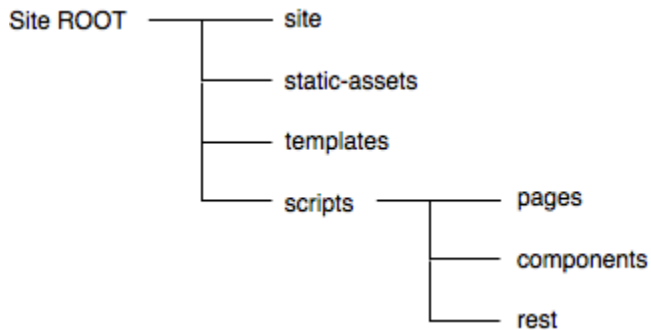


# **Building Services and Controllers for Crafter Engine with Groovy using Crafter Studio**

- Groovy in Crafter Engine
  - File Locations
  - Page and Component Scripts
    - How to add scripts to page or component forms
  - Rest Service Scripts
  - Executing a Query from Groovy
  - Available Resources
  - How do I look up a spring bean from groovy
  - Service Extension
- Help
  - Scripts Folder Creation
  - Content Type Creation
  - Adding Scripts Node Selector

# Groovy in Crafter Engine

## File Locations



Groove scripts are located in scripts directory from individual site ROOT as shown above. Groovy scripts in the 'scripts/pages' and 'scripts/components' are included in page and component XMLs, respectively, and used to generate data model for templates. Other scripts in the 'scripts/rest' directory work as REST services. All files must have .groovy file extension.

- /scripts/pages: included in page XMLs
- /scripts/components: included in component XMLs
- /scripts/rest: REST services

Note: your site might not have the script directory created by default. In this case, refer to [Help - Scripts Folder Creation](#).

## Page and Component Scripts

Those groovy scripts located under /scripts/pages and /scripts/components can be added to page or component contents (XMLs), respectively, and help generating additional data for page rendition. Page scripts run right before the engine renders templates, so the XML DOM and services are available within scripts. To include any groovy scripts to pages and components, corresponding forms must be updated first to allow user selection.

## How to add scripts to page or component forms

See [Help - Adding Scripts Node Selector](#). If you want to start from a brand new form, see [Help - Content Type Creation](#).

First we will need to have a groovy script to upload. Below is an example that finds all top-level folders and put into a model.

### site-map.groovy

```
def topNavItems = []
def siteDir = siteItemService.getSiteTree("/site/website", 2)
if(siteDir) {
    def dirs = siteDir.childItems
    dirs.each { dir ->
        def dirName = dir.getStoreName()
        def dirItem =
siteItemService.getSiteItem("/site/website/${dirName}/index.xml")
        if (dirItem != null) {
            def dirDisplayName = dirItem.queryValue('internal-name')
            topNavItems.put(dirName, dirDisplayName)
        }
    }
}
model.topNavItems = topNavItems;
```

Save this file as 'site-map.groovy'. You can also download this script [here](#). Upload the script using the following steps.

1. Open the site dropdown and click 'Scripts'
2. Right-click on 'page' folder and click 'Upload'
3. Browse to find site-map.groovy on your local file system and select the file.
4. Click 'Upload'.
5. You can update the file by repeating these steps.

Once the file is available in repo, it can be included in a page. Let's create a page and include the script.

1. Open the site dropdown and click 'Pages'
2. Right-click on 'Home' and select 'New Content'
3. Select 'Site Map' (or any content type you have added a scripts node selector)
4. Enter all required properties and go to Groovy Scripts selector
5. Click 'Add' and then 'Browse for Existing'
6. Select 'site-map.groovy' in the list and click 'Add Item'
7. Save & Close the form

After this step, now the script is a part of your page created and ready to provide the first level folders in model. In order to see the result, update sitemap.ftl (or the corresponding template for your content type)

1. Open the site dropdown and click 'Templates'
2. Browse to your template file and right-click
3. Select 'Edit'
4. Insert the following code within the body area

### sitemap.ftl

```
<p>
    <ul>
        <#list topNavItems?keys as prop>
        <li><a href="${prop}">${topNavItems[prop]}</a></li>
        </#list>
    </ul>
</p>
```

Click 'Update' View the page in preview. The page should now list top-level folders.



- [Featured Article](#)
- [Search](#)
- [Site Map](#)
- [GRC Solutions](#)
- [About](#)
- [Industry Solutions](#)
- [Resources](#)
- [Contact Us](#)

## Rest Service Scripts

You can also create a standalone service call using Groovy. Below shows how to create a REST service call using a similar code as site-map.groovy.

1. Create a script and name the file in the format of service-name.http-method.groovy. For example, a script getting site map should be name as site-map.get.groovy

### site-map.get.groovy

```
def result = [:]
def topNavItems = [:]
def siteDir = siteItemService.getSiteTree("/site/website", 2)
if(siteDir) {
    def dirs = siteDir.childItems
    dirs.each { dir ->
        def dirName = dir.getStoreName()
        def dirItem =
siteItemService.getSiteItem("/site/website/${dirName}/index.xml")
        if (dirItem != null) {
            def dirDisplayName = dirItem.queryValue('internal-name')
            topNavItems.put(dirName, dirDisplayName)
        }
    }
}
result.topNavItems = topNavItems;
return result;
```

You can also download the code [here](#).

2. Open the site dropdown and click 'Scripts'
3. Right-click on rest folder and click upload
4. Browse to select the file from your local file system

5. Upload the file
6. Call the service. The URL will be as <http://preview-domain/api/1/services/service-name.response-format>  
e.g. <http://127.0.0.1:8080/api/1/services/site-map.json>

Crafter Engine currently supports JSON or XML response format by specifying a desirable response format at the end of the service call URL. (.xml or .json). It's also possible to put service scripts into sub directories and path to the location will become a part of service call URL. (e.g. /rest/navigation/site-map.get.groovy can be accessed by /api/1/services/navigation/site-map.json)

## Executing a Query from Groovy

Just to provide another example that is a little more complex, below you will find a query execution to back a REST service

### **/scripts/rest/products/jeans.get.groovy**

```
def result = [:]

def queryStatement = "crafterSite:\"rosie\" ";
queryStatement += "AND content-type:\"/component/jeans\" ";

def query = searchService.createQuery();
query = query.setQuery(queryStatement);

def executedQuery = searchService.search(query);
def productsFound = executedQuery.response.numFound;
def products = executedQuery.response.documents;

result.products = products;
result.productsFound = productsFound;

return result;
```

## Available Resources

Crafter Engine provides multiple built-in objects that can be used in groovy scripts.

Name	Use
application	ServletContext
request	HttpServletRequest
response	HttpServletResponse
params	request parameters
headers	request headers
cookies	request cookies
session	request session

logger	a standard logger. debug logging can be enabled by adding below to log4j.xml  <pre>&lt;logger name="org.craftercms.engine.scripting"&gt;   &lt;level value="debug" /&gt; &lt;/logger&gt;</pre>
locale	the current locale (LocaleContextHolder.getLocale())
model	model accessible in FTL
crafterModel	dom4j object containing page or component XML content
authentication	the current authentication object
profile	the user profile. This object doesn't exist if crafter profile is not enabled.

Crafter Engine also provides 4 built-in core objects: `siteItemService`, `urlTransformationService`, `searchService` and `applicationContext`.

You can also access the current `SiteContext` from a script, with `AbstractSiteContextResolvingFilter.currentContext`.



In 2.5, this has changed to `SiteContext.current`.

## How do I look up a spring bean from groovy

`applicationContext.BEAN_NAME`

EXAMPLE

```
context.applicationContext["crafter.searchService"]
```

## Service Extension

It's possible to add custom built service beans and use in groovy scripts. One way is to extend `creater.restScriptsVariables` bean and add more services.

### scripting-context.xml

```
<util:map id="crafter.restScriptsVariables">
  <entry key="siteItemService" value-ref="crafter.siteItemService" />
  <entry key="urlTransformationService"
value-ref="crafter.urlTransformationService" />
  <entry key="searchService" value-ref="crafter.searchService" />
  <entry key="applicationContext" value-ref="crafter.applicationContextAccessor"
/>
</util:map>
```

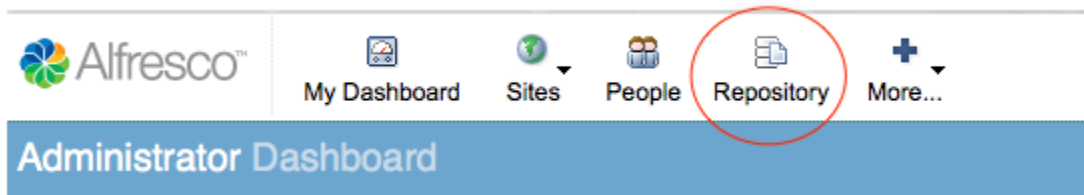
However, this is not necessary since all beans can be located using application object in scripts.

## Help

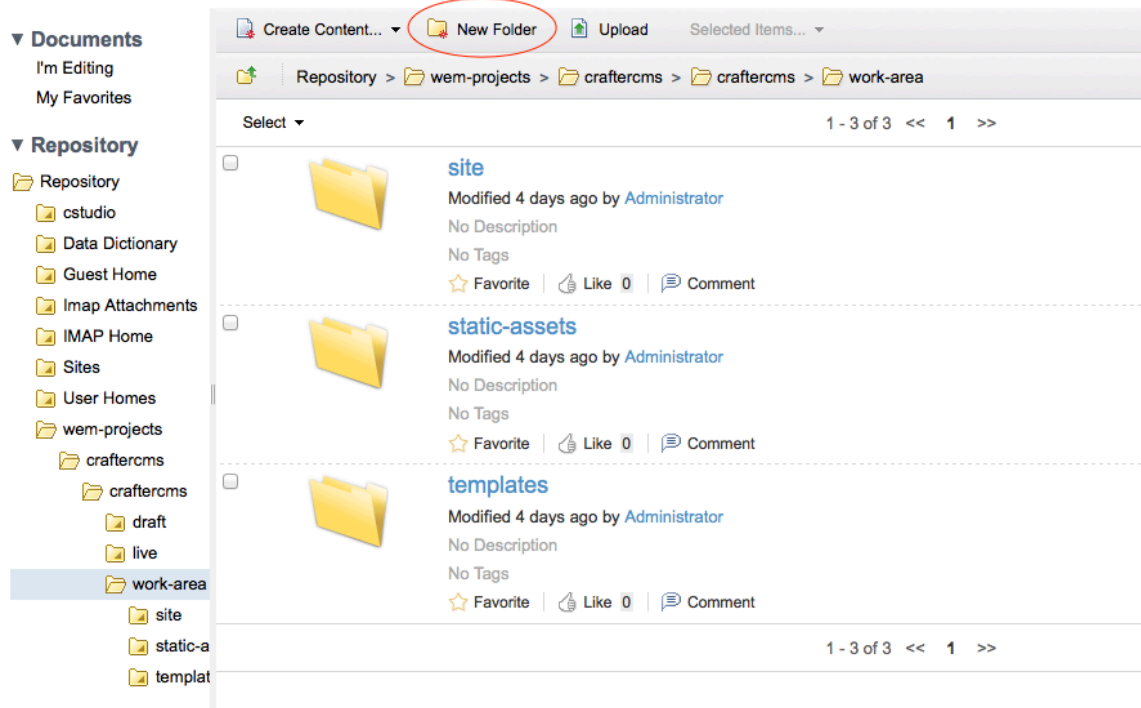
## Scripts Folder Creation

Site might not have the script directory created by default. In this case, create the directory by following the steps below.

1. Login into share as a user with admin privileges
2. Click 'Repository' in the top menu icons

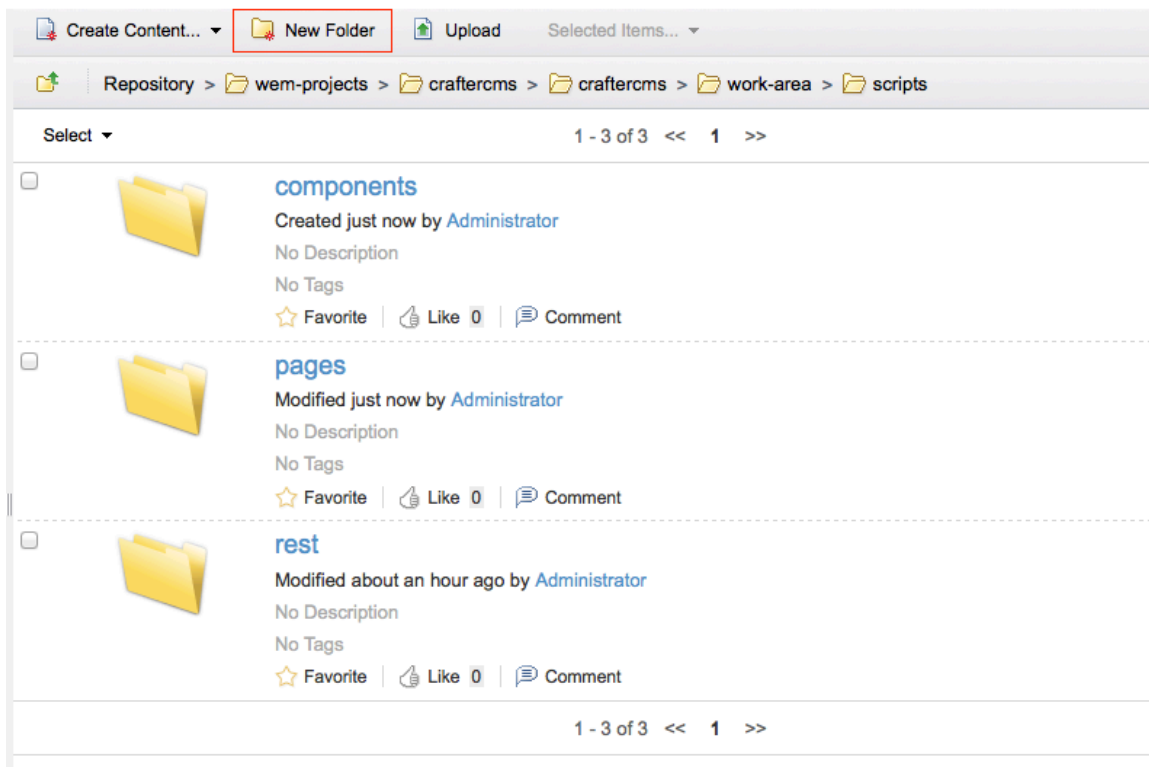


3. Browse to Repository/wem-projects/SITENAME/SITENAME/work-area. Create a 'scripts' folder by clicking 'New Folder'.

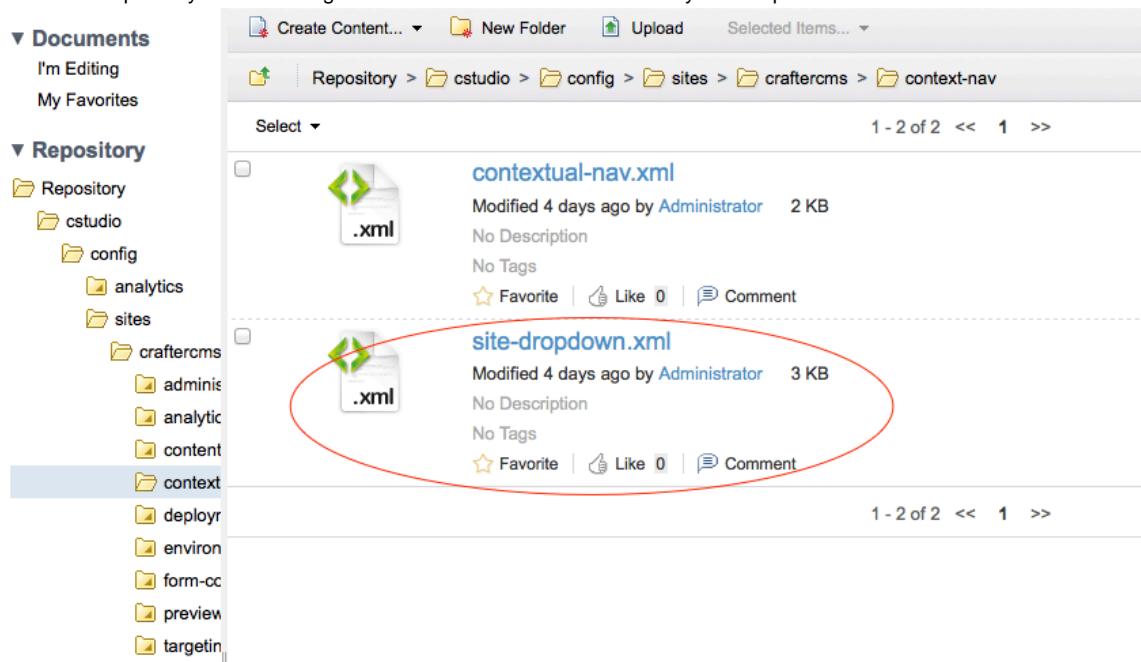


4. Browse into the scripts folder. Create 3 child folders: components, pages and rest using 'New Folder' icon again

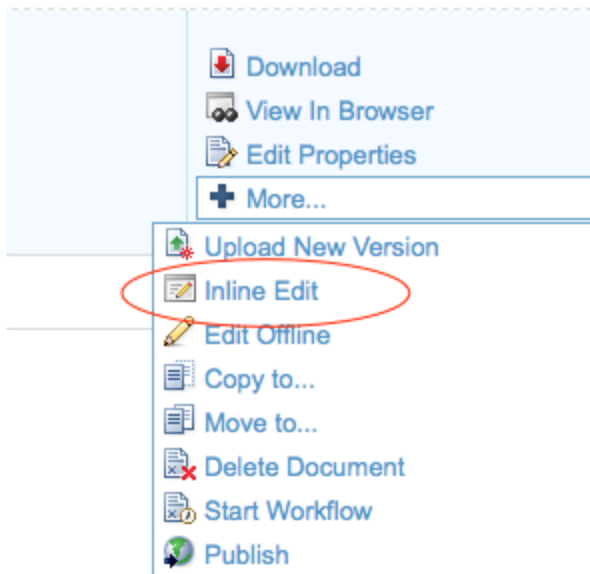




5. Browse to Repository/cstudio/config/sites/SITENAME/context-nav. Identify site-dropdown.xml



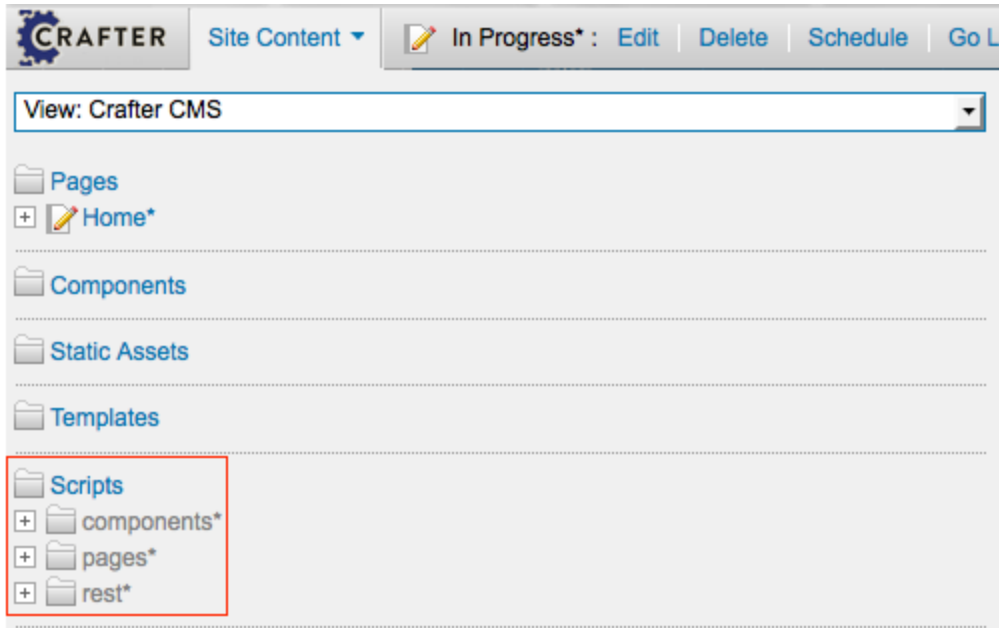
6. Select 'More..' and 'Inline Edit' on site-dropdown.xml



7. Insert the following block under 'Templates' modulehook. Click 'Save'

```
<modulehook>
  <name>wcm-assets-folder</name>
  <showDivider>true</showDivider>
  <params>
    <label>Scripts</label>
    <path>/scripts</path>
    <!-- showRootItem>false</showRootItem -->
    <onClick>none</onClick>
  </params>
</modulehook>
```

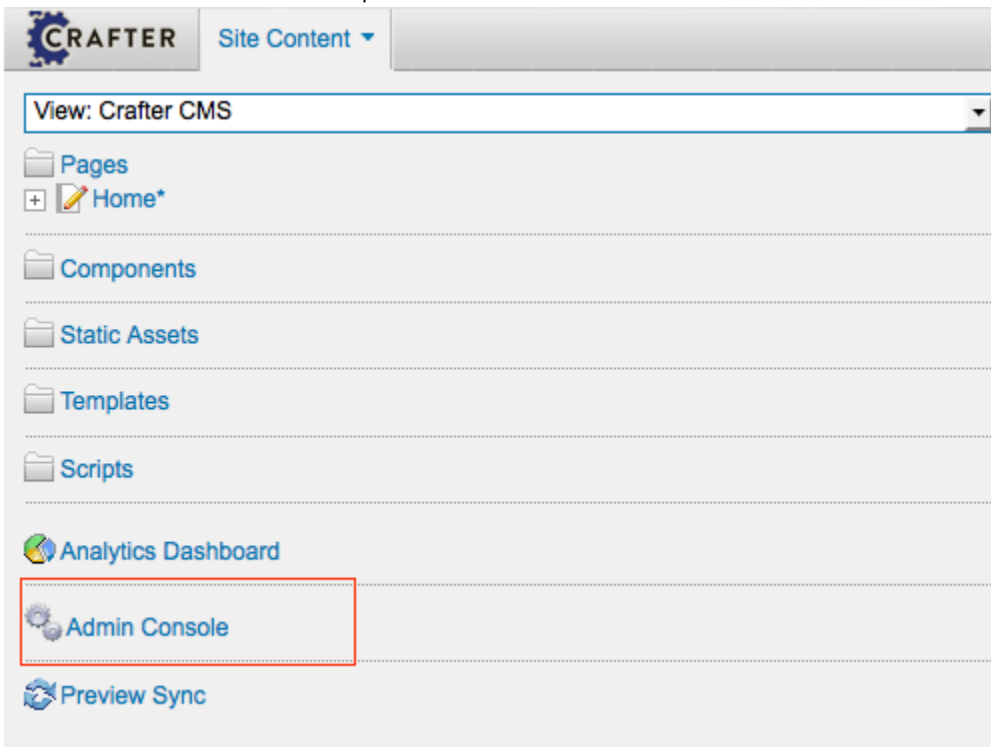
8. Go back to the Site Dashboard and refresh. Scripts folder should be now visible in the site dropdown.



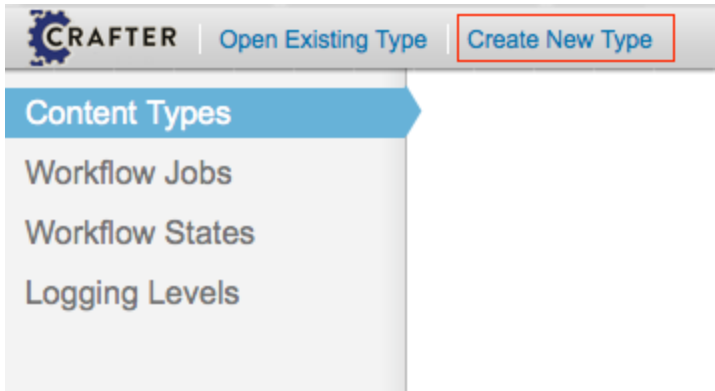
## Content Type Creation

A content type in Crafter CMS is a blueprint of pages or components that share the same content structures. New content types can be created using Crafter Studio Admin Console at anytime. **This example is using a site created from 'corporate' site blueprint. The page template must be changed in order to work with your site.**

1. Go to Admin Console from the site dropdown menu.



2. Select 'Content Types' and then 'Create New Type'





3. Enter content type properties as show below and click 'Create'

4. Select 'Site Map' section

5. Enter Properties as show below and click the pencil icon in 'Display Template' property. This will allow you to create a template add link to the content type.

Property Explorer

Form Basics	
Title	Site Map
Description	
Object Type	page
Content Type	/page/sitemap
Display Template	 
Merge Strategy	inherit-levels

- Enter a template name to be 'sitemap.ftl' and click 'Create'

## Create Template

Provide a filename for the template

sitemap.ftl|

Create

Cancel

- Copy & paste [this file content](#) into the dialog and Save. The template file name should be populated as shown below.

Property Explorer

Form Basics	
Title	Site Map
Description	
Object Type	page
Content Type	/page/sitemap
Display Template	/templates/web/sitemap.ftl
Merge Strategy	inherit-levels

- Add page properties by drag & drop controls

### Site Map

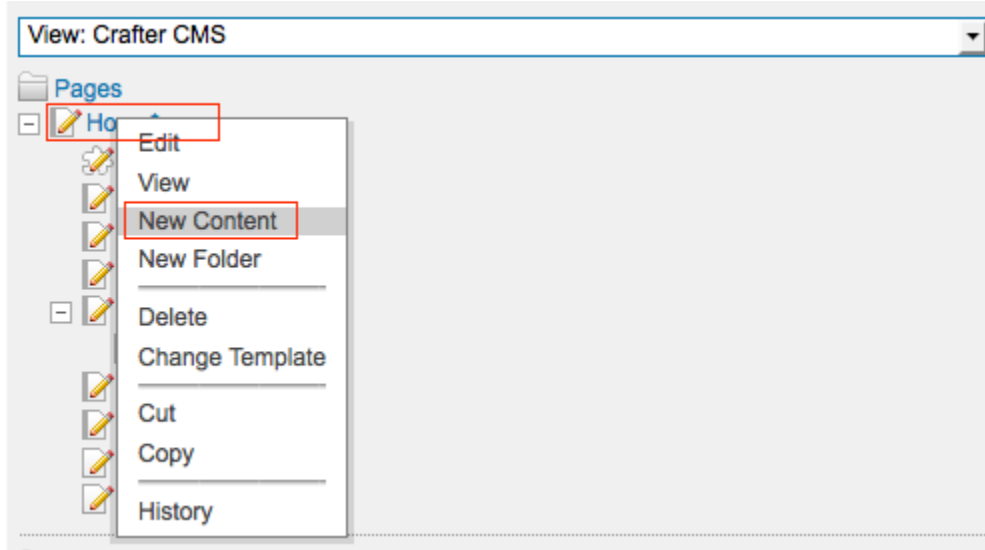
Site Map Properties

Page URL	Input	file-name
Internal Title	Input	internal-name
Title	Input	title
Body	rte	body

Data Sources

Control	Title	Variable / Name	Display Size	Max Length	Required
File Name	Page URL	file-name	100	100	Yes
Input	Internal Title	internal-name	100	100	Yes
Input	Title	title	100	100	Yes
Rich Text Editor	Body	body			No

9. Click 'Save' at the bottom of the screen
10. To confirm the content type created, go back to the site dashboard
11. Right-click on 'Home' and select 'New Content'



12. Enter page contents and click 'Save & Preview'

13. Confirm the preview of the page

[GRC SOLUTIONS](#)

[ABOUT](#)

[INDUSTRY SOLUTIONS](#)

[RESOURCES](#)

[CONTACT US](#)

C



Global Integrity

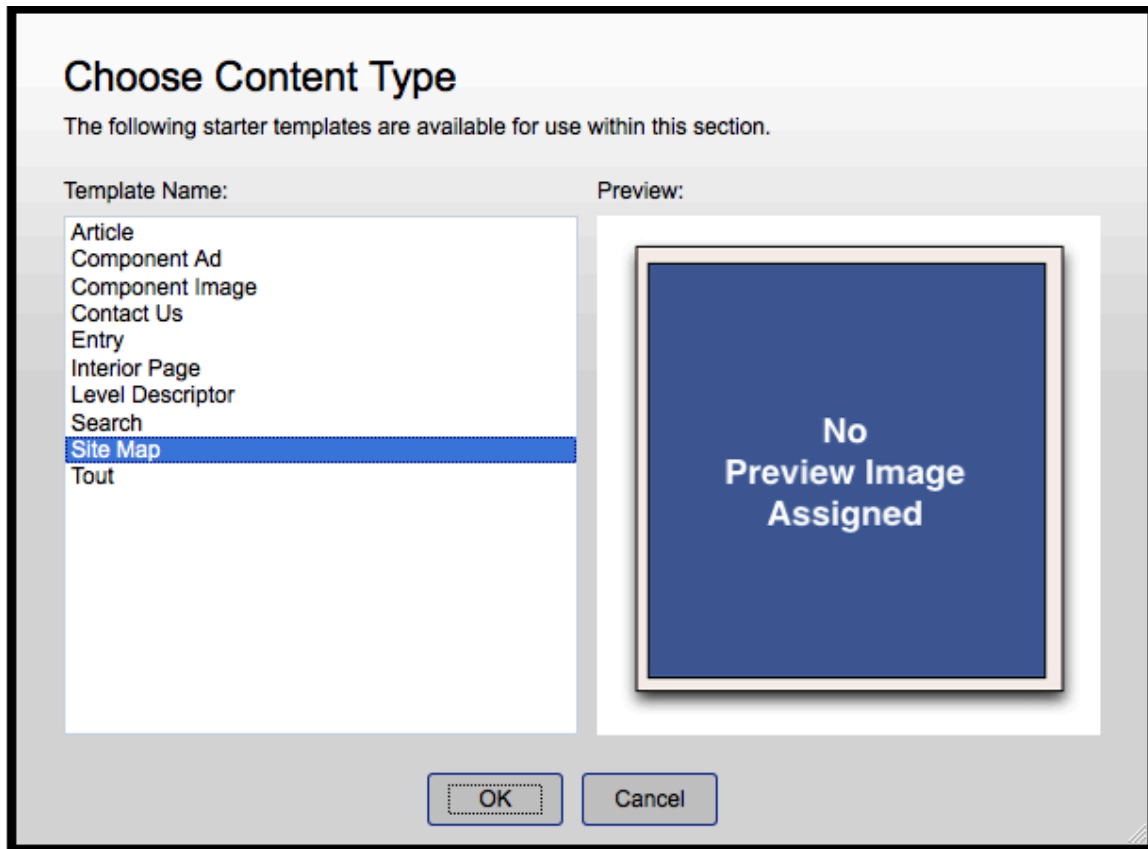
[Site Map](#)

Global

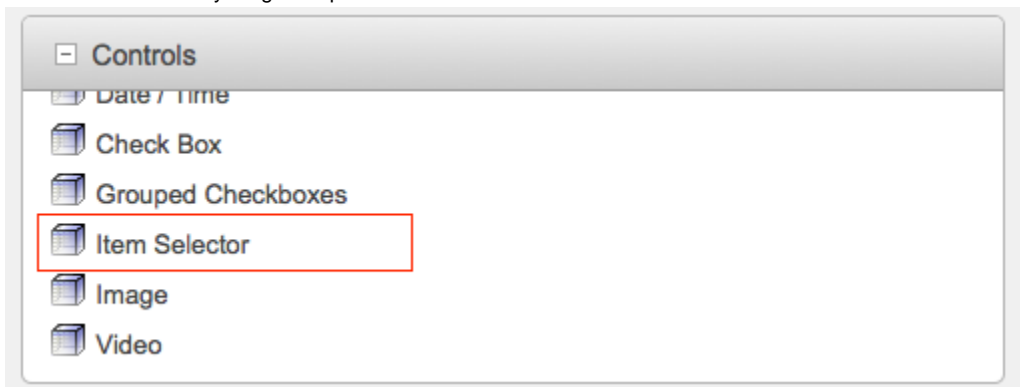
Maecenas tempor laoreet congue. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis ac dolor et justo sagittis dictum at non diam. Mauris at scelerisque velit. Aenean mollis ultrices justo, varius porta erat ornare ac. Etiam eu dignissim ipsum. Praesent tempus tempus neque, sit amet lacinia sem tempor vel. Nam nec viverra dolor. Nullam venenatis venenatis tortor egestas iaculis. Sed vitae elementum diam. Cras bibendum, ipsum id accumsan rhoncus, mauris tellus sagittis mauris, sed fermentum urna nulla non leo. Nunc tempor, mauris eu eleifend pharetra, ante est porta arcu, at condimentum arcu nisi vel dui. Praesent eget turpis nec leo scelerisque pellentesque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nam eu porttitor augue.

- ## Adding Scripts Node Selector

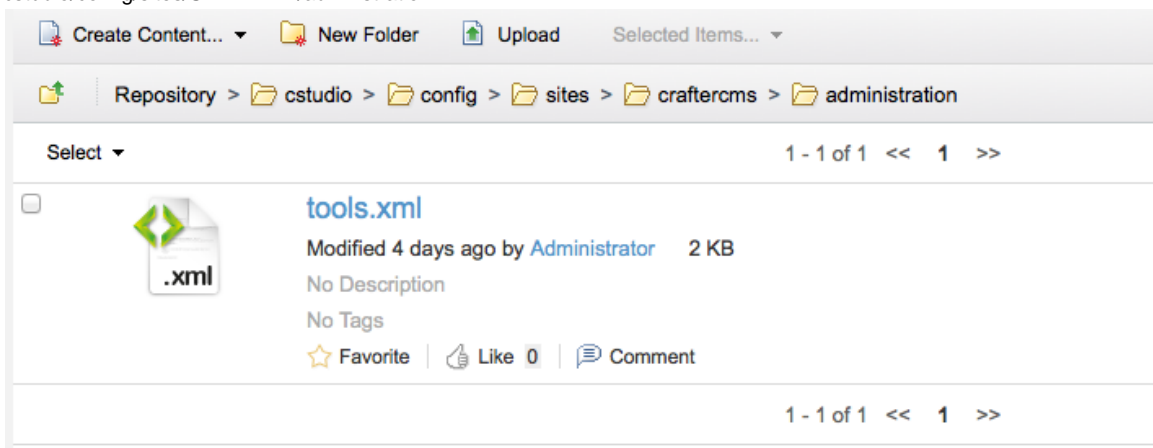
1. Go to 'Admin Console'
2. Select 'Content Types' and 'Open Existing Type'
3. Select the content type to add scripts and click 'OK'



4. Create a datasource by drag & drop 'File Browse' in the Datasources list to Data Sources section in the form

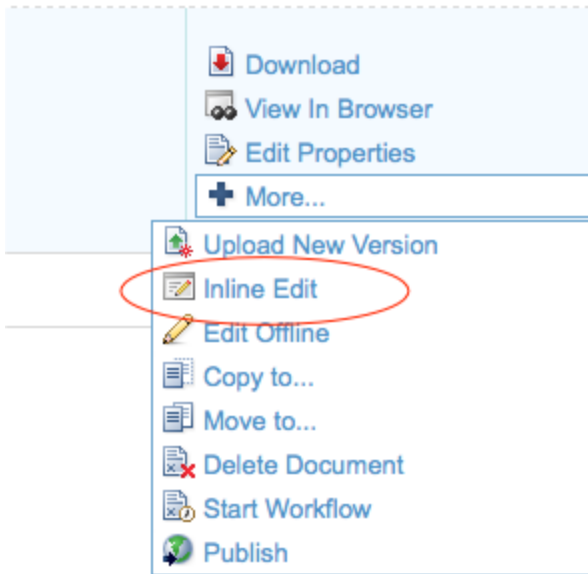


- a. If File Browse control is not available, you will need add it to tools configuration. Go to Repository in Share and browse to cstudio/config/sites/SITENAME/adminstration



- b. Select 'More..' and 'Edit Inline' next to 'tools.xml'





c. Insert `<datasource>file-browse-repo</datasource>` below configured-list

Content:

```
<control>file-name</control>
<control>auto-filename</control>
</controls>
<datasources>
  <datasource>child-content</datasource>
  <datasource>img-desktop-upload</datasource>
  <datasource>img-flickr-upload</datasource>
  <datasource>file-desktop-upload</datasource>
  <datasource>flash-desktop-upload</datasource>
  <datasource>video-desktop-upload</datasource>
  <datasource>key-value-list</datasource>
  <datasource>configured-list</datasource>
  <datasource>file-browse-repo</datasource>
</datasources>
<objectTypes>
```

Save Cancel

d. Save and refresh Admin Console

5. Select the datasource created

Data Sources		
Desktop Images	img-desktop-upload (Image)	desktopImages
	file-browse-repo (Item)	

6. Enter properties as shown below. Repository Path is where the system can find groovy scripts

Property Explorer

Datasource Basics

Title	Groovy Datasource
Name	groovy-ds

Properties

Repository Path	/scripts/pages
-----------------	----------------

- Add an item selector control to the page properties section

Controls

Date / Time

Check Box

Grouped Checkboxes

Item Selector

Image

Video

- Select the node selector created

Article Properties

Page URL	file-name	file-name
Internal Title	input	internal-name
Title	input	title
Place in Nav	page-nav-order	placeInNav
Body	rte	body_html
Author	input	author
Publish Date	date-time	publishDate_dt
Author Image	image-picker	authorImage
node-selector		

- Enter properties. The name must be 'scripts'. Select 'Groovy DataSource' for Item Manager

