



Sitecore CMS 6.4 - 6.6

Item Buckets Administrator and Developer's Guide

A Quick Start Guide and Configuration Reference for Administrators and Developers

Table of Contents

Chapter 1	Introduction	4
1.1	Introduction	5
1.1.1	Sitecore Versions that Support Item Buckets	5
1.1.2	Item Buckets and Sitecore Customer Support.....	5
1.2	Fundamental Concepts	6
Chapter 2	Installation and Configuration Guide	8
2.1	Installation.....	9
2.1.1	Requirements.....	9
2.1.2	Installing the Item Buckets Module.....	9
	Installing Sitecore Item Buckets Module from a NuGet Package	9
	Installing the Sitecore Item Buckets Module from Sitecore Packages.....	9
2.2	Setting up Item Buckets.....	11
2.2.1	Defining an Item Bucket.....	11
2.2.2	Making a Template Bucketable.....	12
2.3	Configuring Search for Item Buckets.....	14
2.3.1	Specifying a Facet for Filtering Search Results	14
2.3.2	Specifying which Fields are Displayed in the Search Results	15
2.3.3	Specifying a Search Result Image and Search Result Text.....	15
2.3.4	Displaying Media Library Images in Search Results.....	16
2.4	Working with Item Buckets	17
2.4.1	Creating a Content Item in an Item Bucket.....	17
2.4.2	Deleting a Content Item from an Item Bucket.....	17
2.4.3	Converting an Item Bucket into a Normal Container.....	18
2.4.4	Synchronizing an Item Bucket.....	18
2.4.5	Locking Parent/Child Relationships.....	19
2.4.6	Managing Item Buckets from the Control Panel	20
2.4.7	Item Bucket Search Settings.....	20
2.4.8	Link Database	21
Chapter 3	Searching.....	22
3.1	Searching and Item Buckets.....	23
3.1.1	Viewing the Search Results.....	23
	Enabling Other Views.....	23
3.1.2	Using Facets to Refine your Search.....	23
	Troubleshooting	25
3.1.3	Language Search	25
3.1.4	Complex Searches	25
	Range Searches	26
	Combining, AND, OR, and NOT	26
	Search Tips.....	26
3.1.5	Opening Items in the Search Results.....	27
3.2	Using Search Filters	29
3.2.1	Auto-Organising	31
3.2.2	Paging Results	31
3.2.3	Search Helpers.....	32
3.2.4	Security and Item Buckets	33
	Locking	33
3.2.5	IA Modifications.....	33
	Keyboard Shortcuts.....	34
3.2.6	Query the Index from any Inbuilt Field that Supports Lists.....	35
	Using a Custom Class to Query the Index	35
3.2.7	Using the New Field Types.....	35
3.2.8	Using Item Buckets with the Data Source of a Control	37
	Tips	37
	Searching for Images	39

3.3	Page Editor and DMS Support.....	40
3.3.1	Personalization and MV Tests	40
3.3.2	Setting the Data Source.....	41
3.4	Inserting and Managing Links.....	42
3.4.1	Inserting a Link.....	42
	Inserting a Link in the Rich Text Editor	42
	Inserting a Bucket Link.....	44
3.4.2	Link Providers	45
3.4.3	Item Resolvers	45
3.4.4	Tagging Associations Across Many Items	45
Chapter 4	Developing with Item Buckets.....	46
4.1	Buckets API	47
4.1.1	Query Examples.....	48
	Using the Bucket Manager	48
	Using the Item Extension Methods	49
	Using Linq Notation with a BucketQuery Collection	50
4.2	Working with New Code	51
4.3	Working with Extension Points.....	52
4.3.1	Working with Existing Code	53
4.3.2	Common Requirements.....	55
4.3.3	Creating a Tag Repository	56
4.3.4	Creating a Facet.....	57
4.3.5	Configuration Files.....	58
4.4	Multiple Index Support.....	60
4.4.1	Adding a New View	61
Chapter 5	Sitecore Items and Big Data	63
5.1	The Configuration Files	64
5.1.1	In-Memory Indexes.....	64
5.1.2	Remote Index.....	64
5.1.3	New Crawlers.....	65
5.1.4	Query Server.....	66
5.1.5	Extending Support with SOLR	66
5.1.6	Installing SOLR on Windows	66
5.1.7	Replicating the Index Across Servers.....	68
	Master Server Configuration.....	69
5.1.8	Slave Server Configuration.....	69
5.1.9	Repeater Server Configuration	69
5.1.10	Running SOLR on Startup	69
5.1.11	Generate a Schema.xml file.....	70
5.1.12	Extending Support with Hadoop Clusters	70
Chapter 6	Appendix.....	71
6.1	Tips and Tricks.....	72
6.2	Known Issues.....	74

Chapter 1

Introduction

This document is designed for Sitecore administrators and developers and describes how to set up, configure, and tune the Sitecore Item Buckets module.

The document contains the following chapters:

- **Chapter 1 — Introduction**
The introduction and the fundamental concepts behind 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 lets you use the module.
- **Chapter 3 — Searching**
This chapter contains practical advice on configuring and using the module from an administrator's perspective.
- **Chapter 4 — Developing with Item Buckets**
This chapter describes the configuration settings and how to use the API.
- **Chapter 5 — Sitecore Items and Big Data**
How to use scaling features with the Item Buckets module.
- **Chapter 6 — Appendix**
This chapter contains some tips and tricks as well as a list of known issues.

1.1 Introduction

Before you install the Item Buckets module, you should determine whether or not this module is the right solution for you.

The Item Bucket module lets you specify that specific items in the content tree should be defined as item buckets that can contain any number of subitems. These subitems are not listed in the content tree and do not have a parent-child relationship with the bucket item.

You can search in each bucket item to find the items that you are interested in.

The Item Buckets module lets content authors:

- Hide items in the content tree
- Use the new search functionality to retrieve content items from buckets.
- Use search functionality to set the value of fields in content items.
- Alter the parent-child relationship for content items. This can cause your code to not work as desired.

For more information, see the section *Developing with Item Buckets*.

You don't have to use the item buckets functionality after you install the module. The buckets system only starts to work when you create the first bucket.

You can create a hybrid structure that consists of content items that are stored in buckets and content items that are structured in the normal way.

You can also define a sub-structure within an item bucket.

1.1.1 Sitecore Versions that Support Item Buckets

The following table lists the versions of Sitecore that support item buckets.

Sitecore Version	.NET Version	Supports Item Buckets
Sitecore 6.6	.NET 4.0	Yes
Sitecore 6.6	.NET 2.0	Yes
Sitecore 6.5	.NET 4.0	Yes
Sitecore 6.5	.NET 2.0	Yes
Sitecore 6.4	.NET 4.0	Yes
Sitecore 6.4	.NET 2.0	Yes
Sitecore 6.3	.NET 2.0	No
Sitecore 6.2	.NET 2.0	No

1.1.2 Item Buckets and Sitecore Customer Support

The Sitecore Item Buckets module is currently distributed as a shared source module.

Sitecore customer Support only provides assistance for customers who have problems with shared source modules if they purchase the *Professional Shared Source Support for all Support Enabled modules* license granule.

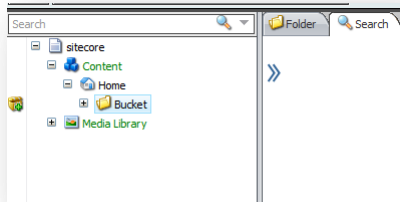
This license granule covers all the shared source modules that have the *Supported by Sitecore* stamp: <http://trac.sitecore.net/Index/wiki/Better#no1>.

1.2 Fundamental Concepts

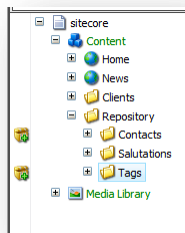
This section explains some of the concepts used in this document.

Bucket

A bucket is a container in the content tree that stores content items. The difference between this and a normal container is that all items stored in this container are hidden and this container has a new Search tab that lets you look up the items in the container. You no longer have to use the content tree to navigate and select items. In addition to this, items in a bucket are automatically organized into folders and the parent to child relationship between items is completely removed.



All the buckets are marked with an icon in the gutter. This allows content authors to see which containers are buckets and which are normal containers. Each individual user can enable this gutter warning by right-clicking the gutter to the left of the content tree and then selecting **Bucket Container**.



Defining items as bucket has many advantages including:

- You can search for content items in each bucket. You can even search for non-bucket items.
A non-bucket item is a content item that is based on a template that doesn't support item buckets.
- You can use the new Bucket API with these items.
- All the items are now automatically organized in a logical format.
- Items that are stored under other items can act as embedded items.
- A single repository can store millions of items without slowing down the UI or congesting the content tree.

Hidden Items (Setup)

Items that are stored in a bucket are not hidden in the content tree by default because the **Hidden Item** checkbox is selected.

To hide the content items that are stored in an item bucket, in the **Content Editor**, on the **View** tab, in the **View** group, clear the **Hidden Items** check box.

We recommend that you clear the **Hidden Items** check box if you are using the Item Buckets module as it will prevent the system from unnecessarily loading the items in the content tree. Developers can work with hidden items if they need to.

Why use a Bucket?

The *Item Buckets* module addresses the problem of managing large numbers of items within the content tree, retrieving them, and working with them in a speedy and efficient manner. To decide if you should turn an item into a bucket, and in-turn, hide all its descendants, you must ask yourself if you care about the structure of the data that lives in the bucket. For example, if you have a *Product Repository*, a *Movie Repository*, or a *Tag Repository* within the content tree, you would most likely want to just dump all the items into the appropriate folder and when you want to work with a particular product, movie, or tag, you simply search for it and open it.

In the Item Buckets module connections are made through semantics, not through hierarchy. For example, traditionally for a selection of products, you would create categories in the content tree and place the individual product items in these categories. With item buckets, you can place all the products in one repository and tag each product with the category that it belongs to.

Chapter 2

Installation and Configuration Guide

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

This chapter contains the following sections:

- Installation
- Setup
- Configuration

2.1 Installation

This section describes the installation process.

2.1.1 Requirements

The following are required to install this module.

Before you install the Item Buckets module:

- Your application pool must use the .NET 4.0 runtime.
- You must have at least 4 GB of RAM. We recommend 8 GB of RAM.
- We recommend that you use an SSD Hard Drive because Lucene.net is very I/O intensive.

2.1.2 Installing the Item Buckets Module

You can install the Item Buckets module from two different packages:

- A NuGet package
- A Sitecore package

Installing Sitecore Item Buckets Module from a NuGet Package

To install the Item Buckets Module from a NuGet package:

In Visual Studio or on <http://nuget.org>, search for Sitecore Item Buckets.

This should give you three results:

- Sitecore Item Buckets — the kernel. You must attach the kernel to every project that uses the Item Buckets Search API.
- Sitecore Item Buckets Client User Interface — the UI. You must attach the UI package to every project that requires the Item Buckets Search UI, for example, your Sitecore website project.
- Sitecore Big Data — an optional package that you can attach to projects that can contain millions of content items in the content tree. This package also contains other features, such as in-memory indexes and remote-indexes.

After you install the NuGet package, Visual Studio notifies you every time a new update is available. You can then decide to install the update or continue using your current version.

Installing the Sitecore Item Buckets Module from Sitecore Packages

The Sitecore Item Buckets Module comes as three separate Sitecore packages:

- Sitecore.ItemBuckets.Kernel — the kernel. You must install this package in every environment that uses item buckets.
- Sitecore.ItemBuckets.UI — the user interface. You only need to install this package in the environments that need the UI, for example, the authoring and development environments.
- Sitecore.BigData — an optional package that you can attach to projects that can contain millions of content items in the content tree. You can install this package alongside the Sitecore Item Buckets kernel.

After you download the Sitecore packages, you should use the Sitecore Install Package Wizard to install them on your Sitecore instance. The wizard will ask you to override some content items. Please

select *Override* or *Merge* if you have already made customizations to the template that is being modified.

The installation also triggers some post installation steps. It runs a Smart Publish and rebuilds the new index that is installed for you.

You must restart the client to see your changes.

Warning

After you install the module, it needs to rebuild/build indexes and it may take some time before you can start using the module. For example, if you immediately run a search and don't see all your facets or results, this is because the index has not finished building. You can check your log files to see when the index is completely rebuilt.

The Item Update Packages

After you install the Sitecore packages or the NuGet packages that contain the Item Buckets module, you must install a set of Sitecore update packages:

- Sitecore.ItemBucket.BigData.Solr.update
- Sitecore.ItemBucket.Core.update
- Sitecore.ItemBucket.Master.update

For more information about installing Sitecore update packages, see the manual *Update Installation Wizard*.

2.2 Setting up Item Buckets

This section describes the how to set up and work with item buckets.

You can create an item bucket from a new item or convert an item that already exists into an item bucket.

If you convert an item that already exists into an item bucket, the item bucket will organize and hide all of its descendants. If the parent item contains 1000's or even millions of items, it can take some time to organize them after you convert it into an item bucket. A progress bar appears that displays a running tally of the items being processed.

Note

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

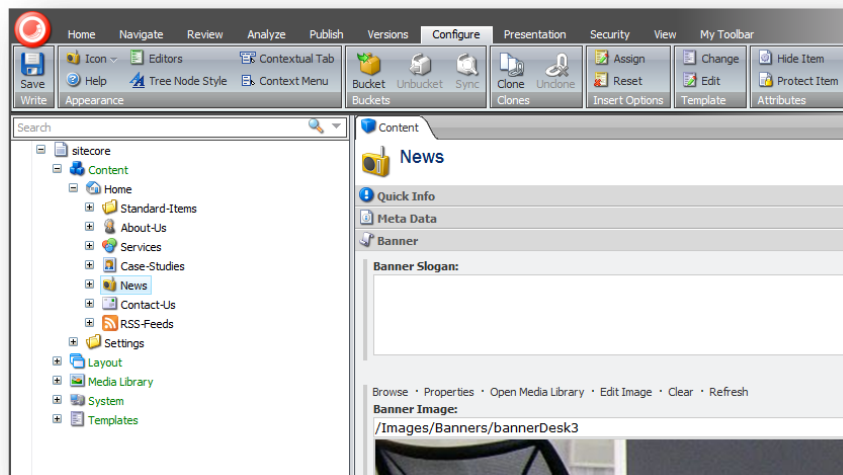
2.2.1 Defining an Item Bucket

You can create an item bucket from a new item or convert an item that already exists into an item bucket.

To define an item bucket:

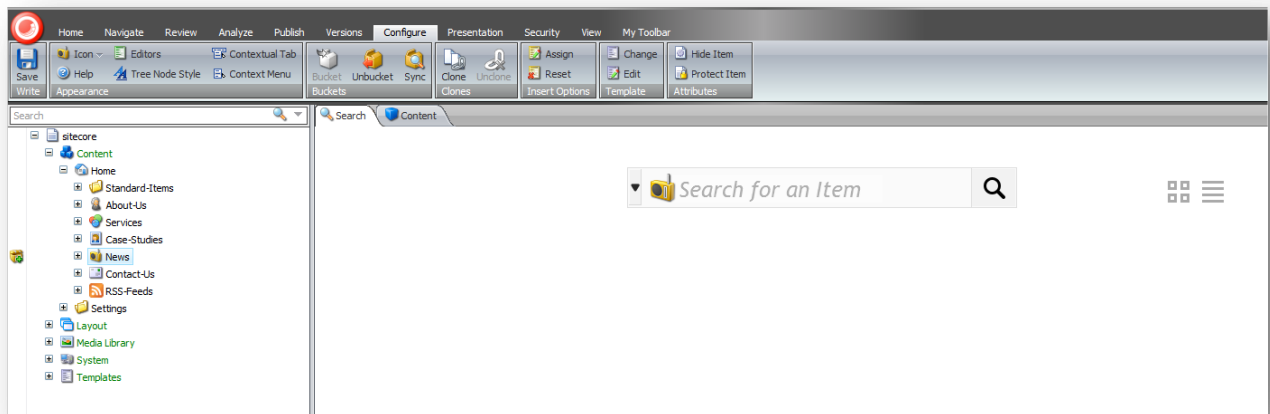
1. In the **Content Editor**, in the content tree, create an item, for example a folder, and give it a suitable name.

Alternatively, select an existing content item that can expand over time to contain countless sub-items.



2. In the content tree, select the content item and then on the **Home** tab, click **Edit** to lock the item.
3. Click the **Configure** tab and then in the **Buckets** group, click **Bucket** to convert the new item into an item bucket.

When you convert a content item into an item bucket, a new **Search** tab appears in the right-hand pane.



You use this tab to search for content items in the item bucket.

Furthermore, the bucket icon appears in the left-hand gutter informing you that this item is an item bucket.

For more information about searching in item buckets, see the section *Searching*

2.2.2 Making a Template Bucketable

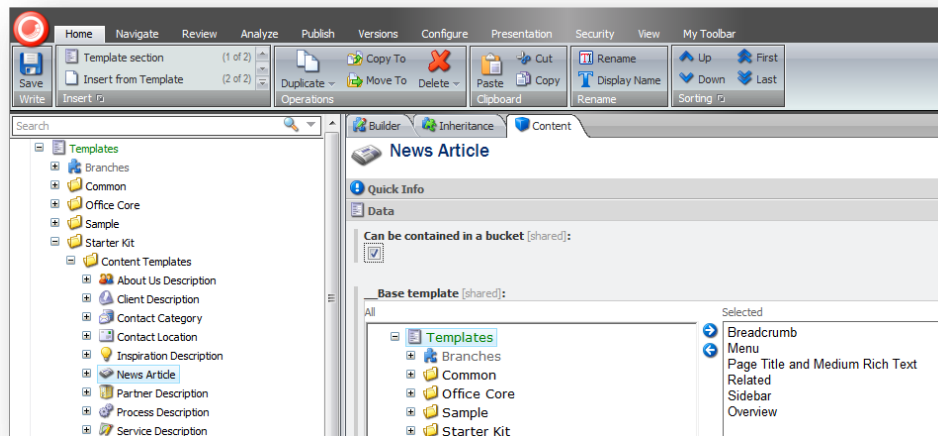
When you set up item buckets, you must ensure that the content items that you want to store in the item bucket are based on templates that can be stored in an item bucket. Content items based on these templates are both hidden and searchable when they are placed in an item bucket.

If the content items are based on a template that cannot be stored in an item bucket, the system will not automatically structure the content items for you. Instead, the content items are treated like normal items in the content tree.

To make content items bucketable:

1. In the **Content Editor**, select one of the content items that you want to make bucketable.
2. In the right-hand pane, on the **Content** tab, expand the **Quick Info** section and click the template link and the template that this content item is based on opens in the **Template Manager**.

3. In the **Template Manager**, in the right hand pane, click the **Content** tab.



4. In the **Data** section, select the **Can be contained in a bucket** check mark.

All the content items that are based on this template are now hidden when they are stored in an item bucket. They are also searchable when you search in an item bucket.

2.3 Configuring Search for Item Buckets

After you have created an item bucket and specified which items can be stored in it, you can configure how search will work with each type of bucketable content item.

You can specify which:

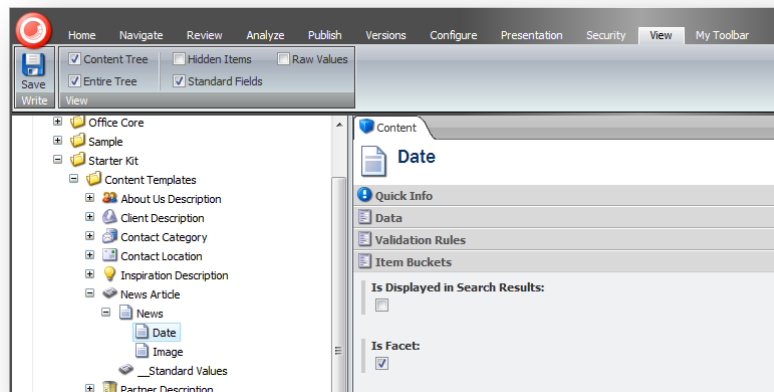
- Facets you can use to filter your search results.
- Fields are displayed in the search results.
- Image is displayed with each content item in the search results.
- Text is displayed with each content item in the search results.

2.3.1 Specifying a Facet for Filtering Search Results

When you search in an item bucket, you might like to filter your search results based on categories. To filter search result by category, you must mark the appropriate template fields as facets.

To mark a template field as a facet:

1. Open the template in the **Template Manager**.
2. Click the **View** tab and then in the **View** group, select the **Standard Fields** check box.
3. In the content tree, expand the template and select the field that you want to define as a facet or category.
4. On the **Content** tab, scroll down to the **Item Buckets** section.



5. In the **Item Buckets** section, select the **Is Facet** check box.

Now you can use this field to filter your search results.

Facets support the following field types:

- Single-Line-Text
- Multi-Link-Text
- Rich Text
- Bucket Rich Text

Once you have set this, the filter should show up in the list of facets that appear on the Item Bucket search tab and you can use them to filter your searches.

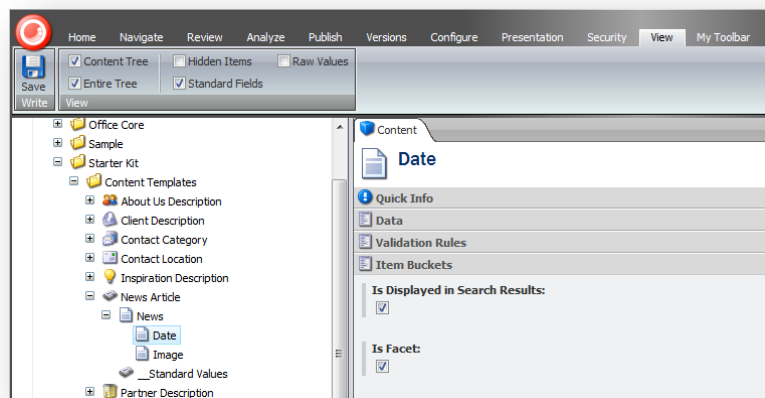
For more information about searching and filtering, see the section *Searching*.

2.3.2 Specifying which Fields are Displayed in the Search Results

You can specify which fields are displayed in the search results when you search an item bucket.

To specify that a field should be displayed in the search result:

1. Open the template in the **Template Manager**.
2. Click the **View** tab and then in the **View** group, select the **Standard Fields** check box.
3. In the content tree, expand the template and select the field that you want to be displayed in the search results.
4. On the **Content** tab, scroll down to the **Item Buckets** section.



5. In the **Item Buckets** section, select the **Is Displayed in Search Results** check box.

2.3.3 Specifying a Search Result Image and Search Result Text

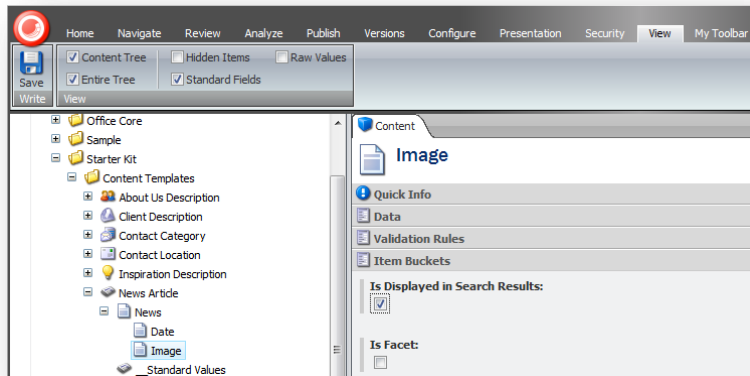
When you search in an item bucket, Sitecore displays an image with each content item that it displays in the search results.

You can specify which of the image fields in the template should be shown in the search results.

To specify which image field should be displayed in the search result:

1. Open the template in the **Template Manager**.
2. Click the **View** tab and then in the **View** group, select the **Standard Fields** check box.
3. In the content tree, navigate to the template and expand it.
4. Select the image field that you want to appear in the search results.

5. On the **Content** tab, scroll down to the **Item Buckets** section.



6. In the **Item Buckets** section, select the **Is Displayed in Search Results** check box.

The image that is displayed in this field will now be shown in the search results. You can also use this check box to specify which text field should be displayed in the search results.

2.3.4 Displaying Media Library Images in Search Results.

The entire content tree can work with item buckets, including the Media Library. If you need to search for media items and want the images to appear in the search results, you must select the **Is Displayed in Search Results** for the following two blob fields:

- /sitecore/templates/System/Media/Versioned/File/Media/Blob
- /sitecore/templates/System/Media/Unversioned/File/Media/Blob

2.4 Working with Item Buckets

Items that are stored in item buckets are just like any other content items — you can create, edit, and delete them.

2.4.1 Creating a Content Item in an Item Bucket

You create a content item in an item bucket in exactly the way as you would create an ordinary structured content item.

To create a content item in an item bucket:

1. In the **Content Editor**, in the content tree, select the item bucket that you want to create an item in.
2. On the **Home** tab, in the **Insert** group, click the type of item that you want to create and the content item is created in the item bucket.

If the template that this content item is based on is bucketable, the item will be hidden in the item bucket and you cannot see it.

If the template that this content item is based on is not bucketable, the item will be visible in the item bucket.

This means that you can easily have an item bucket that is a hybrid of structured and unstructured content items.

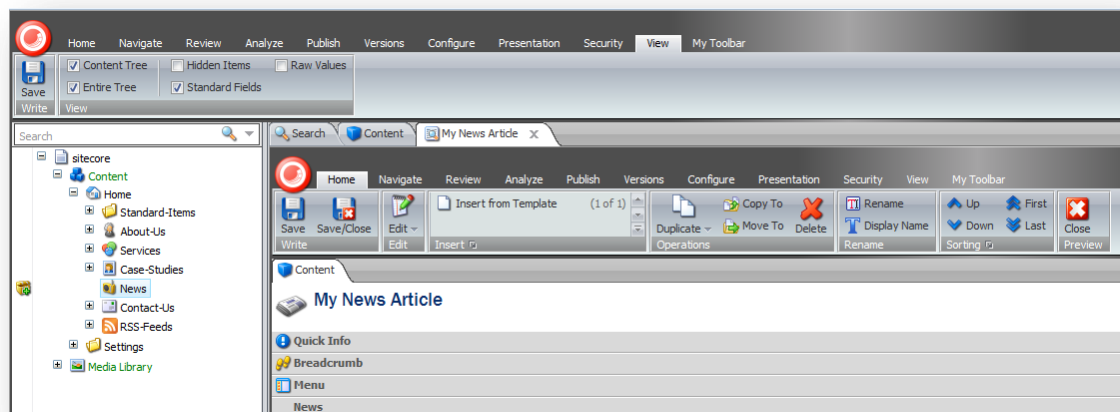
For more information about making a content item bucketable, see the section *Making a Template Bucketable*.

2.4.2 Deleting a Content Item from an Item Bucket

As you work with your website, you will often need to delete a content item from an item bucket.

To delete a content item from an item bucket:

1. In the **Content Editor**, in the content tree, select the item bucket that you want to delete an item from.
2. In the **Search** tab, in the search field, enter a search term that will help identify the content item.
3. In the search results, click the link to the content item that you want to delete and it is displayed in a new tab.



4. In the tab that displays the content item, on the **Home** tab, in the **Operations** group, click **Delete** and the item is deleted.

2.4.3 Converting an Item Bucket into a Normal Container

After you have implemented an item bucket and stored some bucketable items in it, you may decide that this is not the correct structure for these items and need to roll it back.

Important

The process of rolling back an item bucket can be quite complicated and must be performed in the correct order.

To convert an item bucket into a normal container:

1. In the **Content Editor**, navigate to the item bucket that you want to convert into a normal container.
2. On the **View** tab, in the **View** group, select the **Hidden Items** check box.
3. Refresh the content tree.
4. Expand the item bucket and all of the items that are stored in the item bucket appear in the content tree.
5. Make a list of the templates that the different items stored in the item bucket are based on.
6. In the **Template Manager**, select each template and clear the **Can be stored in a bucket** check box.
7. In the **Content Editor**, in the content tree, select the item bucket that you want to convert into a normal container.
8. On the **Configure** tab, in the **Buckets** group, click **Unbucket**.

The item bucket reverts back to being a normal container and all of the sub-items are displayed as structured items.

If you have lots of items stored in one bucket this may take some time. This process runs on a separate thread. If you need to continue on with other work, simply minimize the pop-up box and carry on working.

Warning

If you convert an item bucket into a normal container without ensuring that the items it contains are no longer bucketable, the content items disappear and cannot be recovered.

2.4.4 Synchronizing an Item Bucket

When you create an item bucket, you can store both structured and unstructured content items in it. If you later decide to convert some of the structured items into unstructured items, you must make the templates that they are based on bucketable.

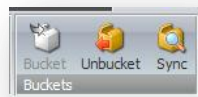
For more information about making templates bucketable, see the section *Making a Template Bucketable*.

After you have made the templates bucketable, you must update the structure of the item bucket.

To update the structure of the item bucket:

1. In the **Content Editor**, navigate to the item bucket whose structure you want to update.

- On the **Configure** tab, in the **Buckets** group, click **Sync**.



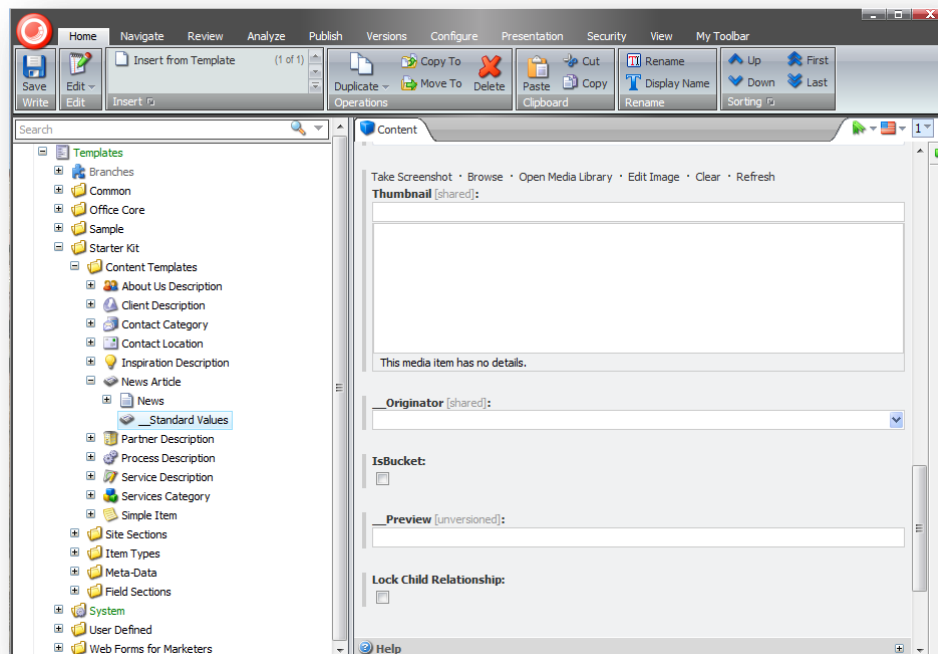
Sitecore analyzes the item bucket to see which content items should be hidden and which should be displayed as structured items.

2.4.5 Locking Parent/Child Relationships

In some cases, you may want to lock the relationship between a parent item that is bucketable and its child items to ensure that they the child items are always stored below the parent item. For example, you might want to lock the parent to child relationship between news articles and comments.

To lock the parent to child relationship:

- In the **Template Manager**, navigate to the template for the parent item. In this case it would be the news article template.
- Expand the template and select the **_Standard Values** item.
- In the right-hand pane, scroll down to the **Appearance** section.



- Select the **Lock Child Relationship** check box.

If you create a content item that is a child of a content item based on this template, it is not automatically structured in the item bucket. Instead it will retain its relationship with the parent item. For example, comments will always be children of the news article that they refer to.

2.4.6 Managing Item Buckets from the Control Panel

When you install the Item Buckets module, a new Item Buckets item appears in the Control Panel that allows you to manage your item buckets.



You can use the control panel to:

- Rebuild the item bucket indexes.
- Rebuild the indexes on a remote machine.
- See how many item buckets there are and how many content items are stored in each item bucket.

2.4.7 Item Bucket Search Settings

When you install the Items Buckets module, a set of settings items are added in the content tree at `/sitecore/system/Modules/Item Buckets/Item Buckets Settings`. You use these settings to configure how search works Item Buckets module.

You can:

- Change the keyboard shortcut for searching.
- Specify which field in your content items is the tag field.
- Specify the way that items open when you click an item in the search results.
- Specify the number of items displayed on a search results page.

You can also specify which field is used when you perform a tag search. To specify which field to use for tag searches, in the *Item Buckets Settings* item, you must ensure that the **Tag Parent** field points to the item in the content tree that contains all your tags.

You can then create a field called *Tags* in any template and set the type to either multi-list or bucket-list. Before you decide which kind of field you would like to use for tags, you should think about how many tags you are going to need.

Multi-list Field

A multi-list field can scale to hundreds. However, performance starts to degrade when you have more than a hundred items in the field.

Bucket-list Field

A bucket list has no limitation and can scale to thousands

We recommend that you use a bucket list.

2.4.8 Link Database

The Items Buckets module overrides the Link Database in your Sitecore installation. It does this so that it can scale to large amounts of items.

If you have code that uses the Link Database, there is a new overload for the `GetRefferes` and `GetReferences` methods for passing the page number. This is useful when an item has many connections, for example, an article template can have many connections to all the items that are based on it. The new overload ensures that when you look up the links between items, Sitecore will only load 20 items into memory at a time, rather than every link.

Chapter 3

Searching

This chapter describes how to search in item buckets.

This chapter contains the following sections:

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

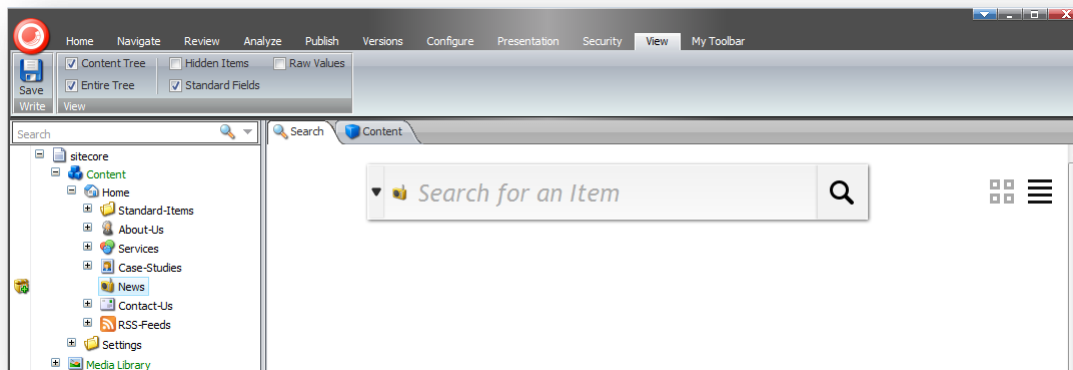
3.1 Searching and Item Buckets

Once you install the Item Buckets module and start to use, you will soon have buckets that contain hundreds if not thousands of content items. This underlines the need for search functionality that can help content authors find the individual content items that they need to edit and update.

To this end, Sitecore has implemented a new search interface as part of the Item Buckets module.

3.1.1 Viewing the Search Results

When you convert a content item into an item bucket, it automatically inherits the new search interface.



To search for an item, enter a search term that you think the item contains, press ENTER, and a list of search results is displayed.

You can use the buttons on the right to specify whether you want the results displayed in a list or in a grid.

Enabling Other Views

The *Item Buckets* module comes with other views that you can use to display search results:

- Gallery
- Infinite Scroll
- Video

These are not enabled by default.

To enable another view:

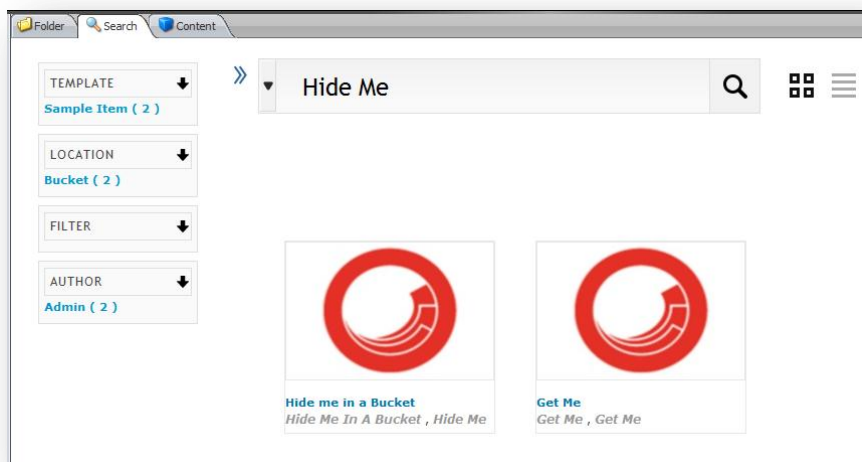
1. In the **Content Editor**, navigate to `/sitecore/system/Modules/Item Buckets/Views` and select the view that you want to enable.
2. In the **View Details** section, select the **Enabled** check box.

3.1.2 Using Facets to Refine your Search

The *Item Buckets* module supports searching for items by their facets.

When you run a search query, it also runs facet searches as well. These facets are listed on the left of the search pane. To see the facets that are available, click the arrow on the left of the **Search** tab.

You can use these facets to filter the results after you have searched in a bucket.



To filter your query, click on one of the blue facet links on the left. For example, in the previous image, the facets show that of the results returned for this search, two are based on the *Sample Item* template, two are located under the *Bucket* item, and two were written by the *Admin* user.

The *Item Buckets* module includes the following facets:

Author

Groups the results according to the authors that created the item.

Date Range

Collects the results into three groups — created within a day, a week, or a month.

Field

Groups the results according to the result of a text search of the tracked fields.

To mark a field as a tracked field, open the template and mark the fields as *Is Facet*. If you have templates that have duplicate field names, this will skew the results because the Lucene index concatenates these results as if they were the same field.

We therefore recommend that you do not use fields with the same name.

File Size

Groups the results according to the size of the field.

File Type

Groups the results according to file type.

Image Dimensions

Groups the results according to the dimensions of the images they contain.

Language

Groups the results according to language.

Location

Searches all the bucket locations to see which buckets the results are stored in.

If one of the results is not in a bucket, it will not appear in the location filter.

Template

Groups the results according to bucketable templates.

Troubleshooting

If you can see that the results that are shown in the facets differ from the results that are shown in the search results when you click the facet, this could be because:

- Lucene has retrieved an item that you do not have read access to and is therefore not shown in the results.

If your search results returns, for example, 12 items and the *Template* facet shows 11 results, this is because

- There are other types of items in the results that you have not marked as bucketable.

To improve performance, the Item Buckets module only applies facets to templates that are marked as bucketable.

For more information about creating facets, see the section *Specifying which Fields are Displayed in the Search Results*.

3.1.3 Language Search

The *Item Buckets* module support many different languages including Chinese, Arabic, and non-UTF based characters.



3.1.4 Complex Searches

The *Item Bucket* module supports 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, enter the text in the search box.

Range Searches

The *Item Bucket* module also supports range searches.

Examples that are supported:

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

Note

For numeric searches, you must pad the values to 8 places, for example, 400 should become 00000400

To run one of these searches, enter the text in the search box.

Combining, AND, OR, and NOT

The *Item Bucket* module also supports complex text searches.

Examples that are supported:

- +Tim +Chao — the results **MUST** contain Tim **AND** Chao.
- +Tim Chao — the results **MUST** contain Tim **AND** can contain Chao.
- Tim Chao — the results **CAN** contain Tim **OR** Chao.
- -Tim Chao — the results **CAN'T** contain Tim and can contain Chao.
- +(Tim Chao) — the results **MUST** contain Tim **OR** Chao.
- +(Tim Chao) Shanee — the results **MUST** contain Tim **OR** Chao and can contain Shanee.
- "Tim Chao" ~ 10 — the results must contain the words Tim and Chao within 10 words of each other.

When you perform a search like this the number you enter must be larger than 1 and there must have quotes around the text.

To run one of these searches, enter the text in the search box.

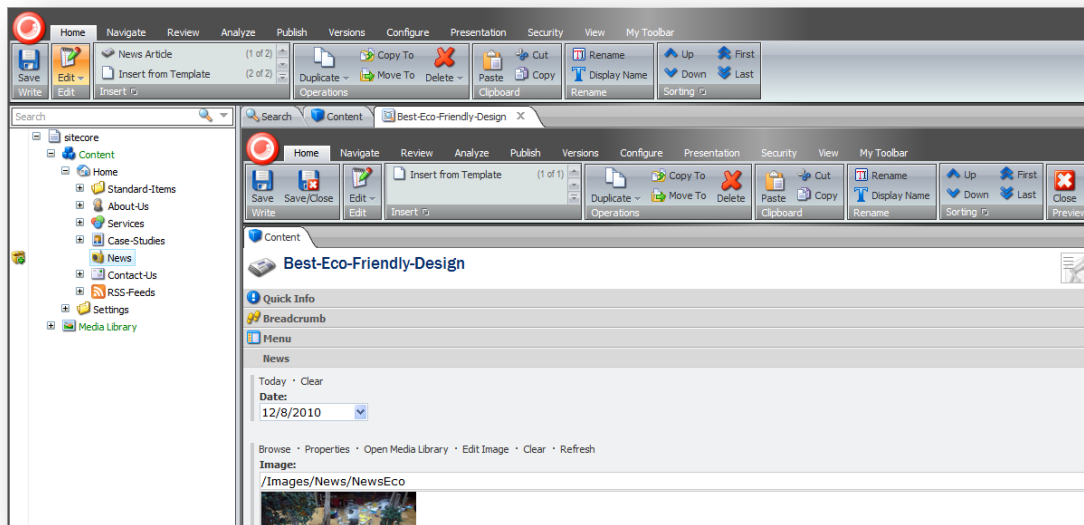
Search Tips

Whenever you run a new search, Sitecore displays a tip that explains how to use some aspect of the advanced search options.

Sitecore randomly selects the tip from a list of predefined tips. These tips are defined in `/sitecore/system/Modules/Item Buckets/Tips`. You can create new tips if you need to.

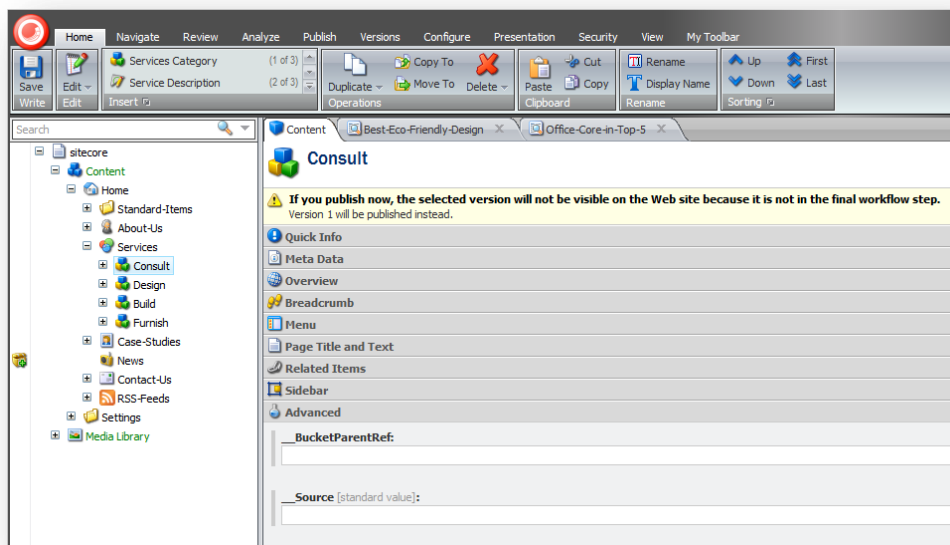
3.1.5 Opening Items in the Search Results

To open an item in the search results, click on the image in the result, the title, or any of the other blue links and the item opens in a new tab in the Content Editor.



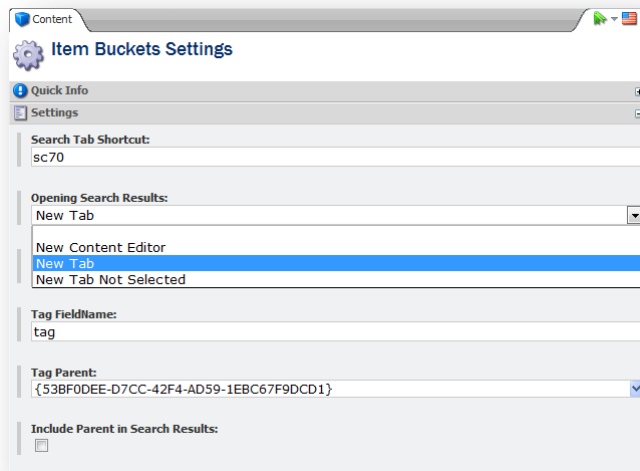
You can open as many of these extra tabs as you need.

This is an extremely handy tool that allows you to open multiple content items at the same time. These content items can be from multiple searches. If I open another content item, these tabs persist.



By default, when you click an item in the search results, the item opens in a new tab in the Content Editor. You can configure this behavior in the *Item Bucket Settings* item —

/sitecore/system/Modules/Item Buckets/Item Buckets Settings



The screenshot shows the 'Item Buckets Settings' dialog box in Sitecore. The dialog has a title bar with 'Content' and a gear icon. Below the title bar is a 'Quick Info' section and a 'Settings' section. The 'Settings' section contains several fields: 'Search Tab Shortcut' with the value 'sc70', 'Opening Search Results' with a dropdown menu showing 'New Tab' selected, 'New Content Editor' with a dropdown menu showing 'New Tab' selected, 'Tag FieldName' with the value 'tag', 'Tag Parent' with a dropdown menu showing '{53BF0DEE-D7CC-42F4-AD59-1EBC67F9DCD1}', and 'Include Parent in Search Results' with an unchecked checkbox.

Content

Item Buckets Settings

Quick Info

Settings

Search Tab Shortcut:
sc70

Opening Search Results:
New Tab

New Content Editor
New Tab

New Tab Not Selected

Tag FieldName:
tag

Tag Parent:
{53BF0DEE-D7CC-42F4-AD59-1EBC67F9DCD1}

Include Parent in Search Results:
☐

3.2 Using Search Filters


The *Item Buckets* module allows you to insert filters into a search string and narrow down the results.

To use a search filter, enter the reserved filter keyword and the *Item Buckets* module either auto-suggests a filter or prompts you to enter a date or a text.


The module contains the following filters:

- Template
- Version
- Language
- Creation Start and End Date
- File Type
- Author
- Tag
- Site
- Advanced Text
- ID
- Custom
- Links
- Sort
- Location



Template Filter


template:Sample Item


Version Filter


Language Filter

language:en


Start and End Date Filter

December 2011

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Clone:false Version:1 Created: 9/12/2011 11:08:05 AM by



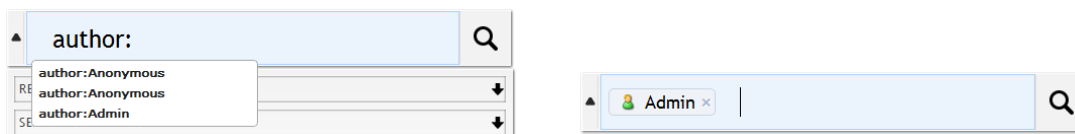
When you are working with the calendar, you can use the following keyboard shortcuts:

- 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
- ESC — close the date picker without making a selection

File Type Filter



Author Filter

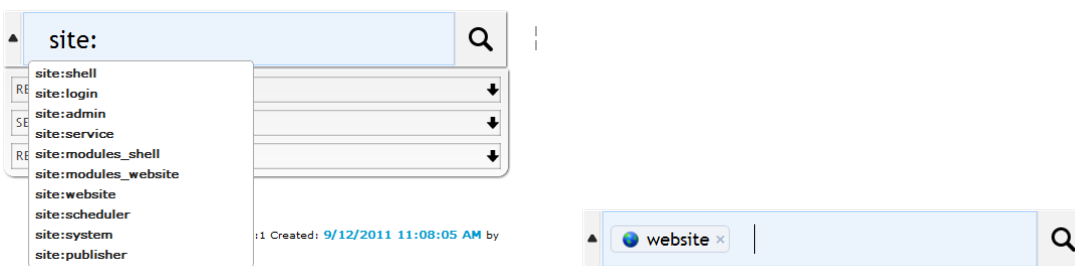


You must enter at least 2 characters before the system makes any suggestions.

Tag Filter

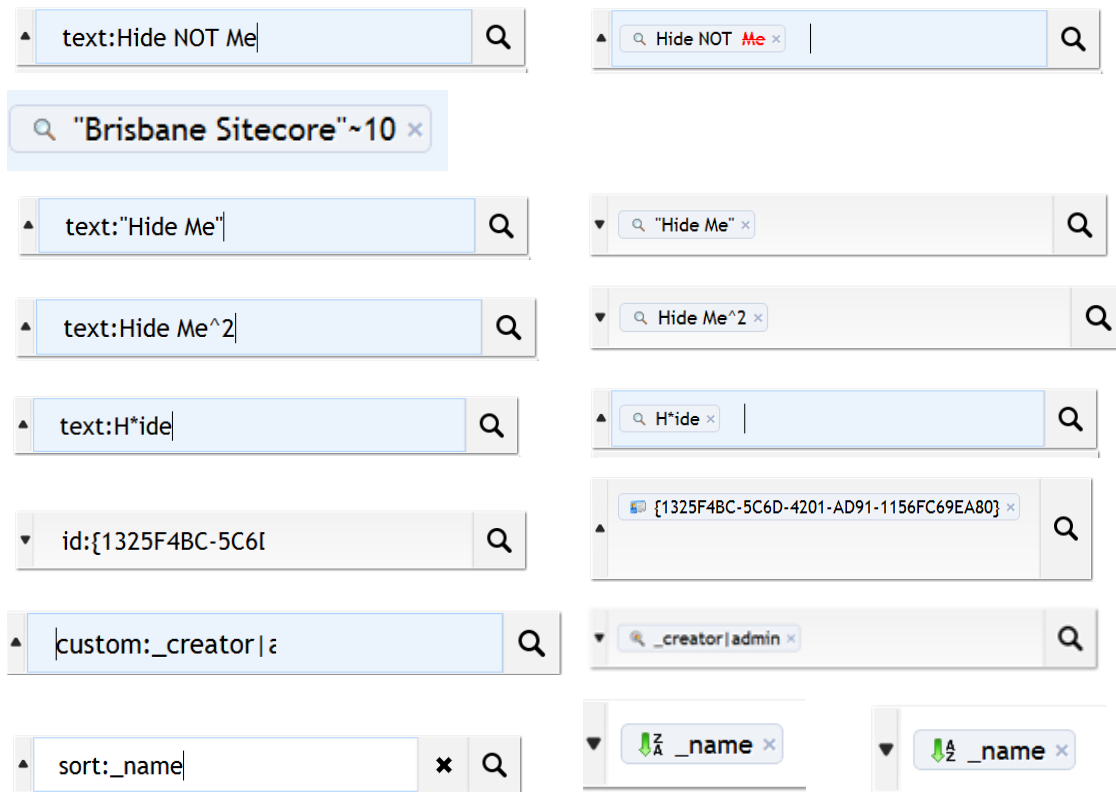


Site Filter



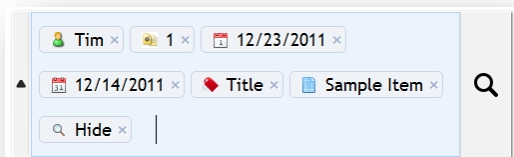
Advanced Text





Combining Filters

You can also combine various different filters.

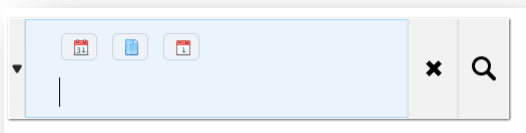


Note

All the filters are currently lowercase sensitive.

3.2.1 Auto-Organising

The Item Buckets modules autoshrinks the searches if you enter too many terms in the drop-down box.



3.2.2 Paging Results

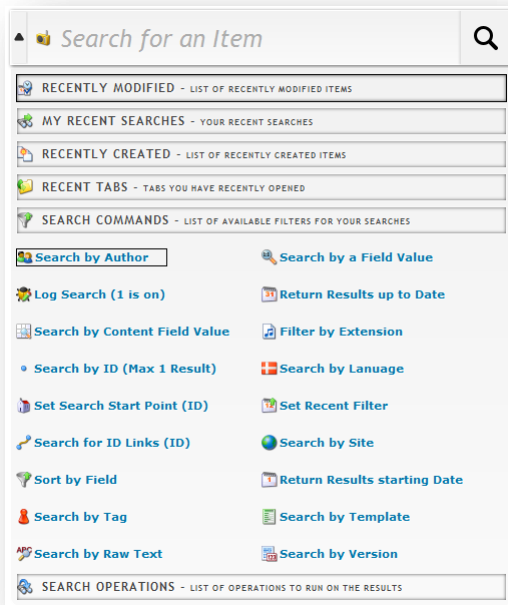
The results appear in lists of 20 items per page.

If more than 10 pages are shown, you can move on to the next 10.



3.2.3 Search Helpers

There is a drop-down list beside the search field. When you click the drop-down arrow, a list of categories appears. You can expand each category to see a more detailed list of search options.



You can add more search helpers to this list. The search helpers are stored at:

`/sitecore/system/Modules/Item Buckets/Settings/Search Box Dropdown.`

The search helper categories are:

Category	Description
Recently Modified	A list of the items that have been modified recently.
My Recent Searches	A list of your recent searches.
Recently Created	A list of the items that have been created recently.
Recent Tabs	A list of the tabs you have recently opened.
Search Commands	A list of filters that you can use in your searches. These filters are stored in — <code>/sitecore/system/Modules/Item Buckets/Search Types</code>

Category	Description
Search Operations	<p>A list of operations that you can perform on the search results.</p> <p>This is a powerful feature that lets you run any operation on the search results.</p> <p>You can add more operations to this list.</p>

3.2.4 Security and Item Buckets

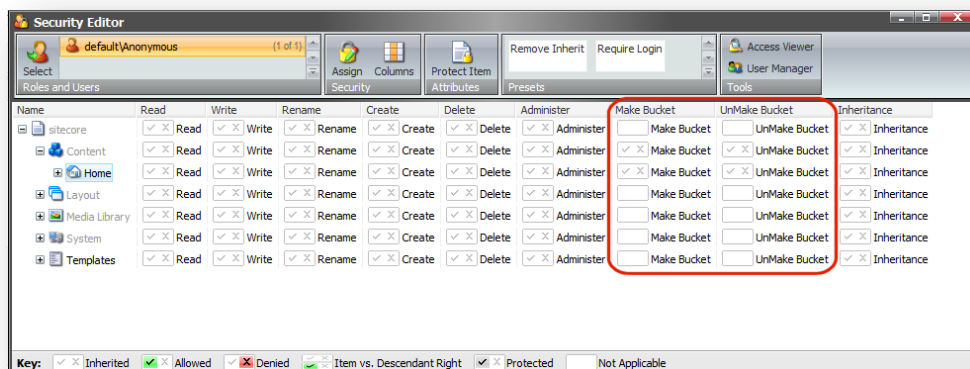
When you install the Item Buckets module, some new security options are added to the Security Editor.

If you need to restrict the ability of a user or role to convert a content item into an item bucket or to convert an item bucket back into a normal item, you can use the Security Editor to change their access rights.

The Item Buckets module adds two extra access rights to the Sitecore security system:

- Make Bucket
- UnMake Bucket

If these security settings are not immediately available in the **Security Editor**, in the **Security** group, click **Columns** and then in the **Columns** dialog box, select *Make Bucket* and *UnMake Bucket*.



Tip

We recommend that you use the Sitecore security system to prevent particular users from performing certain operations. For example, you do not want someone to accidentally delete a lot of items.

Locking

To minimize the possibility of accidentally creating an item bucket or making a template bucketable, even an Administrator *must* place a lock on the item before they can place it in a bucket.

3.2.5 IA Modifications

The Item Buckets module drastically changes the way that identification and authentication — IA — is managed on a website. Items that are stored in a bucket no longer maintain a child to parent relationship and the item bucket is simply a pool of available items that are searched with a custom search index.

The normal IA operations are still supported with item buckets including:

- Copy To
- Copy From
- Move To
- Move From
- Clone To
- Clone From
- Delete

If you copy or move a content item that is based on a bucketable template into an item bucket, the item is placed in the bucket and automatically organized in the bucket structure.

If you copy or move a content item that is not based on a bucketable template into an item bucket, it is placed in the bucket and is treated like a normal content item.

If you delete an item bucket, a message appears informing you that by deleting the bucket you could potentially be deleting the content items that are stored in it. You can restore these content items from the recycle bin.

You can also

- Drag a copy into an item bucket.
- Drag a copy out of an item bucket.
- Drag an item to move it into an item bucket.
- Drag an item to move it out of an item bucket.

Keyboard Shortcuts

Here is a list of the keyboard shortcuts that you can use with item buckets.

Shortcut	Description
CTRL + SHIFT + B	Converts a content item into an item bucket.
CTRL + SHIFT + D	Converts an item bucket into a content item.
CTRL + X	Clears the text box.
CTRL + SHIFT + S	Adds a search tab above any content item, but does not convert it into an item bucket or convert an item bucket into a content item.
CTRL + SHIFT + A	Uses a search interface to find the closest ancestor.
Space Bar	Scrolls the results when the text box is not given focus.
CTRL + Space Bar	Show the dropdown options when the text box is given focus.
ESC	This hides the drop-down list if it is shown.
CTRL + B	Shifts the focus to the text box if it is outside the textbox.
SHIFT + Number	Runs the search in a particular view. 1 will run the first, 2, the second, and so on.
1-9	The numeric characters move focus to the corresponding page of the search results.

3.2.6 Query the Index from any Inbuilt Field that Supports Lists

In an item bucket you can potentially store many thousands of items. This also means that you could have thousands of items showing up in a multilist field, a treelist, and so on.

To query these fields quickly, you can use the new *Lucene* syntax of setting a datasource.

To set a datasource, start your query with the word *lucene* followed by the name of a field and the value you would like to search for, for example:

```
lucene:_name:Sitecore;location:<item guid>;
```

Using a Custom Class to Query the Index

To query a field quickly, you can use a compiled class that returns an `Item[]` as the source of your fields. Start your query with the word *code* and then enter the `.class`, `assembly namespace` as the source field.

You can do this by implementing the `IDataSource` interface.

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

3.2.7 Using the New Field Types

There are some 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 degrading performance.

Bucket List Field

Allows you to attach a search query to a multilist field and display the search results as selectable items. For example, if you want a multilist of all the product items, you can set the source field of the field to `TemplateFilter="Product ID"` and it will return the items in the list.

Here is a list of the available filters. The filters are all optional:

- `IndexName=bucket`
- `Language=en`
- `TemplateFilter=<Template ID>`
or
- `query://` (replace "=" in your query with "->")
 - For example: `query://*[@@templatename->Product]`
- `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 will not return any results.

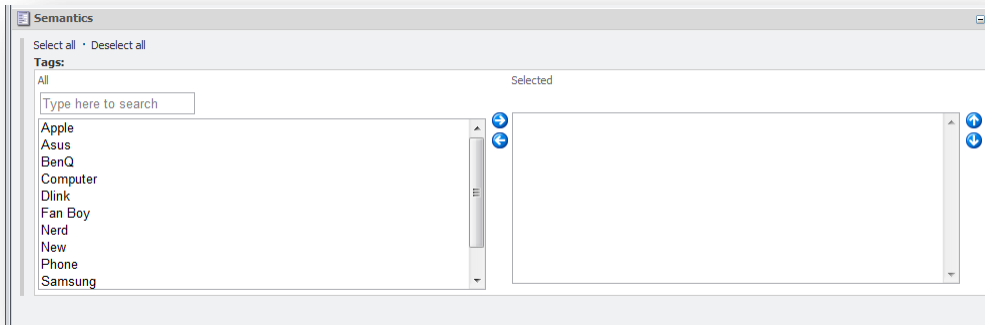
Example 1

```
TemplateFilter=<Product Template ID>|<Another Template ID>
```

Example 2

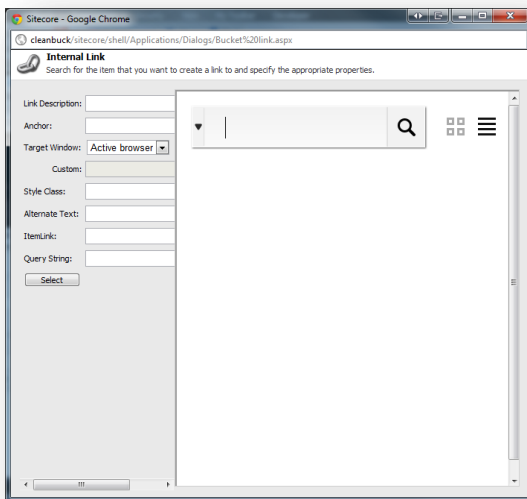
```
TemplateFilter=<Product Template ID>&FullTextQuery=Nicam
```

These queries populate the list that you can then select the items from. You can also use the search filter to filter the list even more.



Bucket Link Field

This allows you to link to items that are contained in a bucket.



Bucket IFrameField

If you would like to search for items as well as add and remove them from a list, you can setup a field.

```
Name = <anything>
FieldType = IFrame
Style = height:800px;
Field Source =
http://<sitename>/sitecore%20modules/Shell/Sitecore/ItemBuckets/IFrameField.aspx
```



This is no new control; instead it utilizes an existing field type within Sitecore to show a search interface.

This gives the following control in your content item:



To remove items from the field, in the **Selected Item List**, click the X next to the items. To add items to the list, click the search result links.

3.2.8 Using Item Buckets with the Data Source of a Control

Because everything contained within a bucket is hidden and cannot be selected by users, a user can now choose the data source for a control by specifying a search query. This section describes the syntax for setting the data source to run a Bucket Query.

Types of queries:

- Template
- Version
- Language
- Creation Start and End Date
- File Type
- Author
- Tag (Facet)
- Site
- Advanced Text
- ID
- Custom

You must list the queries with a semicolon separated list. For example, to search for all the Nicam products, you could specify:

```
text:Nicam;template:<Product Template ID>;
```

This is passed to your control as a string and you can then use the `BucketManager` to resolve this to a list of items:

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

Tips

Sitecore runs the query in the context of its location, for example, if you run the query on a control, it will start the query from the context item and work down through all its descendants. If you need to perform a global search or search in another part of the content tree, you can pass the location parameter to the query for the data source. For example, the following query looks for all the items that are tagged *Nicam* and that exist under the `/sitecore/content/home` node — the ID of the home item is shown:

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

If you do not want your users to enter these queries manually, you have many choices.

Change the inbuilt type of the **Type** field of the `/sitecore/templates/System/Layout/Rendering Parameters/Standard Rendering Parameters/General/Data Source` item.

You must change it from the **Internal Link** type to the **Bucket Datasource** type.

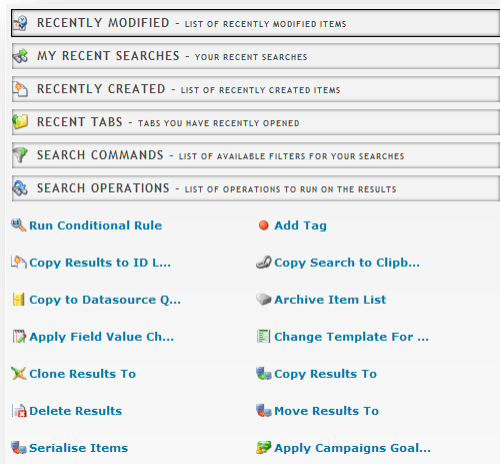
This allows users to use a search interface to build their queries. Click **Build Query** and a search interface opens that you can use to build up your query. You can either build up a search or click on one of the results — in this case; it will just return that one item as the result.

Tip

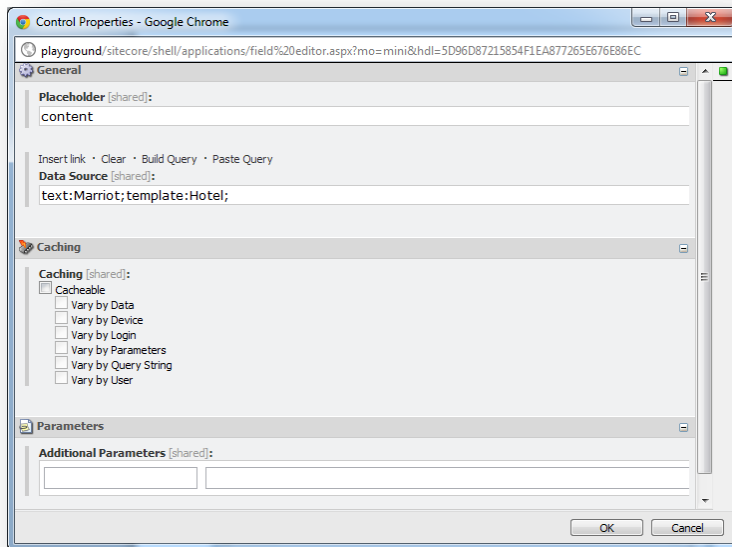
If you want to lock the results so that new items that are created after you set the data source are not shown, put a start date and end date filter in the search. You must set both the start date and the end date for this to work.

It also allows you to copy and paste queries from a search that you have performed on an item.

To paste queries, search for content in the normal way. Once you have a filter, click the drop down menu and under **Search Operations**, click **Copy to Datasource Query**.



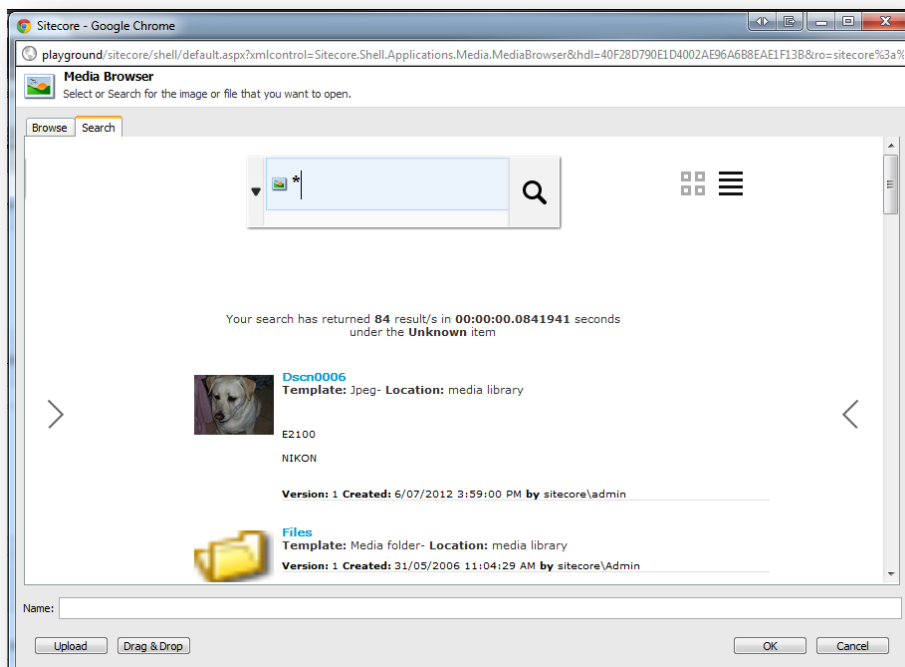
When you configure your presentation data source, you can paste this query into the **Data Source** field.



Searching for Images

The Item Buckets module replaces the inbuilt image field type 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.



3.3 Page Editor and DMS Support

This section describes how the Page Editor supports item buckets.

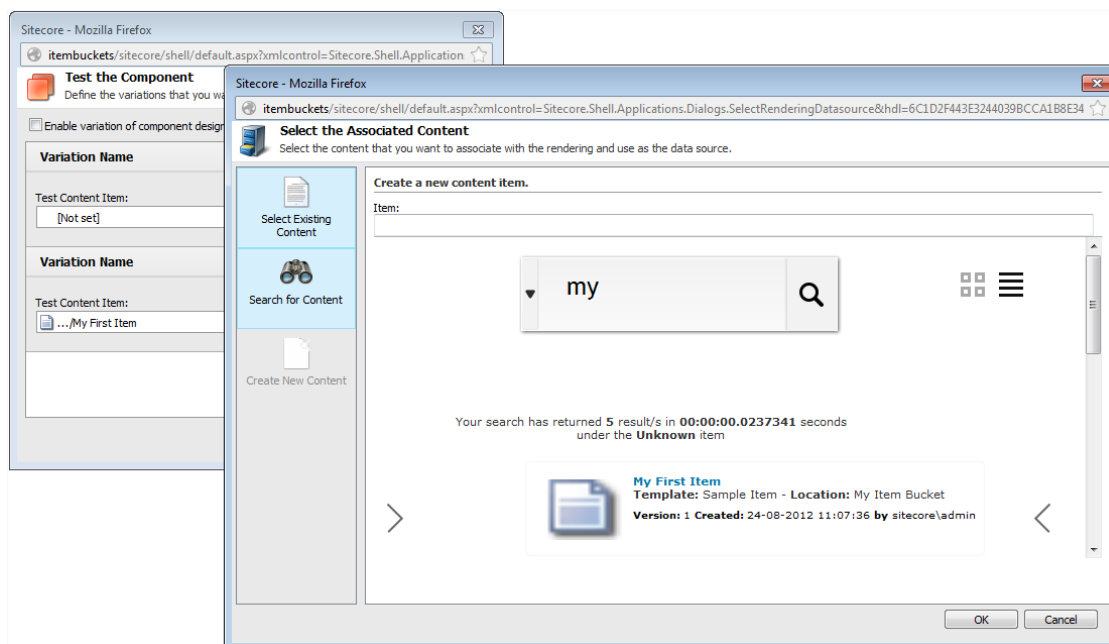
When you work in Sitecore, you may need to link to an item that is stored in an item bucket. For example, you may need to insert a link to a bucketable item or use a bucketable item as a variation when you are creating an MV test. To help you locate items in these situations, the Item Buckets module provides search functionality in the appropriate dialog boxes in the Page Editor and in the Content Editor.

3.3.1 Personalization and MV Tests

When you set up a personalization rule or create an MV test, you must specify the data source for each variant in the test. To facilitate this process, the Item Buckets module provides a search tab in the appropriate dialog boxes so that you can use to search for the content item that you want to use.

For example, when you create an MV test, you must select the content items that should be used as variants in the test. Selecting a normal content item is pretty straight forward — you browse the content tree and select the item in question.

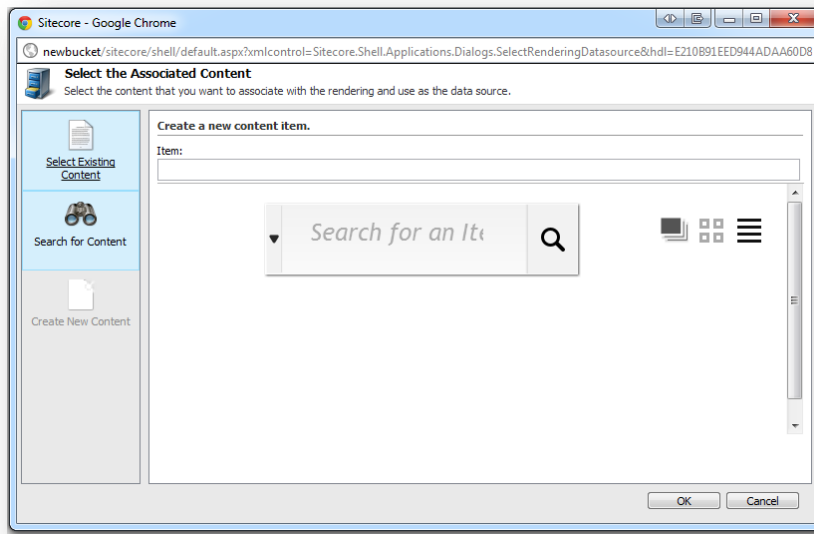
However, this is not so simple if you want to use a content item that is stored and hidden in an item bucket. The Item Buckets module adds a search tab in the Select the Associated Content dialog box.



For more information about searching, see the section *Searching and Item Buckets*.

3.3.2 Setting the Data Source

When you specify the datasource item for a control, you can also use the new search functionality to either set an individual item or a list of items as the data source.



3.4 Inserting and Managing Links

This section describes how to manage links to items that are stored in item buckets.

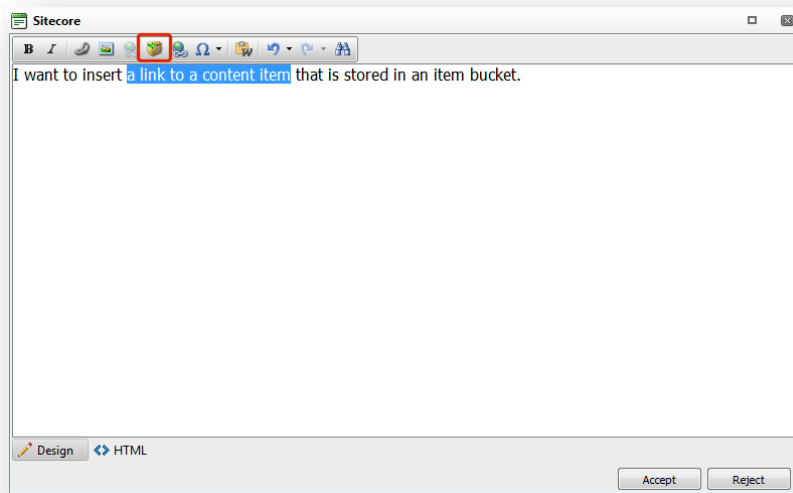
3.4.1 Inserting a Link

To link to an item that is stored in an item bucket,

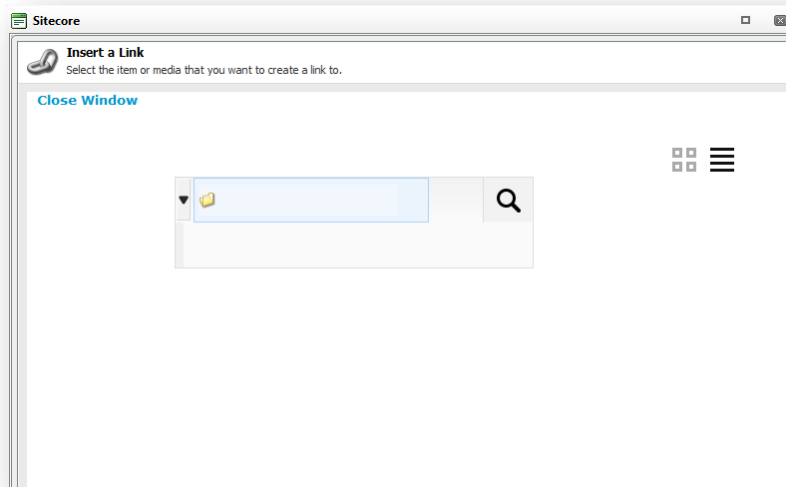
Inserting a Link in the Rich Text Editor

To insert a link with a Rich Text Editor field:

1. In the **Content Editor**, open the content item that you want to edit.
2. Scroll down to the rich text field that you want to edit and click **Show Editor**.
3. In the **Rich Text Editor**, select the text that you want to use as a link.

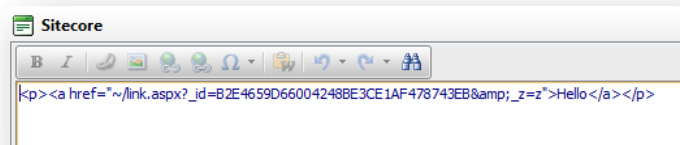


4. Click the **Insert Item Bucket Link** button and an **Insert a Link** dialog box appears that contains the item buckets search functionality.



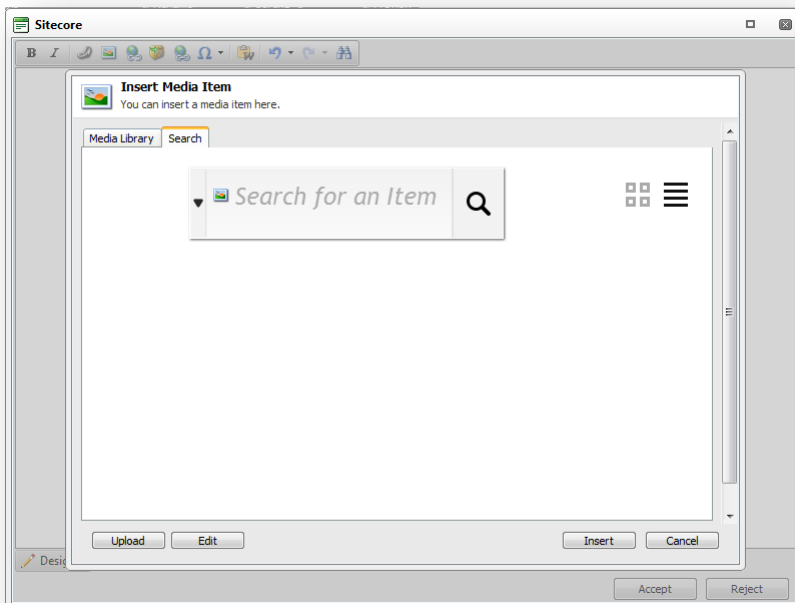
5. In the **Insert a Link** dialog box, in the search field enter the search words.
6. In the search results, click the item that you want to link to.

The link to this item is inserted into the text.



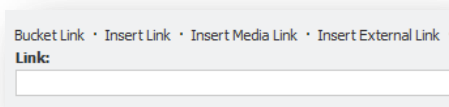
If you want to insert an image that is stored in an item bucket into a rich text field, you can use the same search functionality to locate the image.

The **Insert Media Item** dialog box contains a search tab.

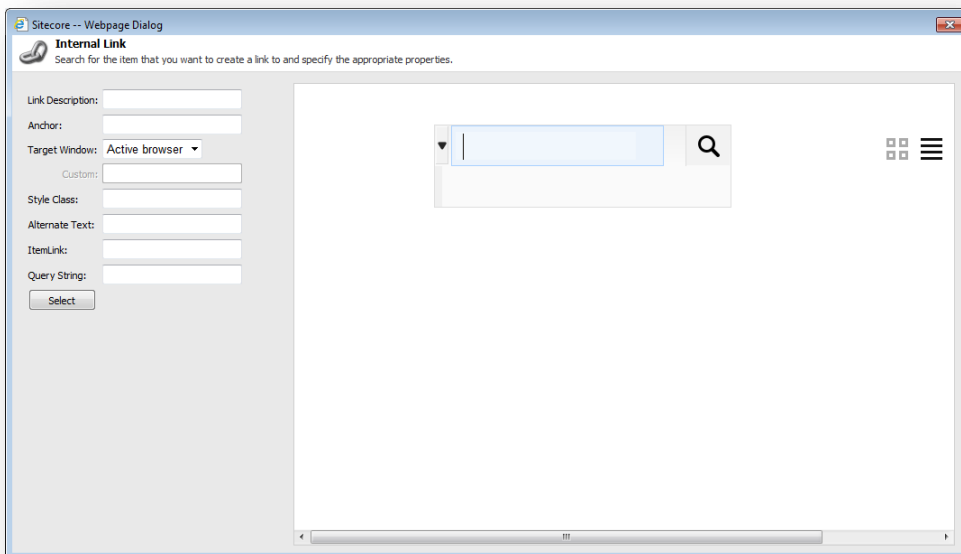


Inserting a Bucket Link

The Item Buckets module also contains some new field types. One of these is the Bucket Link field.



To insert a link into a *Bucket Link* field, click **Bucket Link** and the **Insert a Link** dialog box opens.



For more information about the bucket link field types, see the section *Developing with Item Buckets*.

3.4.2 Link Providers

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

The built in Link Provider converts a bucketed item. For example, it converts:

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

into:

<http://site/products/computers/apple/macbook%20pro-201112221021>

The logic formats the link so that it is SEO Friendly and embeds any tags that are associated with the content item into the URL.

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

```
<!-- 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>-->
```

3.4.3 Item Resolvers

The Item Bucket module ships with an Item Resolver that manages requests to items that are stored in a bucket folder. You must extend this if you include your own Link Provider to override items contained within Buckets.

<http://site/products/computers/apple/macbook%20pro-201112221021>

into:

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

3.4.4 Tagging Associations Across Many Items

Because the parent to child relationship is removed, content items that are stored in an item bucket need a way to connect to other content items. Item buckets supports a semantic tagging system which allows you to tag associations on all items. To support the tagging of items, add a Bucket List field to the template and make sure to call it *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

You must set the *IndexName* because the context switch that Item Buckets performs depends upon which part of the content tree that you are querying from.

This is where semantics comes into play. You can add any tag to any item. For example, if you tag items as *Work in Progress*, you can later search for all items that are marked *Work in Progress*.

Chapter 4

Developing with Item Buckets

This chapter describes how to use the Sitecore API to work with item buckets.

This chapter contains the following sections:

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

4.1 Buckets API

The Item Buckets module is designed to address the challenges that come with managing millions of content items in the content tree. This involves taking both performance and management into consideration.

When developers write code, they must use the API to reference items in a new way.

The following standard Sitecore API method and property do not work with items that are stored in an item bucket:

- `GetChildren()` and `Children`

The following methods work but we do not recommend them because of performance considerations:

- Axes Methods
- Properties

Instead, developers can use a new API to run queries on items and talk directly to the Bucket Index.

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

You can use this table to migrate your own code so that it works with the new item bucket system. As you can see, the performance gains are considerable. These measurements were made with only 1000 items in the content section of the content tree.

You can see the performance gains for your presentation components on the `stats.aspx` page

Sublayout: /layouts/Site/Sublayouts/Components/CallOut.ascx	website	4	0	0.0032	0	0.0038	0	00:00:00.0000131	0
Sublayout: /layouts/Site/Sublayouts/Components/DateList.ascx	website	4	0	0.0043	0	0.006	0	00:00:00.0000175	0
Sublayout: /layouts/Site/Sublayouts/Components/LinkList.ascx	website	4	0	0.0046	0	0.0054	0	00:00:00.0000186	0
Sublayout: /layouts/Site/Sublayouts/Components/MainArticle.ascx	website	4	0	0.4726	0	1.7431	0	00:00:00.0018905	0

Note

The average loading time is extremely low because the presentation components go straight to the index and then fill in the objects in the Bucket data layer.

4.1.1 Query Examples

Note

For simplicity, the following methods have excluded the `hitCount` attribute that returns through an out parameter and allows you to page results.

Using the Bucket Manager

```
//Test 1 - Get All Items Under the Home Item based on the 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 where the Title Field starts 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 on the template with ID
//"{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);
```

4.2 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 the statements in your CS file you get extension method replacements for the <code>GetChildren()</code> , <code>Children</code> and <code>Axes</code> methods.
Sitecore.ItemBucket.Kernel.ItemExtensions.Axes	ItemExtensions	This allows you to have new extension methods and properties on an item object and to run queries on an item like <code>item.Search("")</code> or <code>item.FullSearch("")</code>
Sitecore.ItemBucket.Kernel.Managers	BucketManager	This is your main entry point for working with bucket containers. It contains methods such as <code>IsBucket()</code> , <code>IsBucketItem()</code> , <code>GetParentBucket()</code> and so on. This also allows you to run searches.
Sitecore.ItemBuckets.Kernel.Search.Query	BucketQuery	The entry point for running Lambda or Linq expressions that are converted into Lucene queries.
Sitecore.ItemBucket.Kernel.Util	SitecoreItem	To keep memory usage to a minimum, all searches return a <code>SitecoreItem</code> which is a stripped-down representation of the <code>Item</code> class.
Sitecore.ItemBucket.Kernel.Kernel.Hooks	QueryWarmUp	An abstract class that allows you to specify warm-up queries that run when Sitecore is initializing. It is useful to run common queries so that they are cached and then returned faster when you request them later. This sacrifices startup time for operational performance.
Sitecore.ItemBucket.Kernel.Presentation	BucketPresentationExtensions	A helper class for converting a string datasource into a bucket query.

4.3 Working with Extension Points

If the functionality that is available in the Item Buckets module is not all that you need, there are many extension points that you can use to extend the features.

Interfaces

The following table lists the interfaces that you can extend if necessary.

Namespace	Interface	Description
Sitecore.ItemBucket.Kernel.FieldTypes	IDataSource	If you want the list of field types within your Sitecore template to take advantage of populating itself from a Lucene query, implement this interface.
Example of IDataSource Implementation	BucketListQuery	<pre>public class BucketListQuery : IDataSource { public Item[] ListQuery(Item itm) { return itm.Children.ToArray(); }</pre>
Sitecore.ItemBucket.Kernel.Interfaces	IBucketController	If you want to build your own UI layer that can send the Bucket Handler a request and receive a list of items, implement this interface.
Sitecore.ItemBucket.Kernel.Interfaces	IBucketSearchQuery	There are many filters that come with Item Buckets, for example, Author, Start Date, Text, Tags, and so on. If you want to implement a new filter, implement this interface.
Sitecore.ItemBucket.Kernel.Interfaces	ITag	A tag repository works with the ITag Interface. If you have tags that are pulled from external systems, you must implement this interface so that you can apply these tags to items and then use them to search with.
Sitecore.ItemBucket.Kernel.Interfaces	ITagRepository	The Item Buckets module comes with a Tag Repository that uses items in the content tree. If you have a tag repository and would like to use this to search for tagged content within Sitecore, you must implement the <i>ITagRepository</i> interface.

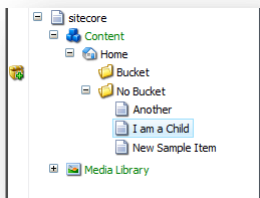
Namespace	Interface	Description
Sitecore.ItemBuckets.Kernel.Kernel.Search.SearchOperations	ISearchOperation	If you want to introduce new actions on a list of search results, you must implement this interface. For example, if you want to search for all the items in the content tree that have <i>\$name</i> in any of the fields and replace them, you can implement a new <i>ISearchOperation</i> so that content authors can do this.
Sitecore.ItemBuckets.Kernel.Search	IFacet	Item Buckets comes with 5 different types of faceting: <ul style="list-style-type: none"> • Templates • Fields • Dates • Locations • Authors <p>If you want to introduce your own faceting categories, you must implement the <i>IFacet</i> interface.</p>
Sitecore.ItemBuckets.Kernel.Search	ISearchDropDown	When you run a search, you see a dropdown from the textbox where you enter your text. If you want to add your own, you must implement this interface. Then, you must add an item to the content tree /sitecore/system/Modules/Item Buckets/Settings/Search Box Dropdown — to register this class and ensure that it appears in the drop-down menu.

4.3.1 Working with Existing Code

If you install the Item Buckets module in an existing solution, you must be extra careful and we recommend that you test this first. We recommend that you never install this module directly onto an authoring, production, or any other server. You should test the functionality to ensure that it works with your existing code.

Example 1

If you have a control 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 shows the name of the item in the data source. */
}
```

This renders the following subitems:

- Another
- I am a Child
- New Sample Item

If you turn the *No Bucket* folder into a bucket, and use the same code, it renders:

- 2011

You should change your code to use the Bucket API:

```
protected void Page_Load(object sender, EventArgs e)
{
    repeater.DataSource = Sitecore.Context.Item.Search(text: "");
    repeater.DataBind();
    /* We can assume that the markup shows the name of the item in the data source. */
}
```

or

```
protected void Page_Load(object sender, EventArgs e)
{
    repeater.DataSource = new BucketQuery().WhereContentContains("").Run();
    repeater.DataBind();
    /* We can assume that the markup shows the name of the item in the data source. */
}
```

or

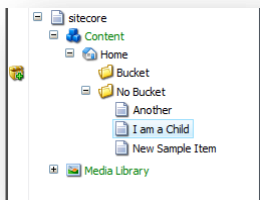
```
protected void Page_Load(object sender, EventArgs e)
{
    repeater.DataSource = BucketManager.Search(text: "")
    repeater.DataBind();
    /* We can assume that the markup shows the name of the item in the data source. */
}
```

This renders:

- Another
- I am a Child
- New Sample Item

Example 2

If you have a control 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 renders:

- Another
- I am a Child
- New Sample Item

If you turn the *No Bucket* folder into a bucket, and use the same code, it renders:

- 2011
- 12
- 9
- 15
- 1
- Another
- I am a Child
- New Sample Item

You should change your code to use the Bucket API:

```
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 renders:

- Another
- I am a Child
- New Sample Item

4.3.2 Common Requirements

A lot of websites are very similar and have similar requirements.

Here is a list of example queries that retrieve items for common web controls.

Example

Side Menu — get all descendants based on the *Site Section* template.

```
//This uses 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 uses the string that is specified in the method to query the buckets for items.
var items = BucketManager.ParseDataSourceQueryForItems(("<Insert Query Here>",
Sitecore.Context.Item, 0, 20);
```

4.3.3 Creating a Tag Repository

To create a new tag repository and to enable the tag filter in searches, you must create a new class in your Visual Studio solution and implement the `ITagRepository` interface.

You must implement methods that return *All Tags*, *Single Tags*, *First Tag*, *GetTags by Name*, *Get Tags by Value*.

The following code pulls 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 also come from different sources. Once you have created your code, you must create a Tag Repository item in the content tree, under /sitecore/system/Modules/Item Buckets/Tag Repositories. You must point to the class you just compiled with the namespace.class assembly syntax.

The Item Buckets module comes with a built-in tag repository.

To enable the built-in tag repository on your website:

1. Create a field called **Tags** in any template. This must be a multi-list or bucket-list field.
2. Navigate to the *Sitecore Item Buckets Settings Item* —
/sitecore/system/Modules/Item Buckets/Item Buckets Settings.
3. In the **Tag Parent** field, point to the parent item that stores all the tags.

The parent item that stores all the tags can also be a bucket if need be. The parent item that stores all the tags is a suitable candidate for a bucket.

4.3.4 Creating a Facet

If you would like your content authors to be able to filter their searches on different facets, you must implement some code. You must implement the `IFacet` interface.

Here is an example that displays all the item buckets in the content tree as facets on the item bucket search page:

```

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.ConstructQuery(query), false, true, 0,
0, "_path", Filters);
            return results;
        }
    }
}

```

You must then create a *Facet* item in the `/sitecore/system/Modules/Item Buckets/Facets` folder and fill in the `namespace.class` assembly.

4.3.5 Configuration Files

The Item Buckets module installs a `Sitecore.ItemBuckets.config` file which contains some configuration settings. This section describes this configuration 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>
```

This configuration declares a new index called *buckets*. It then uses a custom crawler to tokenize and list the field types. This enables you 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 that contains 100,000 items or more. You may need to tweak these numbers depending upon the number of items in your content tree.

Custom Settings

The `Sitecore.ItemBuckets.config` file contains some custom settings, including:

```
<settings>
  <setting name="BucketTriggerCount" value="100"/>
  //When an item has 100 Siblings, it asks you if you want to automatically convert the parent
  item into an item bucket.

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

  //If you want to change the template of the folder item that organizes all the hidden
  bucketable items, you must change this setting to point to the GUID oof the new folder item.
  For the first revision of this module, we recommend that you use the default value.

  <setting name="SecuredItems" value="blur"/>
  //This setting determines what happens to the results that are returned when the user does not
  have permission to access them. The options are "hide" and "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>
```

The `Sitecore web.config` file contains the following setting:

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

//This is the index update interval that is set when unstructured items are created, deleted,
modified etc. in the Web database. If you have item creation, deletion or modification on your
Web database, the items won't automatically be included in your index. This interval
determines how often the index is updated on the Web database. This is necessary once you
start working with over 100,000 unstructured items in your index.
```

4.4 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" />
          </fieldTypes>
        </ItemBucketSearch>
      </locations>
    </index>
```

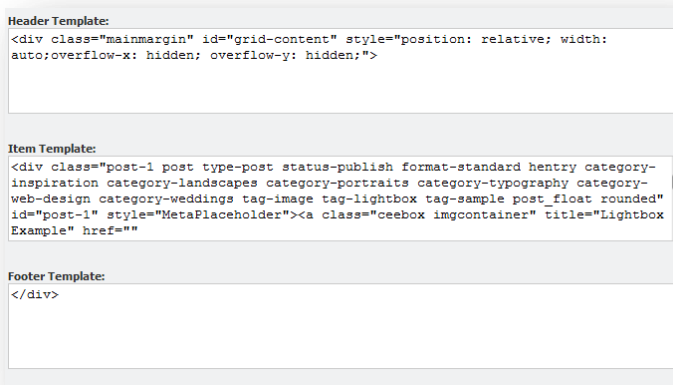
```
<fieldType name="tree list" storageType="NO" indexType="TOKENIZED"
vectorType="NO" boost="1f" />
</fieldTypes>
<include hint="list:ExcludeTemplate">
  <layout>{ADB6CA4F-03EF-4F47-B9AC-9CE2BA53FF97}</layout>
</include>
</ItemBucketSearch>
</locations>
</index>
```

4.4.1 Adding a New View

Developers can add new views to the search results to cater for different situations, for example, browsing an image gallery and seeing a small version of the image in the results. The default views are *Grid* and *List*.

To add a new view:

1. Navigate to the `/sitecore/system/Modules/Item Buckets/Views` item and create a new view item.
2. Set the **Header Template**, **Item Template**, and **Footer Template** fields to the HTML tags that you want to return in the search results.



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

Placeholder	Description
MetaPlaceholder	The CSS style that you want to use when the results are displayed.
LaunchTypePlaceholder	Whether it will launch the result in a new tab or in a new Content Editor window.
ItemIdPlaceholder	The item ID.
ImagePathPlaceholder	The path to the image of the item.
NamePlaceholder	The name of the item
TemplatePlaceholder	The name of the template that the item is based on.
BucketPlaceholder	The bucket that this result comes from.
ContentPlaceholder	The content of the result.
VersionPlaceholder	The version of the content item.
CreatedPlaceholder	The date that the content item was created.

Placeholder	Description
CreatedByPlaceholder	The person who created this item.

Chapter 5

Sitecore Items and 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, having remote querying, and using 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

5.1 The Configuration Files

This section describes how to use the `Sitecore.ItemBuckets.BigData` configuration files to scale the Item Buckets module.

5.1.1 In-Memory Indexes

The default Bucket Indexes are written to the file system and persisted across the application pool resets. If you have any read-only information within the content tree that will not change, you can have this stored within an in-memory index that is initialized when the application starts up. Depending on the size of this index, this slows down the application start time of Sitecore but gives you have a much faster index than the file system based index.

Advantages	Disadvantages
Speed — it is fast.	Results in a slower application startup time.
No storage needed on the hard disk.	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>
```

5.1.2 Remote Index

The default Bucket Indexes are written to the file system in the default data directory specified in the `web.config` file. Rebuilding indexes can be quite slow and intensive on the web server. It is therefore a good idea to rebuild your indexes on a remote machine.

Advantages	Disadvantages
Your remote machine can have an SSD drive on it which will make the indexing a lot faster.	While the index is copied back into the local data directory, there is a short time when the index is locked for reading and writing.
The hit on the server for re-indexing is done on the target machine, not the machine serving up the content.	

In the `RemoteIndexLocation` setting, enter a network path that has full read and write access.

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

You can then specify your index:

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



```

        <indexes>
<index id="itembuckets_buckets_remote"
type="Sitecore.ItemBuckets.BigData.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>

```

5.1.3 New Crawlers

For all the other indexes, you can use the new CustomCrawler. You can customize the CustomCrawler.

```

//ExcludeTemplate allows you to specify templates that are not indexed
    <include hint="list:ExcludeTemplate">
        <layout>{ADB6CA4F-03EF-4F47-B9AC-9CE2BA53FF97}</layout>
    </include>
//IncludeTemplate is only needed if you turn "IndexAllFields" off
    <include hint="list:IncludeTemplate">
        <layout>{ADB6CA4F-03EF-4F47-B9AC-9CE2BA53FF97}</layout>
    </include>
//IncludeField is only needed if you turn "IndexAllFields" off

<include hint="list:IncludeField">
    <fieldId>{8CDC337E-A112-42FB-BBB4-4143751E123F}</fieldId>
</include>
//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>
//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>

```

5.1.4 Query Server

You can specify a dedicated query server that runs all the bucket queries through the UI. This may be useful if you have many content authors and you don't want the overhead from their queries to affect the performance of the rest of the system.

Use the *QueryServer* setting, to route all the 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, for example, *AutoSuggest* will still run on the local server as these are very lightweight.

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

5.1.5 Extending Support with SOLR

SOLR includes powerful full-text search, hit highlighting, faceted search, dynamic clustering, database integration, rich document — for example, Word, PDF — handling, and geospatial search. SOLR is highly scalable, and provides both distributed search and index replication.

If you require an even more scalable index and greater query power, you can extend the item buckets implementation to work with SOLR. If enabled, queries are sent to SOLR to run and the *History* engine will call SOLR every time an update, insertion, or deletion is made to the index. If you require an indexing server that can scale to many servers, you should consider using SOLR.

5.1.6 Installing SOLR on Windows

This section describes how to install SOLR on Windows.

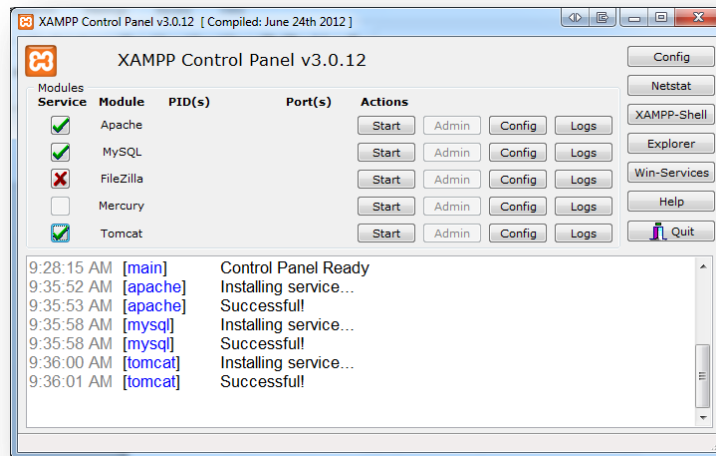
Download the following software:

- Download *Xampp For Windows*, Basic Package.
<http://www.apachefriends.org/en/xampp-windows.html>
- Download *Java JDK*
<http://java.sun.com/javase/downloads/index.jsp>
- Download *Apache Solr* from one of the mirrors.
<http://www.proxytracker.com/apache/lucene/solr/>
There is currently no current support for SOLR 4.0.
- Download the *Solr PHP Client*.
<http://code.google.com/p/solr-php-client/>

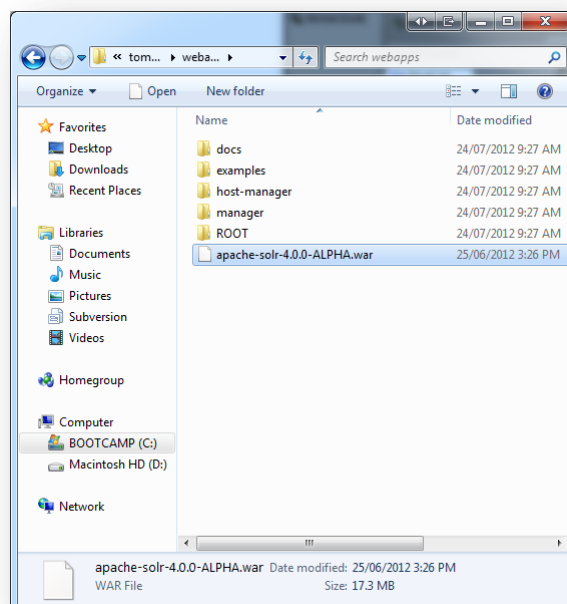
To install SOLR:

1. Install all the software that you downloaded and use the defaults settings.

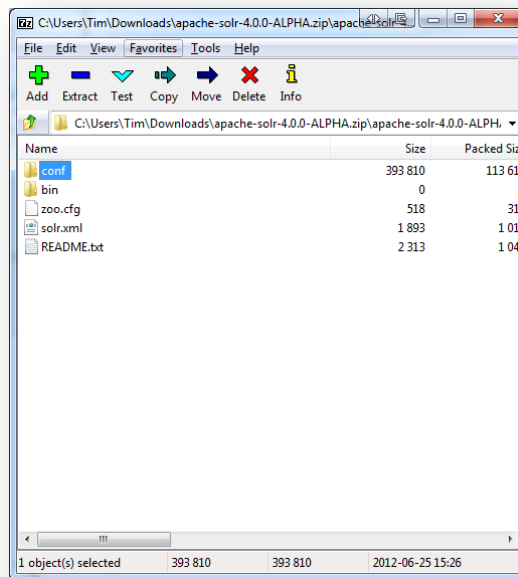
2. Run *Xampp* and enable the Apache, MySQL, and Tomcat services.



3. Copy the *.war file into the `c:\xampp\tomcat\webapps` folder from your *Apache-Solr* download. Make sure the Apache Tomcat Service is not running currently.



4. Create a directory called *solr* under `c:\xampp` and copy the entire `examples/solr` directory in the `apache-solr.zip` file to this directory.



5. On `C:\`, create a folder called *solr*. Copy the *etc*, *lib*, *logs*, *solr*, *webapps*, and *start.jar* folders from the `apache-solr.zip` file to `C:\solr`.
6. Open the `C:\solr\solr` folder and copy the contents back to the root `C:\solr` folder.
7. Open *Powershell* or `CMD.exe` and navigate to the `c:\solr` folder and run:


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

You must restart this every time the server resets or SOLR crashes.
8. Open Sitecore and use the installation wizard to install the `Sitecore.ItemBucket.BigData.SOLR.zip` package.

When the package is installed, you should see the *SOLR Dashboard* in the Start menu.
9. Open the *Core* database and locate the `sitecore/content/applications/SOLR` item.
10. Ensure that the application points to the URL of your site, for example, `http://<sitename>:8983/solr/`
11. 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.
12. Add `style="background-color:white;"` to the `<body>` tag.
13. Open the SOLR application in the start menu.

Every update that you make to your local file system index is replicated in the SOLR index as well.

Every query is now redirected to use the SOLR server instead of the local index. You can always have some application pools that use the local index and some that use SOLR.

5.1.7 Replicating the 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 describes one possible architecture for a master/slave environment.

The `Sitecore.ItemBuckets.SOLR` package includes some sample config files that you can use to setup distributed searching. Once you have setup this configuration, restart SOLR to see the new options in the admin screen.

Master Server Configuration

Navigate to the `conf` folder and open the `solrconfig.xml` file and insert your server details:

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

5.1.8 Slave Server Configuration

After you have installed SOLR on another server on your network, open the `solrconfig.xml` and add the following code to 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>
```

5.1.9 Repeater Server Configuration

After you have set up a master and potentially many slave servers, you may need to set up a server that takes the 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>
```

5.1.10 Running SOLR on Startup

The SOLR service must run all the time to make queries.

To run SOLR when a computer starts, 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

5.1.11 Generate a Schema.xml file

You must create a `schema.xml` file for SOLR to know which fields are in your index and what type they are.

To generate a `schema.xml` file:

1. In Sitecore, open the **Control Panel**.
2. Click **Item Buckets**.
3. Click **Generate a SOLR Schema.xml file** and then select the indexes that you want this done for.

5.1.12 Extending Support with Hadoop Clusters

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

Chapter 6

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
- Known Issues

6.1 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 depending on how many items are in the content tree and how many similar searches have been processed.

Setup Tweaks

- Keep the number of facets to a minimum. If you more than 100 and you will see a small degradation of performance.
- When you import a lot of content programmatically, you must truncate the *PublishingQueue*, *History* and *Event Queue* tables in the *Master* and *Web* databases and rebuild the indexes on the database tables otherwise your Sitecore installation will probably not start. If you don't do this, the *PublishingQueue*, *History*, and *EventQueue* tables will get very large and this will slow down processing.

To clear the tables, you must rebuild the index and run a smart publish instead of an incremental.

Content Author Tips

- You can use wildcards to search for IDs.

For example, if you know the first 4 characters of the GUID of an item, you can type, for example, `id:c728*` and click search and it will find the corresponding item.

Environment Tweaks

- If possible, disable the inbuilt Windows Search Index as well as any other indexer that is running on the computer that runs the index or the web server itself. This uses essential Disk I/O that Lucene.net needs.
- Don't run processes on the index to create a backup. This is an expensive operation and the index will probably be out of date by the time the backup is complete.
- It is very important that you set up a SQL Maintenance task that rebuilds your indexes. When you create a lot of content, index fragmentation will increase, especially with bulk import of content.

The hotspots will be the *Items*, *Versioned*, *Unversioned*, *Shared*, *Blobs*, and *Links* tables. 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 the indexes in 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 you import a lot of content into Sitecore, it is best to use the `BulkUpdateContext` class. When you have finished importing the content, rebuild the Lucene index.
- If you import a lot of content, do it in batches of, for example, 1000 and then bucket or re-sync the bucket to avoid overloading the process with items.

Bucket Config Tweaks

- You can tweak your index so that it doesn't index certain things that you don't want in the index. This decreases rebuild time and improves search time.
- Consider rebuilding your indexes on a computer 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 deploys them to an environment, ensure that that computer has an SSD. Indexes will not be so big, so a small SSD will suffice — for example 64GB.
- Don't shard too many indexes. The Item Buckets module has to context switch between them and this slows down search time.
- If you have very large caches, you can 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, enable Debug mode in the config file. In Debug mode all the queries are logged as well as how long they take to run and how many clauses they contain. This can help identify the issue. Wildcard and range queries are probably the main culprits.
- Optimize the out-of-the-box indexes. The module has been designed for it and optimization 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 the config file.
- Disable all the dropdowns that you are not using in the `/sitecore/system/Modules/Item Buckets/Settings/Search Box Dropdown` item. *Recently modified* and *recently created* are the most expensive lookups.
- Add all the items in `/Sitecore/System/Modules/ItemBuckets` to your prefetch cache.
- If you have disabled Debug Mode but would like to debug a single query, in the search field enter `debug:1`, press tab and enter the search term. Now Sitecore will only add that search query to your log file.
- Ensure that the **Hidden Items** option is not selected.

Scaling Tweaks

- To use indexes for large amounts of data, check the BigData DLL. Look at the `InMemoryIndex` which can hold parts of your content tree in an Index in RAM. The `InMemoryIndex` disappears when the application stops but is there when the application starts again. The `InMemoryIndex` is much quicker but does not store data.

6.2 Known Issues

- The copy/move/transfer functionality in the context menu is only supported for item buckets on Internet Explorer.
- When you try to turn a protected item into an item bucket and add a search tab, you must refresh the item to see the search tab.
- When you mark a template as bucketable, you must click the item again to reload it and see the changes in the ribbon.
- By default, Sitecore does 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 click an item in the search results to open it in a new tab, a **Save and Close** button and a **Close** button appear on the **Home** tab. These buttons do nothing.
- Sitecore indexes the HTML tags in your fields by default. These tags are searchable and appear in your search results.
For example, if you search for *strong* and one of the HTML fields contains a `` tag, it appears in the search results.
- In FireFox, when you close an item tab, none of the other tabs are selected automatically.
- When you use the author filter in a search to retrieve a list of content authors, you must first type the domain that the user belongs to.
- If an item does not have a version in any language then it will not appear in the search results. An item must contain content before it can appear in the search results.
- If your installation contains items in multiple languages, search results always contain the latest copy of the versions on that item regardless of which language is the context language. If you would like to search for a specific language, you must use the *language:* filter when you search through the UI.
- When you use the **Search** button on the context menu to add a search tab, you must click the item again to make the tab appear.
- In the *Bucketlist* field type, if the user clicks to see the next page of results, Sitecore tries to fetch the results even if there are no items to fetch. An empty list is displayed to the user.
- If you put the config file in the wrong directory, the item bucket buttons appear in the ribbon but do nothing.
- Field Filter on Bucket List doesn't work.
- Grid View in Bucket Link and RTE has styling issues.
- The UI is designed mainly for Firefox and Chrome and there are some styling issues in IE.