve items for common web controls.

**Example 1:** Side Menu (Get all descendants of type Template “Site Section”.)

**Code**

**//**This will use the data source that is specified on a control to query the buckets for items.

var items = BucketManager.ParseDataSourceQueryForItems(((Sublayout) this.Parent).DataSource, Sitecore.Context.Item, 0, 20);

**//**This will use the string that is specified in the method\ query the buckets for items.

var items = BucketManager.ParseDataSourceQueryForItems((“<Insert Query Here>”, Sitecore.Context.Item, 0, 20);

### Creating a Tag Repository

To create new tag repositories and to enable the tag filter in searches, create a new class in your Visual Studio Solution and implement the ITagRepository interface.

You will need to implement methods for returning All Tags, Single Tags, the First Tag, Get Tags by Name, Get Tags by Value.

The following code will pull a tag repository from a bucket in the content tree.

public class SitecoreHostedTagRepository : ITagRepository

{

public IEnumerable<Tag> All()

{

var tagParent = Context.ContentDatabase.IsNull()

? Context.Database.GetItem(((ReferenceField)Sitecore.ItemBucket.Kernel.Util.Constants.SettingsItem.Fields["Tag Parent"]).TargetItem.ID)

: Context.ContentDatabase.GetItem((

(ReferenceField)Sitecore.ItemBucket.Kernel.Util.Constants.SettingsItem.Fields["Tag Parent"]).TargetItem.ID);

var hitsCount = 0;

return tagParent.Search(out hitsCount, location: "{836B232E-0A67-4A73-9306-F5844DB74733}", templates: "{58DA2398-0F91-4989-AB76-78DAC905E775}").Select(itemreturn => new Tag(itemreturn.GetItem().Name, itemreturn.GetItem().ID.ToString()));

}

public Tag Single(Func<Tag, bool> exp)

{

var tagParent = Context.ContentDatabase.IsNull()

? Context.Database.GetItem(((ReferenceField)Sitecore.ItemBucket.Kernel.Util.Constants.SettingsItem.Fields["Tag Parent"]).TargetItem.ID)

: Context.ContentDatabase.GetItem((

(ReferenceField)Sitecore.ItemBucket.Kernel.Util.Constants.SettingsItem.Fields["Tag Parent"]).TargetItem.ID);

var hitsCount = 0;

return tagParent.Search(out hitsCount, location: "{836B232E-0A67-4A73-9306-F5844DB74733}").Select(itemreturn => new Tag(itemreturn.GetItem().Name, itemreturn.GetItem().ID.ToString())).Single();

}

public Tag First(Func<Tag, bool> exp)

{

var tagParent = Context.ContentDatabase.IsNull()

? Context.Database.GetItem(((ReferenceField)Sitecore.ItemBucket.Kernel.Util.Constants.SettingsItem.Fields["Tag Parent"]).TargetItem.ID)

: Context.ContentDatabase.GetItem((

(ReferenceField)Sitecore.ItemBucket.Kernel.Util.Constants.SettingsItem.Fields["Tag Parent"]).TargetItem.ID);

var hitsCount = 0;

return tagParent.Search(out hitsCount, location: "{836B232E-0A67-4A73-9306-F5844DB74733}").Select(itemreturn => new Tag(itemreturn.GetItem().Name, itemreturn.GetItem().ID.ToString())).First();

}

public IEnumerable<Tag> GetTags(string contains)

{

var tagParent = Context.ContentDatabase.IsNull()

? Context.Database.GetItem(((ReferenceField)Sitecore.ItemBucket.Kernel.Util.Constants.SettingsItem.Fields["Tag Parent"]).TargetItem.ID)

: Context.ContentDatabase.GetItem("{380E56C2-801A-486F-BA5C-4E545701C146}");

var hitsCount = 0;

return tagParent.Search(out hitsCount, location: "{836B232E-0A67-4A73-9306-F5844DB74733}", text: contains, templates: "{58DA2398-0F91-4989-AB76-78DAC905E775}").Select(itemreturn => new Tag(itemreturn.GetItem().Fields["Name"].Value, itemreturn.GetItem().ID.ToString()));

}

public Tag GetTagByValue(string value)

{

var tagParent = Context.ContentDatabase.IsNull()

? Context.Database.GetItem(((ReferenceField)Sitecore.ItemBucket.Kernel.Util.Constants.SettingsItem.Fields["Tag Parent"]).TargetItem.ID)

: Context.ContentDatabase.GetItem("{380E56C2-801A-486F-BA5C-4E545701C146}");

var hitsCount = 0;

return tagParent.Search(out hitsCount, location: "{836B232E-0A67-4A73-9306-F5844DB74733}", id: value).Select(itemreturn => new Tag(itemreturn.GetItem().Fields["Name"].Value, itemreturn.GetItem().ID.ToString())).First();

}

}

You can have as many tag repositories as you need and they can quite easily come from different sources as well. Once you have created your code above, you will need to create a Tag Repository Item under /sitecore/system/Modules/Item Buckets/Tag Repositories in the content tree. You will need to point to the class you compiled above with the namespace.class, assembly syntax.

There is a built in tag repository implementation. To enable this in your site, you need to create a field called “Tags” on any template. This field must be of type multi-list or bucket-list. You then need to point the “Sitecore Item Buckets Settings Item” (/sitecore/system/Modules/Item Buckets/Item Buckets Settings) – Tag Parent Field to the parent where all the tags live. The parent of the tags items can be a bucket if need be. Tags is a great candidate to be a bucket.

### Creating a new Facet

If you would like to offer your Authors the ability to filter there search on different facets then you will need to do this via code. You will need to implement the IFacet interface. Here is an example that facets on all the bucket locations in the content tree.

class LocationFacet : IFacet

{

public List<FacetReturn> Filter(List<SearchStringModel> searchQuery, string locationFilter)

{

var buckets = new List<SitecoreItem>();

using (var searcher = new IndexSearcher(Util.Constants.Index.Name))

{

if (locationFilter.IsNotEmpty())

{

buckets.AddRange(

searcher.GetItemsViaFieldQuery("isbucket", "1").Value.Where(item => item.GetItem().IsNotNull()).Where(

itm => Context.ContentDatabase.GetItem(locationFilter).Axes.IsAncestorOf(itm.GetItem())));

}

}

var returnFacets =

GetSearch(buckets.Select(item => item.GetItem().ID.ToString()).ToList(), searchQuery,

locationFilter).Select(

facet =>

new FacetReturn

{

KeyName = Sitecore.Context.ContentDatabase.GetItem(facet.Key).Name,

Value = facet.Value.ToString(),

Type = "location",

ID = facet.Key

});

return returnFacets.ToList();

}

public Dictionary<String, int> GetSearch(List<String> Filters, List<SearchStringModel> \_searchQuery, string LocationFilter)

{

using (var searcher = new IndexSearcher(Util.Constants.Index.Name))

{

DateRangeSearchParam query = SearchHelper.GetBaseQuery(\_searchQuery, LocationFilter);

var results = searcher.RunFacet(searcher.ContructQuery(query), false, true, 0, 0, "\_path", Filters);

return results;

}

}

}

You will then need to create a Facet Item under the /sitecore/system/Modules/Item Buckets/Facets folder in the content tree and fill in the namespace.class, assembly.

### Configuration Files

The modules installs with a Sitecore.ItemBuckets.config file which contains some configuration. This section explains the anatomy of this config file.

**Custom Index**

<search>

<configuration>

<indexes>

<index id="buckets" type="Sitecore.Search.Index, Sitecore.Kernel">

<param desc="name">$(id)</param>

<param desc="folder">buckets</param>

<Analyzer ref="search/analyzer" />

<locations hint="list:AddCrawler">

<ItemSearch type="Sitecore.ItemBucket.Kernel.Crawlers.CustomCrawler,Sitecore.ItemBucket.Kernel">

<Database>master</Database>

<Root>/sitecore/content</Root>

<IndexAllFields>true</IndexAllFields>

<fieldTypes hint="raw:AddFieldTypes">

<fieldType name="multilist" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

<fieldType name="treelist" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

<fieldType name="treelistex" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

<fieldType name="checklist" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

<fieldType name="tree list" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

</fieldTypes>

<include hint="list:ExcludeTemplate">

<layout>{ADB6CA4F-03EF-4F47-B9AC-9CE2BA53FF97}</layout>

</include>

</ItemSearch>

</locations>

</index>

</indexes>

</configuration>

</search>

In the above configuration we are declaring a new index called “buckets”. We are then using a custom crawler so that we can tokenize and list based field types. This makes it possible to search within list items.

**Custom Cache**

<database id="master" singleInstance="true" type="Sitecore.Data.Database, Sitecore.Kernel">

<cacheSizes hint="setting">

<data>100MB</data>

<items>100MB</items>

<paths>4MB</paths>

<standardValues>4MB</standardValues>

</cacheSizes>

</database>

<!-- web -->

<database id="web" singleInstance="true" type="Sitecore.Data.Database, Sitecore.Kernel">

<cacheSizes hint="setting">

<data>20MB</data>

<items>20MB</items>

<paths>4MB</paths>

<standardValues>4MB</standardValues>

</cacheSizes>

</database>

Some preliminary work has been done to preconfigure the cache levels for a site containing 100,000 items or more. You may need to tweak these numbers depending upon the number of items in your content tree.

**Custom Settings**

<settings>

<setting name="BucketTriggerCount" value="100"/>

**//**When an item has 100 Siblings, it will start to ask you to automatically bucket your content

<setting name="BucketTemplateId" value="{ADB6CA4F-03EF-4F47-B9AC-9CE2BA53FF97}" />

**//**If you would like to change the template of the item that organises all the hidden bucket items, then change this. For the first revision of this module, it is best to keep this as default.

<setting name="Indexing.UpdateInterval" value="00:00:30"/>

**//**This is the index update interval set for when items are created, deleted, modified etc. This is necessary once you start working with over 100,000 unstructured items in your index.

<setting name="SecuredItems" value="blur"/>

//This setting determines what happens to results that are returned and the user does not have access to Options include "hide" or "blur"

<setting name="LuceneQueryClauseCount" value="1024"/>

//This setting allows you to move the Clause count for Lucene up and down dependent upon how big you think the queries could grow.

</settings>

## Multiple Index Support

The item buckets supports having multiple indexes to support the content tree. A practical example would be that you may want to have a separate index for your content section, system section and media library. Having one index will satisfy most people’s requirements but for clients whom are expecting millions of content items, millions of media items etc, then this section is for you.

Add as many iterations of the following configuration to your Sitecore.ItemBuckets.Config file and change only the <Root> element, the <id> and the <folder> attribute. The example below shows that on top of the default buckets index (which should not be removed), you will now have the ItemBuckets manager context switch in-between indexes based off where within the content tree you are querying from.

It is important that the more “specific” your <Root> is, the higher it needs to be in the configuration as the Context Switcher will take the indexes in order. For example, if you have an index <Root> of **/sitecore/content/Home**, it should be located below the index for a <Root> of **/sitecore/content/Home/Flights**.

To be a candidate for the Index Context Switcher, you must preface your crawler declaration with <ItemBucketSearch> like shown below. This allows for you to run non bucket indexes and bucket indexes and Item Buckets will only use the ones that were intended for buckets.

<index id="systemfolder" type="Sitecore.Search.Index, Sitecore.Kernel">

<param desc="name">$(id)</param>

<param desc="folder">systemfolder</param>

<Analyzer ref="search/analyzer" />

<locations hint="list:AddCrawler">

<ItemBucketSearch type="Sitecore.ItemBucket.Kernel.Crawlers.CustomCrawler,Sitecore.ItemBucket.Kernel">

<Database>master</Database>

<Root>/sitecore/system</Root>

<IndexAllFields>true</IndexAllFields>

<fieldTypes hint="raw:AddFieldTypes">

<fieldType name="multilist" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

<fieldType name="treelist" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

<fieldType name="treelistex" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

<fieldType name="checklist" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

<fieldType name="tree list" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

</fieldTypes>

<include hint="list:ExcludeTemplate">

<layout>{ADB6CA4F-03EF-4F47-B9AC-9CE2BA53FF97}</layout>

</include>

</ItemBucketSearch>

</locations>

</index>

<index id="medialibrary" type="Sitecore.Search.Index, Sitecore.Kernel">

<param desc="name">$(id)</param>

<param desc="folder">medialibrary</param>

<Analyzer ref="search/analyzer" />

<locations hint="list:AddCrawler">

<ItemBucketSearch type="Sitecore.ItemBucket.Kernel.Crawlers.CustomCrawler,Sitecore.ItemBucket.Kernel">

<Database>master</Database>

<Root>/sitecore/media library</Root>

<IndexAllFields>true</IndexAllFields>

<fieldTypes hint="raw:AddFieldTypes">

<fieldType name="multilist" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

<fieldType name="treelist" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

<fieldType name="treelistex" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

<fieldType name="checklist" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

<fieldType name="tree list" storageType="NO" indexType="TOKENIZED" vectorType="NO" boost="1f" />

</fieldTypes>

<include hint="list:ExcludeTemplate">

<layout>{ADB6CA4F-03EF-4F47-B9AC-9CE2BA53FF97}</layout>

</include>

</ItemBucketSearch>

</locations>

</index>

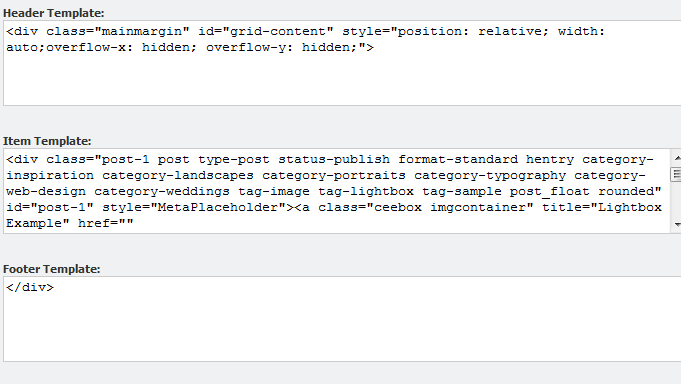
### Adding a new View

Developers can add new views to the search results to cater for different situations e.g. browsing an image gallery with the image of items. The default views are Grid and List. To add a new view, you need to

**1: Create a new View Item under /sitecore/system/Modules/Item Buckets/Views**

**2: Set the Header, Item and Footer Template (think Repeaters)**

**e.g.**



**You can use the following placeholder names to display the values of the items.**

**MetaPlaceholder - Style**

**LaunchTypePlaceholder – Whether it will launch the result in a new tab or a new content editor.**

**ItemIdPlaceholder – The Item ID**

**ImagePathPlaceholder – The Path to the Image of the Item**

**NamePlaceholder – The Name of the item**

**TemplatePlaceholder – The Template Name of the item**

**BucketPlaceholder - The Bucket that this result is part of**

**ContentPlaceholder – The Content of the result**

**VersionPlaceholder - The Version**

**CreatedPlaceholder -The Created Date**

**CreatedByPlaceholder – The Author**

# Sitecore Items with Big Data

This chapter describes how to enable the Big Data implementation of Item Buckets. Use this if you envisage storing millions of items, rebuilding indexes often, having remote indexing, remote querying and in memory indexes.

This chapter contains the following sections:

* Explanation of the Configuration Files
* In Memory Index
* Remote Index
* New Crawlers
* Query Server
* Configuring SOLR

## Explanation of the configuration files

This section explains the Sitecore.ItemBuckets.BigData configuration files for scaling Item Buckets.

### In Memory Index

The default Bucket Indexes will be written to file system and persisted across application pool resets. If you have read-only information within the content tree that will not change, you can have this stored within an in-memory index that is initialized at application startup. Depending on the size of this index, this will slow the application restart time of Sitecore but will mean that you have an index that is much faster than the file system based index.

**Advantages**

* **Fast**
* **No storage needed on the hard disk**

**Disadvantages**

* **Results in a slower application startup time**
* **No information persisted to disk**
* **Read-only**

<configuration type="Sitecore.ItemBuckets.BigData.RamDirectory.SearchConfiguration, Sitecore.ItemBuckets.BigData" singleInstance="true">

<indexes>

<index id="itembuckets\_buckets\_inmemory" type="Sitecore.ItemBuckets.BigData.RamDirectory.InMemoryIndex, Sitecore.ItemBuckets.BigData">

<param desc="name">$(id)</param>

<param desc="folder">itembuckets\_buckets\_inmemory</param>

<Analyzer ref="search/analyzer" />

<locations hint="list:AddCrawler">

<ItemSearch type="Sitecore.ItemBucket.BigData.Crawlers.RamCrawler,Sitecore.ItemBuckets.BigData">

<Database>master</Database>

<Root>/sitecore/content/Test Stub</Root>

<IndexAllFields>true</IndexAllFields>

</ItemSearch>

</locations>

</index>

</indexes>

</configuration>

### Remote Index

The default Bucket Indexes will be written to file system under the default data directory specified in the web.config file. Rebuilding of indexes can be quite slow and intensive on the web server. Hence you can have your indexes be rebuilt on a remote machine.

**Advantages**

* **Your remote machine can have an SSD drive on it which will make the indexing a lot faster.**
* **The hit on the server for re-indexing is done on the target machine, not the machine serving up the content.**

**Disadvantages**

* **There is a small time when the index will be locked for reads and writes while the index is copied back into the local data directory.**

Firstly, you need to set the RemoteIndexLocation setting to a network path with full read and write access.

<setting name="RemoteIndexLocation" value=""/>

You can then specify your index like so:

<configuration type="Sitecore.ItemBuckets.BigData.RamDirectory.SearchConfiguration, Sitecore.ItemBuckets.BigData" singleInstance="true">

<indexes>

<index id="itembuckets\_buckets\_remote" type="Sitecore.ItemBuckets.BigData.RemoteIndex.RemoteIndex, Sitecore.ItemBuckets.BigData">

<param desc="name">$(id)</param>

<param desc="folder">itembuckets\_buckets\_remote</param>

<Analyzer ref="search/analyzer" />

<locations hint="list:AddCrawler">

<ItemSearch type="Sitecore.ItemBucket.Kernel.Crawlers.CustomCrawler ,Sitecore.ItemBucket.Kernel">

<Database>master</Database>

<Root>/sitecore/content/Test Stub</Root>

<IndexAllFields>true</IndexAllFields>

</ItemSearch>

</locations>

</index>

</indexes>

</configuration>

### New Crawlers

For all other indexes, you can use the new CustomCrawler which can be heavily customized via configuration.

//The ExcludeTemplate allows you to specify templates that are not indexed

<include hint="list:ExcludeTemplate">

<layout>{ADB6CA4F-03EF-4F47-B9AC-9CE2BA53FF97}</layout>

</include>

//The IncludeTemplate is only needed if you turn “IndexAllFields” off

<include hint="list:IncludeTemplate">

<layout>{ADB6CA4F-03EF-4F47-B9AC-9CE2BA53FF97}</layout>

</include>

//The IncludeField is only needed if you turn “IndexAllFields” off

<include hint="list:IncludeField">

<fieldId>{8CDC337E-A112-42FB-BBB4-4143751E123F}</fieldId>

</include>

//The Exclude will remove inbuilt fields and custom fields from being indexed.

<include hint="list:ExcludeField">

<\_\_DefaultWorkflow>{CA9B9F52-4FB0-4F87-A79F-24DEA62CDA65}</\_\_DefaultWorkflow>

<\_\_Lock>{001DD393-96C5-490B-924A-B0F25CD9EFD8}</\_\_Lock>

<\_\_WorkflowState>{3E431DE1-525E-47A3-B6B0-1CCBEC3A8C98}</\_\_WorkflowState>

<\_\_LongDescription>{577F1689-7DE4-4AD2-A15F-7FDC1759285F}</\_\_LongDescription>

<\_\_Originator>{F6D8A61C-2F84-4401-BD24-52D2068172BC}</\_\_Originator>

<\_\_Owner>{52807595-0F8F-4B20-8D2A-CB71D28C6103}</\_\_Owner>

<\_\_ReadOnly>{9C6106EA-7A5A-48E2-8CAD-F0F693B1E2D4}</\_\_ReadOnly>

<\_\_Renderings>{F1A1FE9E-A60C-4DDB-A3A0-BB5B29FE732E}</\_\_Renderings>

<\_\_Revision>{8CDC337E-A112-42FB-BBB4-4143751E123F}</\_\_Revision>

<\_\_Security>{DEC8D2D5-E3CF-48B6-A653-8E69E2716641}</\_\_Security>

<\_\_ShortDescription>{9541E67D-CE8C-4225-803D-33F7F29F09EF}</\_\_ShortDescription>

<\_\_SortOrder>{BA3F86A2-4A1C-4D78-B63D-91C2779C1B5E}</\_\_SortOrder>

<\_\_UpdatedBy>{BADD9CF9-53E0-4D0C-BCC0-2D784C282F6A}</\_\_UpdatedBy>

<\_\_ValidFrom>{C8F93AFE-BFD4-4E8F-9C61-152559854661}</\_\_ValidFrom>

<\_\_Source>{1B86697D-60CA-4D80-83FB-7555A2E6CE1C}</\_\_Source>

<\_\_Workflow>{A4F985D9-98B3-4B52-AAAF-4344F6E747C6}</\_\_Workflow>

<\_\_Updated>{D9CF14B1-FA16-4BA6-9288-E8A174D4D522}</\_\_Updated>

<\_\_Created>{25BED78C-4957-4165-998A-CA1B52F67497}</\_\_Created>

</include>

//The Remove Special Fields will remove system fields from the index.

<fields hint="raw:RemoveSpecialFields">

<remove type="both">Links</remove>

<remove type="both">Editor</remove>

<remove type="both">Icon</remove>

<remove type="both">DisplayName</remove>

<remove type="both">AllTemplates</remove>

<remove type="both">Hidden</remove>

<remove type="both">Created</remove>

<remove type="both">Updated</remove>

</fields>

### Query Server

You can specify a dedicated query server where all bucket queries through the UI are run. This may be of use if you have many authors and you don’t want the overhead of the queries impacting the performance of the rest of the system.

By setting the “QueryServer” setting, this allows to route all queries to a dedicated authoring server. You can scale to many query servers if necessary with some authoring servers using QUERYSERVER 1 and some authoring servers using QUERYSERVER 2. Each individual server can specify a query server. Some of the helper methods e.g. *AutoSuggest*, will still run on the local server as these are very lightweight.

<setting name="QueryServer" value="http://query.authoring.site.com"/>

### Extending Support with SOLR

Solr includes powerful full-text search, hit highlighting, faceted search, dynamic clustering, database integration, rich document (e.g., Word, PDF) handling, and geospatial search. Solr is highly scalable, providing distributed search and index replication. You can extend the buckets implementation to work with SOLR if you require an even more scalable index and query power. If enabled, queries will be sent to SOLR to run and the History engine will call SOLR everytime an update, insertion or deletion is made to the index. You would consider using SOLR if you require an indexing server that can scale to many servers.

### Installing SOLR on Windows

This process will take you through how to install SOLR on Windows.

You will need to download the following software:

1. Download Xampp For Windows, Basic Package. <http://www.apachefriends.org/en/xampp-windows.html>

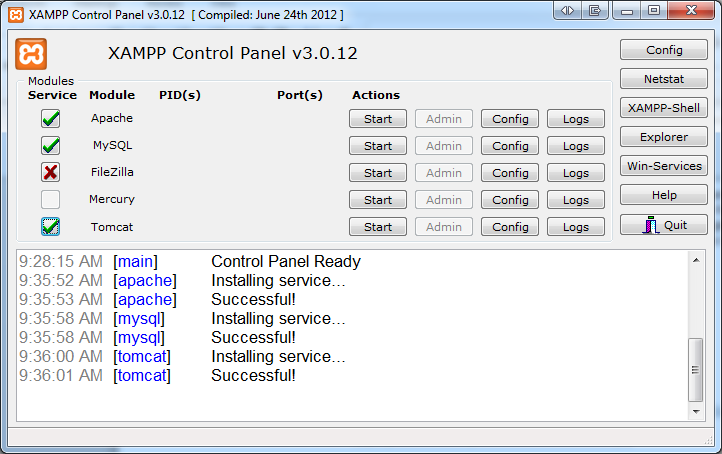
2. Download Java JDK <http://java.sun.com/javase/downloads/index.jsp>

3. Download *Apache Solr* from one of the mirrors (no current support for SOLR 4.0). <http://www.proxytracker.com/apache/lucene/solr/>

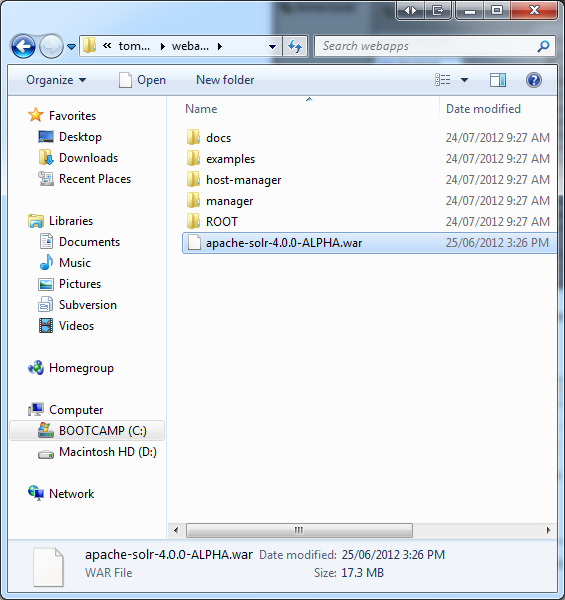
4. Download the Solr PHP Client. <http://code.google.com/p/solr-php-client/>

**Step 1:** Install all the software above with the defaults.

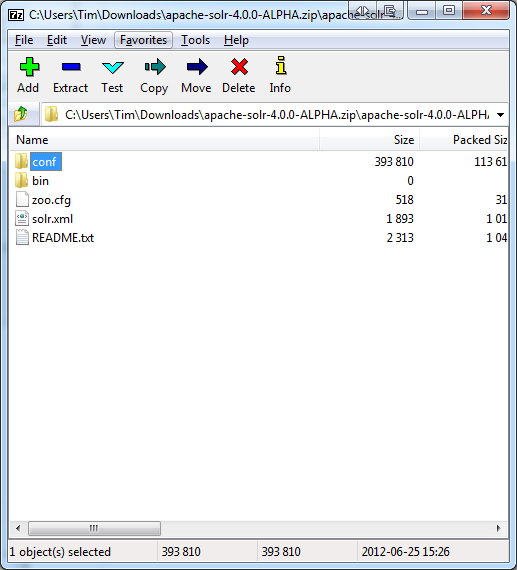
**Step 2:** Run the Xampp and enable the Apache, MySQL and Tomcat serives.



**Step 3:** Copy the \*.war file into the c:\xammp\tomcat\webapps folder from you apache-solr download (make sure the Apache Tomcat Service is not currently running)



**Step 4:** Create a directory under c:\xammp called “solr” and copy the entire examples/solr directory in you apache-solr.zip to this directory.



**Step 5:** Create a folder on the C:\ called “solr”. Copy the etc, lib, logs, solr, webapps, and start.jar folders from you apache-solr.zip to C:\solr.  Now open the C:\solr\solr folder and copy the contents back to the root C:\solr folder.

**Step 6:** Open Powershell or CMD.exe and navigate to your “c:\solr” directory and run (you will have to restart this every time the server resets or SOLR crashes):

***java -Dsolr.solr.home=c:/solr/ -jar start.jar***

**Step 7:** Open Sitecore and install the Sitecore.ItemBucket.BigData.SOLR.zip through the installation wizard. Once it is finished you should have the SOLR Dashboard in the Start Menu. You will need to go into the “core” database and look for the item “sitecore/content/applications/SOLR. Modify the Application to be the URL of your site e.g. http://<sitename>:8983/solr/

**Step 8:** Navigate to C:\Users\<your user>\AppData\Local\Temp\jetty-0.0.0.0-8983-solr.war-\_solr-any-\webapp and open the admin.html file in a text editor. Add ***style=”background-color:white;”*** to the <body> tag.

**Step 9:** Open the SOLR application in the start menu.

**Step 10:** Now all updates to your local filesystem index will also go the SOLR index as well. All queries will now be redirected to use the SOLR server instead of the local index. You can always have some application pools use the local and some use SOLR.

### Replication of Index Across Servers

If you find that one indexing and query server is not enough, you can introduce new slave servers into your environment. This section will discuss one possible architecture for a master/slave environment. The Sitecore.ItemBuckets.SOLR package will have included sample configs that you can use to setup distributed searching. Once you have setup the configuration below, restart SOLR to see new options in the admin screen through the Sitecore desktop.

### Master Server Configuration

Take the solrconfig.xml file that sits in your conf directory and insert your server details

<requestHandler name="/replication" class="solr.ReplicationHandler" >

<lst name="master">

<str name="replicateAfter">commit</str>

<str name="replicateAfter">startup</str>

<str name="replicateAfter">optimize</str>

<str name="confFiles">schema.xml,stopwords.txt,elevate.xml</str>

<str name="confFiles">solrconfig\_slave.xml:solrconfig.xml,x.xml,y.xml</str>

<str name="commitReserveDuration">00:00:10</str>

</lst>

</requestHandler>

### Slave Server Configuration

After you have installed SOLR on another server on your network, open the solrconfig.xml and place this into the file.

<requestHandler name="/replication" class="solr.ReplicationHandler" >

<lst name="slave">

<str name="masterUrl">http://remote\_host:port/solr/corename/replication</str>

<str name="pollInterval">00:00:20</str>

<str name="httpBasicAuthUser">username</str>

<str name="httpBasicAuthPassword">password</str>

</lst>

</requestHandler>

### Repeater Server Configuration

Now that you have a Master and potentially many slave servers, you may need to set up a server that takes load off the master and acts as both a master and a slave server.

<requestHandler name="/replication" class="solr.ReplicationHandler">

<lst name="master">

<str name="replicateAfter">commit</str>

<str name="confFiles">schema.xml,stopwords.txt,synonyms.txt</str>

</lst>

<lst name="slave">

<str name="masterUrl">http://master.solr.company.com:8983/solr/replication</str>

<str name="pollInterval">00:00:60</str>

</lst>

</requestHandler>

### Running SOLR on startup.

You must have the SOLR service running all the time to be able to query. To run SOLR on startup of a machine you can use this batch script:

***cd c:\solr***

***java -Dsolr.solr.home=c:\solr -jar c:\solr\start.jar***

***pause***

Create a shortcut to this file and then place the shortcut in

***C:\Users\<UserName>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup***

### Generate A Schema.xml file.

You will need to create a schema.xml file for SOLR to know which fields are in your index and what type they are. You can have Sitecore generate this for you from the Control Panel. Select “Generate a SOLR schema file and then choose which indexes you will want this done for.

### Extending Support with Hadoop Clusters

Currently no support for Hadoop Clusters. This is intended for a future release.

# Appendix

This chapter describes how various internal processes work and contains information that will help you to extend or modify the module.

This chapter contains the following sections:

* Tips and Tricks

## Tips and Tricks

**Standard Web.Config Tweaks**

* Change the web.config setting of Indexing.UpdateInterval to 30 seconds or lower depending upon performance.
* You should periodically tweak the cach­e dependent upon how many items are in the content tree and how many similar searches that are processed.

**Setup Tweaks**

* Keep the number of facets to a minimum (any more than 100 and you will see a small degradation of performance.)
* When importing a lot of content programmatically, you will need to truncate the PublishingQueue, History and Event Queue table for Master and Web and will need to run a rebuild of the indexes on the Database tables otherwise your Sitecore site will most likely not start. This is due to the fact that you will have huge PublishingQueue, History and EventQueue tables which will take a long time to process. Clearing them simply means that you will need to rebuild your index and run a smart publish instead of an incremental.

**Author Tips**

* You can search for IDs by wildcards. For example, if you know the first 4 characters of a guid for an item then you can simply type id:c728\* and click search and it will find the corresponding items.

**Environment Tweaks**

* If possible, disable the inbuilt Windows Search Index an any other indexer running on the machine that will be running the index or the web server itself. This is use essential Disk I/O that Lucene.net uses a lot of.
* Don’t run processes on your index to backup as it will be an expensive operation and the index will most likely be out of date by the time the backup runs.
* It is very important that you have a SQL Maintenance task that rebuilds your indexes. Once you create a lot of content, the index fragmentation will increase, especially with bulk import of content. The hotspots will be the Items, Versioned, Unversioned, Shared, Blobs and Links table. It is best to set rebuilds for all tables just to be safe. If you don’t do this, you will find that the CMS gets sluggish. Here is a script for rebuilding all indexes on your databases.

-- Show fragmentation for all tables

EXEC sp\_MSforeachtable @command1="print '?' DBCC SHOWCONTIG('?')"

--Rebuild all indexes (note this method locks the tables while the indexes are being rebuilt)

USE [Sitecore\_Master] --Change this to your database name

DECLARE @TableName varchar(255)

DECLARE TableCursor CURSOR FOR

SELECT table\_name FROM information\_schema.tables

WHERE table\_type = 'base table'

OPEN TableCursor

FETCH NEXT FROM TableCursor INTO @TableName

WHILE @@FETCH\_STATUS = 0

BEGIN

DBCC DBREINDEX(@TableName,' ',90)

FETCH NEXT FROM TableCursor INTO @TableName

END

CLOSE TableCursor

DEALLOCATE TableCursor

**Importing Data Tweaks**

* When importing a lot of content into Sitecore, it is best to do so using the BulkUpdateContext class and once the import is done, it is best to rebuild the lucene index.
* If importing lots of content, do it in batches of e.g. 1000 and then bucket or re-sync that bucket to not overload the process with items.

**Bucket Config Tweaks**

* You can tweak your index to not index certain things that you don’t want in the index. Doing this will decrease rebuild time and will decrease search time.
* Consider rebuilding your indexes on a machine that has a Solid State Disk. Incremental updates do not have to be on SSD but will benefit from this as well. If you have one dedicated server that rebuilds indexes and then deploys them to an environment then have that one machine have a SSD. Indexes will not be so big, so a small SSD is suffice (e.g. 64gb).
* Don’t shard too many indexes. Item Buckets will have to context switch in between them which will slow down search time.
* If you have very large caches then you will see large memory spikes when you run a search. This is normal as a search is filling the ItemCache for the results. Be careful with under-optimised caches as it will keep as much of the search results in cache and may not be optimal.
* If you see lag in searches or results that are taking a long time to facet then enable Debug mode in the config and this will Log all the queries, how long they take to run and how many clauses they contain. This can help you find your issue. Wildcard and RangeQueries will be the main culprits.
* Optimise the out-of-the-box indexes. This has been designed for it and will speed up Index Rebuilding time and to some small degree, query time as well.
* Disable the search Tips if you don’t need them. This can be done in config.
* Disable all the dropdowns in the **/sitecore/system/Modules/Item Buckets/Settings/Search Box Dropdown** that you are not using. The recently modified and recently created will be the most expensive of the lookups.
* Add all the items under /Sitecore/System/Modules/ItemBuckets to your prefetch cache.
* If you have disabled the Debug Mode but would like to run a debug on a single query, simply add Debug:1 and press tab to any search. It will write just that search query to your log file.
* Double check that Hidden Items is turned **OFF**.

**Scaling Tweaks**

* Check the BigData DLL to use Indexes for huge amounts of data and also look at the InMemory Index which can hold parts of your content tree in an Index in RAM. This will disappear when the application stops but will be their when the application starts again. The InMemoryIndex is much quicker but does not store data.

## Known Issues

* The functionality of copy/move/transfer in the context menu is not supported for Item Buckets (only in IE)
* When trying to bucket or add a search tab to a protected item, you will need to refresh the item to see the tab.
* When marking a template as bucketable you will need to click the item again to load it and see the change in the ribbon.
* By default, Sitecore will not search for the following words :

"a", "an", "and", "are", "as", "at", "be", "but", "by", "for", "if", "in", "into", "is", "it", "no", "not", "of", "on", "or", "such", "that", "the", "their", "then", "there", "these", "they", "this", "to", "was", "will", "with".

* When you open items in a new tab from the search results, you will be given a Save and Close button and a Close button in the Home tab. These do not do anything.
* By default, Sitecore will index HTML tags in your fields and these tags will be searchable and appear in your search results. For example, if you have a <strong> tag in your HTML fields, and search for “strong” then it will show up in the results.
* In FireFox, when closing an item tab, it will not select any of the other tabs.
* When using the author filter in the search you will need to type the domain that the user is in first to retrieve the list of authors.
* If your item does not have a version in any language then it will not show in the search results.
* If you have multiple languages then the results will always bring back the latest version of the versions on that item despite what language is the context language. If you would like to search a specific language then you will need to use the “language:” filter when searching through the UI.
* When using the Search button within the Context Menu to add a search tab, you will need to click the item again to see the tab.
* In the Bucketlist FieldType if the user clicks to go to the next page of results, it will try to fetch them even if there are no items to fetch. This simply means that the user will get an empty list.
* If you don't put the config in the right directory, the buttons will all still show up but will do nothing.
* Field Filter on Bucket List doesn't work.
* Grid View in Bucket Link and RTE has styling issues.
* UI is designed mainly for Firefox and Chrome and there are some styling issues in IE.