

Mura Core Developer's Training

STEPHEN J. WITHTINGTON, JR.

Housekeeping

- Hi, I'm Steve

Housekeeping

- Hi, I'm Steve
- Restrooms

Housekeeping

- Hi, I'm Steve
- Restrooms
- Breaks & Lunch
 - *Fill out lunch order form!*

Housekeeping

- Hi, I'm Steve
- Restrooms
- Breaks & Lunch
 - *Fill out lunch order form!*
- Parking Lot (for questions/ideas)

Prerequisites

- Content Manager's Training
- Theme Developer's Training
- SQL DBMS Knowledge
- OOP Knowledge
- CFML (and .CFCs) is Helpful

Before we begin ...

- Download:
 - <https://github.com/stevewithington/mura-training>
 - Follow **README** instructions

Before we begin ...

- Make sure we can all access <http://localhost:8008>
- Launch your IDE (e.g., Visual Studio Code, Atom, Sublime Text, etc.), and create a new project or open the folder located under:

`.. /Desktop/mura-training/www/`

Mura Core Developer's Training

- Introduction
- Where Mura is Installed
- Mura Directory Structure
- Mura Scope
- Mura Events
- Mura Beans & Objects
- Mura Rendering
- Mura Plugins
- Where to Go From Here
- Getting Support

Introduction

WELCOME TO MURA CORE DEVELOPER'S TRAINING

Help developers integrate their custom business logic and applications into Mura.

MY OBJECTIVE

Where Mura is Installed

THE MURA WEBROOT

Where Mura is Installed

- {context}
 - Where Mura is installed within the webroot

Where Mura is Installed

- {context}
 - Where Mura is installed within the webroot
 - Typically, just under the webroot
 - C:\inetpub\wwwroot\

Where Mura is Installed

- {context}
 - Where Mura is installed within the webroot
 - Typically, just under the webroot
 - C:\inetpub\wwwroot\
 - m.globalConfig('context')

Where Mura is Installed

- {context}
- m.globalConfig('context')
 - C:\inetpub\wwwroot\
 - **Returns** empty string

Where Mura is Installed

- {context}
- m.globalConfig('context')
 - C:\inetpub\wwwroot\
 - **Returns** empty string
 - C:\inetpub\wwwroot\muracms\
 - **Returns** '/muracms'

Where Mura is Installed

- {context}
- m.globalConfig('context')
 - C:\inetpub\wwwroot\
 - **Returns** empty string
 - C:\inetpub\wwwroot\muracms\
 - **Returns** '/muracms'
 - C:\inetpub\wwwroot\somedirectory\muracms\
 - **Returns** '/somedirectory/muracms'

Mura Directory Structure

AN OVERVIEW OF MURA'S FILESYSTEM

Mura Directory Structure

- The “config” Directory
- The “core” Directory
- The “sites” Directory
- The “themes” Directory
- The “modules” Directory
- The “resource_bundles” Directory
- How to Update Mura

Mura Directory Structure

```
▸ www
  ▷ admin
  ▷ config
  ▷ core
  ▷ modules
  ▷ plugins
  ▷ resource_bundles
  ▷ sites
  ▷ themes
  ◇ .gitignore
  ! .travis.yml
  ↗ Application.cfc
  { } box.json
  ≡ htaccess.txt
  ↗ index.cfm
  ≡ Installation.txt
  ≡ license.txt
  ↖ Mura_CMS_contribution_agreement.pdf
  ⓘ README.md
  ≡ robots.txt
  { } server.json
  ⚙ web.config
  ≡ web.config.txt
```

Mura Directories & Files

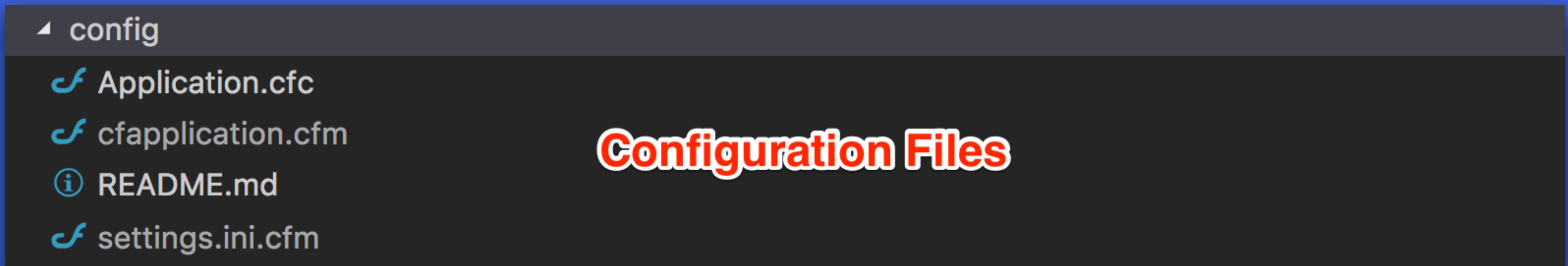
The “config” Directory

The “config” Directory

- {context}/config/

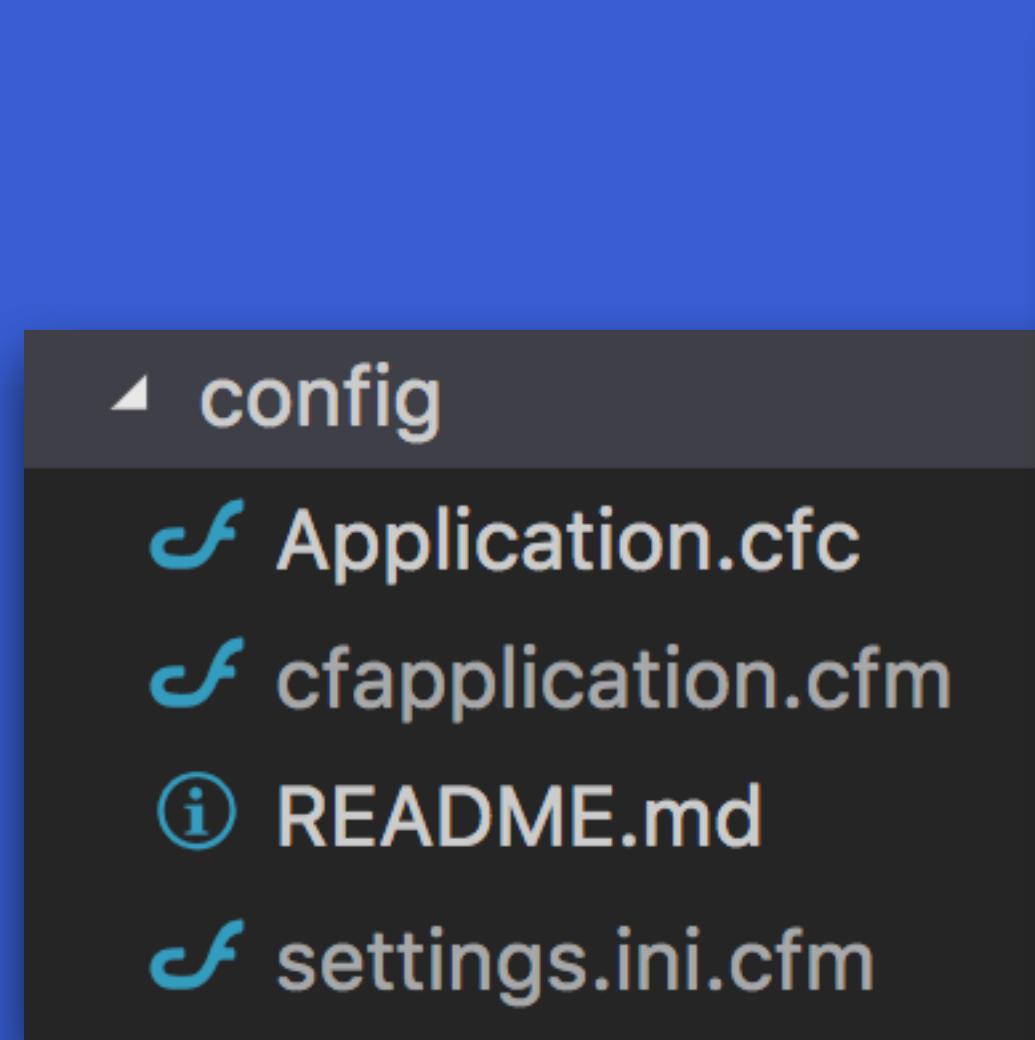
The “config” Directory

- {context}/config/



The “config” Directory

- {context}/config/



| Directory | Editable | Description |
|------------------------------------|----------|---|
| {context}/config/Application.cfc | No | This file prevents site visitors from navigating directly to any of the .CFM files contained within this directory. |
| {context}/config/cfapplication.cfm | Yes | Experienced developers may add custom Application.cfc variables and/or ColdFusion/CFML mappings here. |
| {context}/config/README.md | No | This file contains information about the files contained in this directory. |
| {context}/config/settings.ini.cfm | Yes | This is an extremely important file, and contains many of the default settings for Mura. View the settings.ini.cfm reference guide for details. |

The “settings.ini.cfm” File

The “settings.ini.cfm” File

- Contains many configuration settings for Mura

The “settings.ini.cfm” File

- Contains many configuration settings for Mura
- Simple text file with key-value pairs

The “settings.ini.cfm” File

- Contains many configuration settings for Mura
- Simple text file with key-value pairs
- For example:

```
1. [production]
2. adminemail=steve@blueriver.com
3. dbtype=mysql
4. yourCustomKey=Your Value
```

The “settings.ini.cfm” File

- Reference:

<http://docs.getmura.com/v7-1/mura-developers/mura-directory-structure/the-config-directory/settings-ini-cfm/>

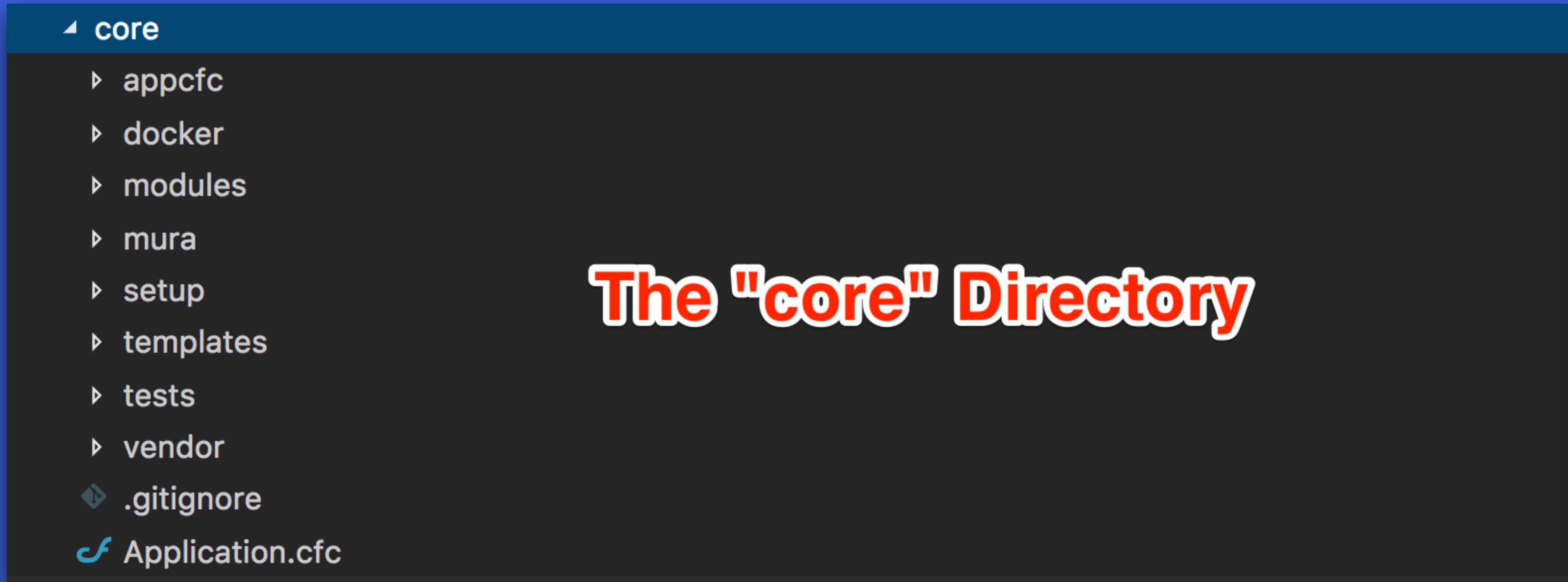
The “core” Directory

The “core” Directory

- {context}/core/

The “core” Directory

- {context}/core/



The “core” Directory

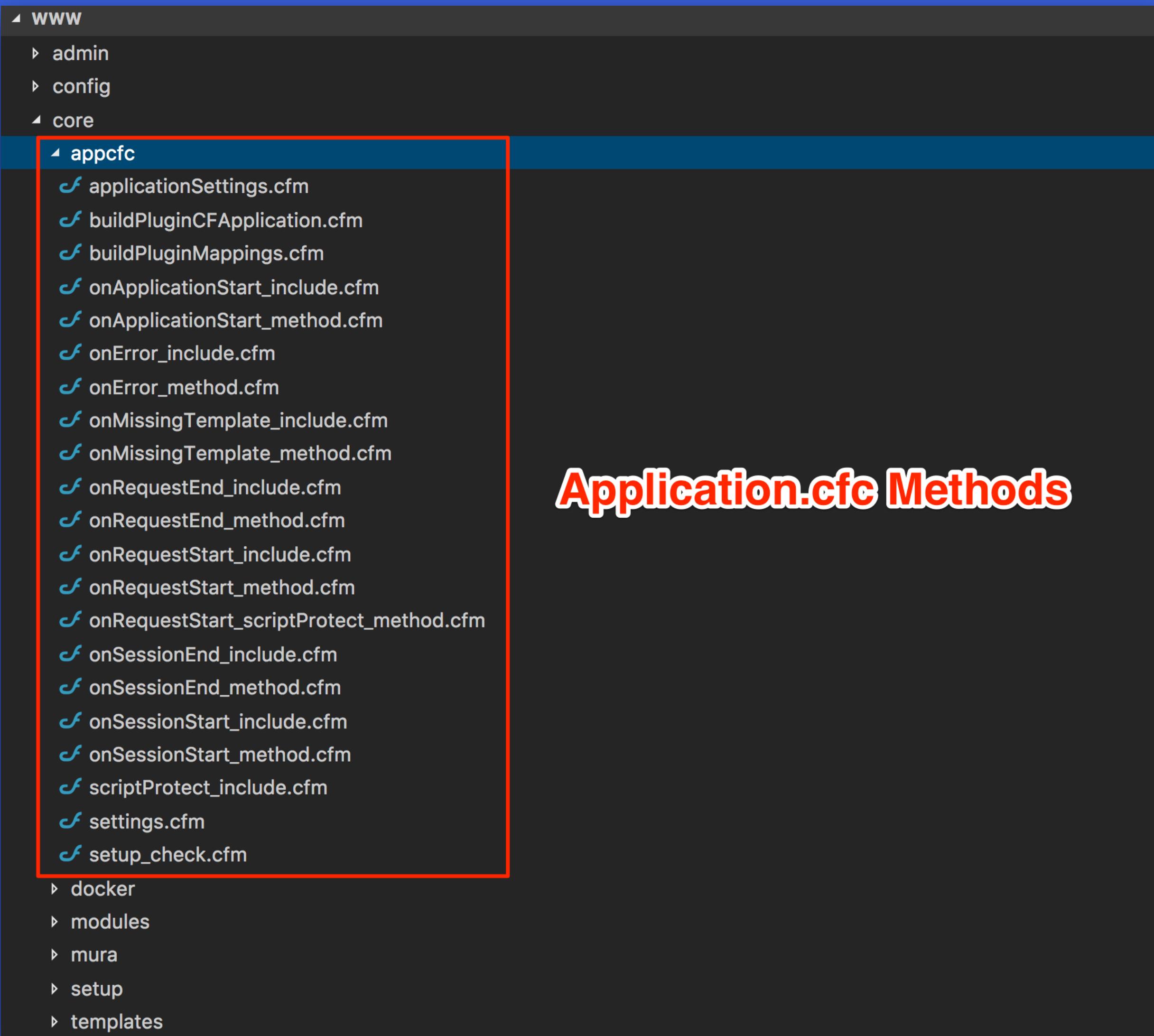
| Directory or File | Editable | Description |
|--------------------------------|----------|---|
| {context}/core/appcfc/ | No | The files contained in this directory are included in Mura's main Application.cfc file. See The "appcfc" Directory section for details. |
| {context}/core/docker/ | No | This directory contains files related to running Mura on Docker . |
| {context}/core/modules/ | No | The directories and files located here represent the default display objects and modules used by Mura. This is covered more extensively in the Mura Modules/Display Objects section . |
| {context}/core/mura/ | No | The files located here are essentially the beating heart of Mura. We'll take a deeper look at some of its directories and files later. |
| {context}/core/setup/ | No | This directory contains files required during the installation process. |
| {context}/core/templates/ | No | This directory contains "templates" used to create core files such as the settings.ini.cfm , robots.txt , and some site files such as the Site contentRenderer.cfc , and Site eventHandler.cfc . These files are typically only used during the installation process. |
| {context}/core/tests/ | No | This directory contains files for conducting unit tests. |
| {context}/core/vendor/ | No | This directory contains third-party code which Mura relies on for various parts of its functionality such as CKEditor . |
| {context}/core/.gitignore | No | The Mura Team uses Git for distributed version control, and this is the site .gitignore file used to specify intentionally untracked files. |
| {context}/core/Application.cfc | No | This particular file merely prevents users from navigating to any of the .cfm or .cfc files directly via the browser. |

The “appcfc” Directory

The “appcfc” Directory

- Contains code used in Mura’s root Application.cfc

The “appcfc” Directory



The “appcfc” Directory

```
boolean function onApplicationStart() output=false {  
    param name="application.instanceID" default=createUUID();  
    lock name="appInitBlock#application.instanceID#" type="exclusive" timeout="200" {  
        include "/muraWRM/core/appcfc/onApplicationStart_include.cfm";  
    }  
    return true;  
}
```

The “sites” Directory

The “sites” Directory

- {context}/**sites/**

The “sites” Directory

- {context}/sites/

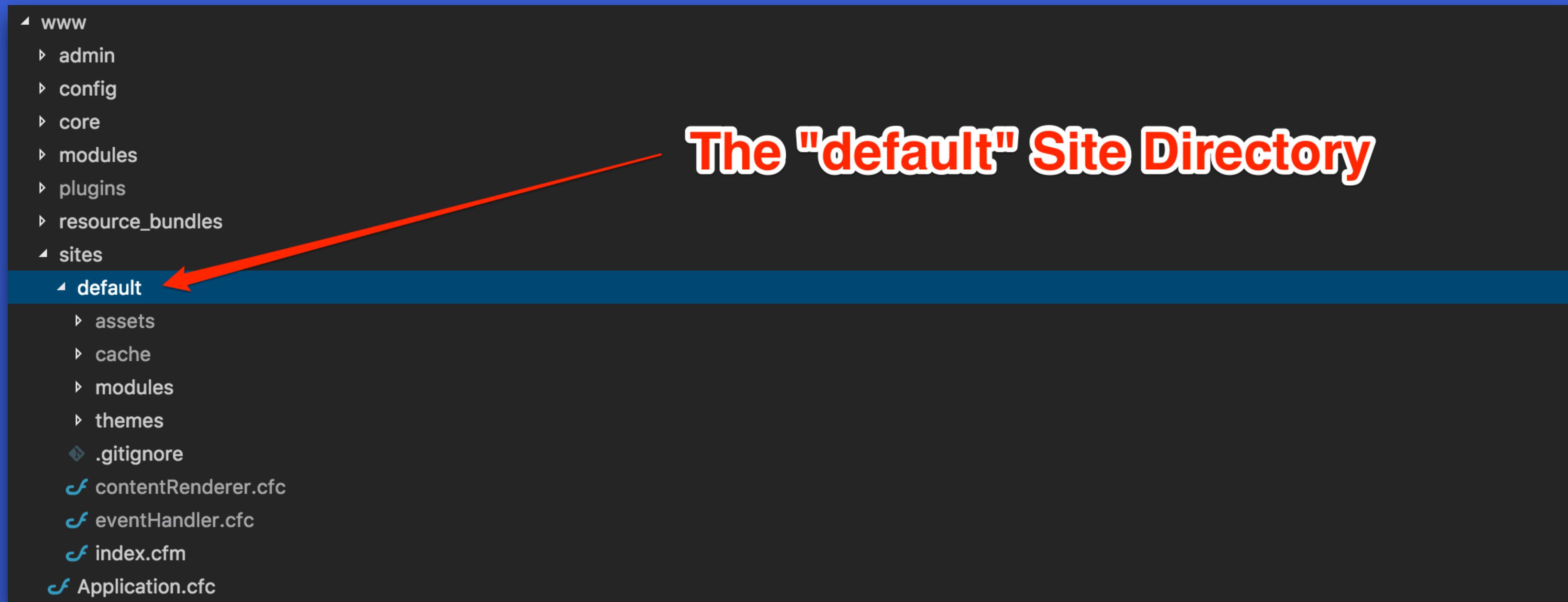
| Directory or File | Editable | Description |
|---------------------------------|----------|---|
| {context}/sites/Application.cfc | Yes | Required. |
| {context}/sites/default/ | Yes | Required. Each Mura installation also has a minimum of one site, labeled the "default" directory. The "default" directory name itself should not be modified, nor should any other Mura-generated directory name located under the "sites" directory. Doing so will break functionality. See below for more information on the "default" directory. |

The “sites” Directory

- {context}/sites/**default**/

The “sites” Directory

- {context}/sites/**default**/

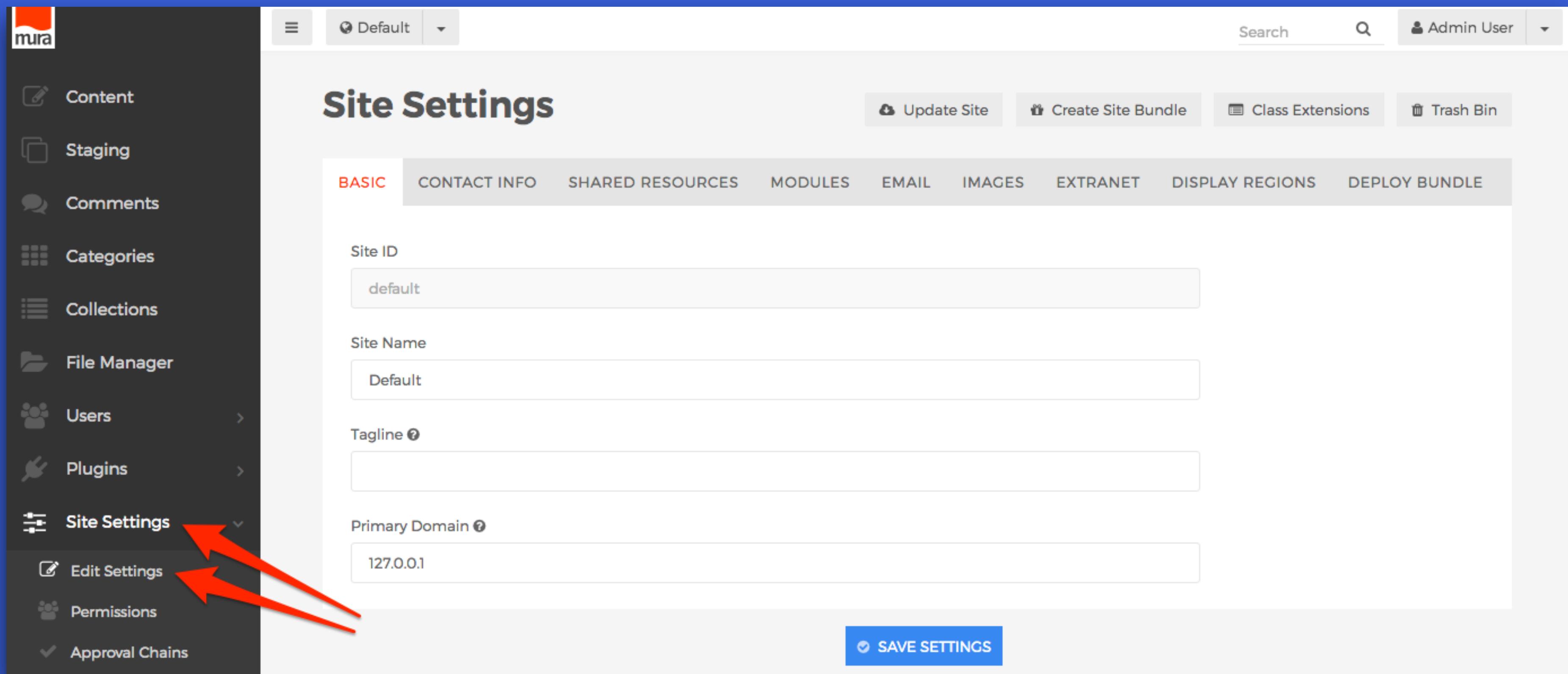


The "default" Site Directory

```
▶ www
  ▶ admin
  ▶ config
  ▶ core
  ▶ modules
  ▶ plugins
  ▶ resource_bundles
  ▶ sites
    ▶ default
      ▶ assets
      ▶ cache
      ▶ modules
      ▶ themes
      ◆ .gitignore
      ✓ contentRenderer.cfc
      ✓ eventHandler.cfc
      ✓ index.cfm
      ✓ Application.cfc
```

The “sites” Directory

- {context}/sites/{SiteID}/



The “sites” Directory

- {context}/sites/{SiteID}/

The screenshot shows the Mura CMS Site Settings page. On the left is a dark sidebar with various navigation items: Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings (which is expanded to show Edit Settings, Permissions, and Approval Chains), and a bottom item with a checkmark. The main area has a header with tabs: BASIC (highlighted with a red arrow), CONTACT INFO, SHARED RESOURCES, MODULES, EMAIL, IMAGES, EXTRANET, DISPLAY REGIONS, and DEPLOY BUNDLE. Below the tabs are several input fields: Site ID (containing "default", highlighted with a red box), Site Name (containing "Default"), Tagline (empty), Primary Domain (containing "127.0.0.1"), and a blue "SAVE SETTINGS" button at the bottom right.

The “sites” Directory

- {context}/sites/{SiteID}/
 - assets/
 - cache/
 - modules/
 - resource_bundles/
 - themes/
 - contentRenderer.cfc
 - eventHandler.cfc
 - index.cfm

The “themes” Directory

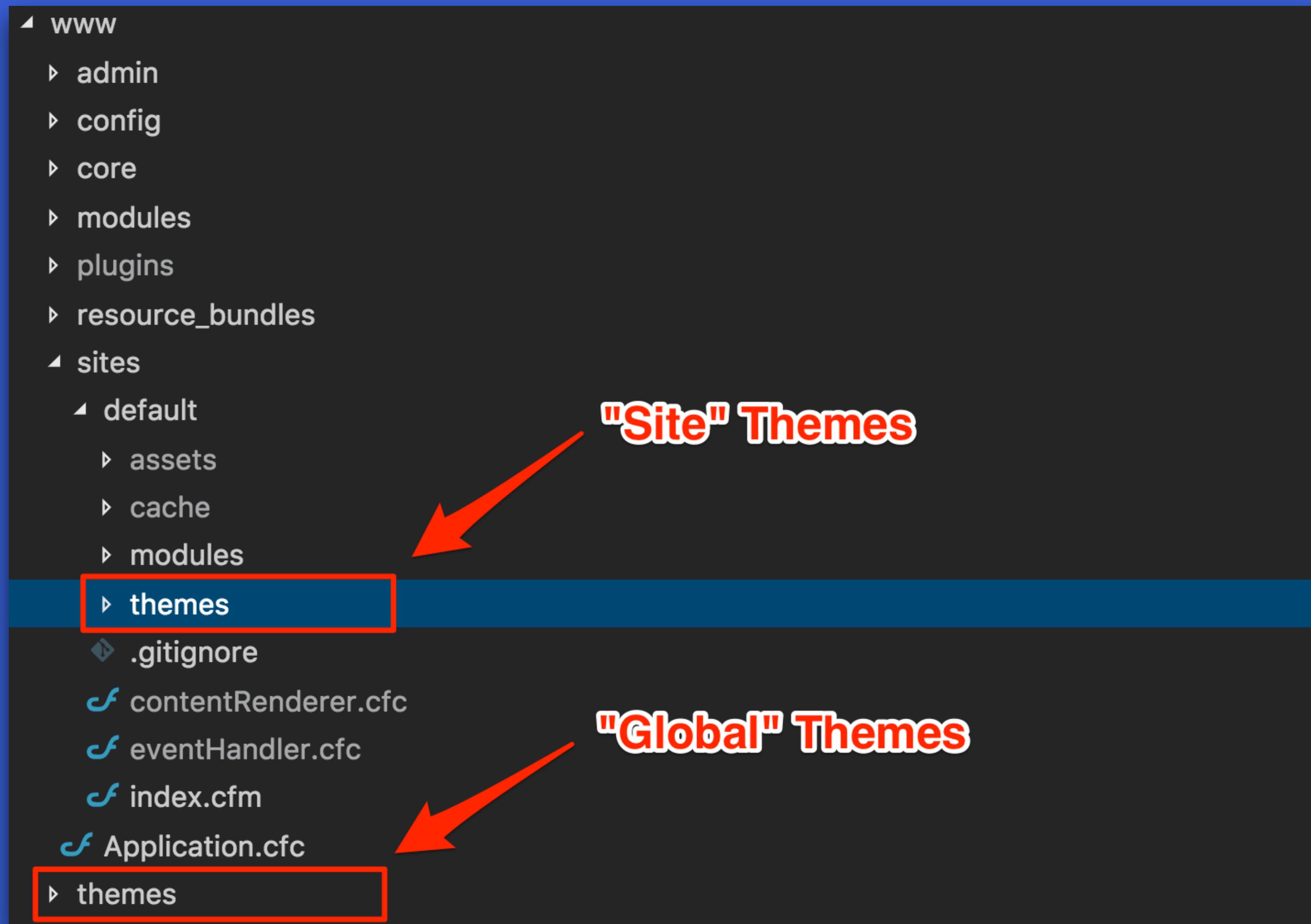
The “themes” Directory

- **Global Themes**
 - `{context}/themes/`
 - Themes located under the “Global” themes directory will automatically be available as a theme option for all sites under the same Mura instance

The “themes” Directory

- **Global Themes**
 - `{context}/themes/`
 - Themes located under the “Global” themes directory will automatically be available as a theme option for all sites under the same Mura instance
- **Site Themes**
 - `{context}/sites/{SiteID}/themes/`
 - Themes located under a site’s “themes” directory are available to the specified site, and/or any other sites sharing the same “Display Object Pool”

The “themes” Directory



The “themes” Directory

- Any subdirectories of the “themes” directory are automatically registered as a theme in Mura

The “themes” Directory

- Any subdirectories of the “themes” directory are automatically registered as a theme in Mura
- The directory name will be listed as an option for authorized admins to apply as a theme to their site(s)

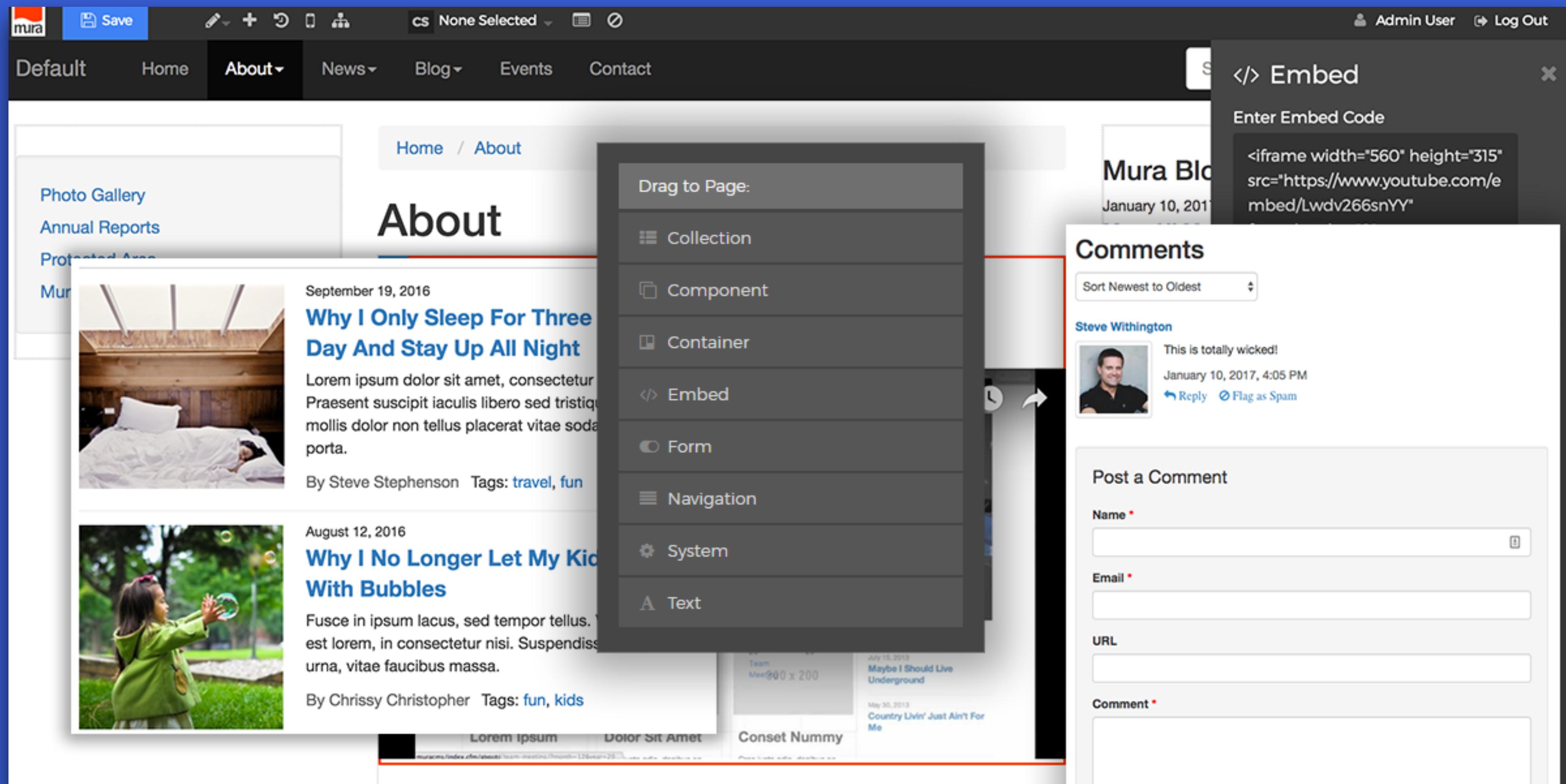
The “themes” Directory

- Any subdirectories of the “themes” directory are automatically registered as a theme in Mura
- The directory name will be listed as an option for authorized admins to apply as a theme to their site(s)
- When Mura is first installed, if a theme does not exist, Mura will attempt to download a theme using the “`defaultthemewurl`” setting found in the `{context}/config/settings.ini.cfm` file

The “modules” Directory

RIDICULUS MALESUADA

The “modules” Directory



The “modules” Directory

- Lookup Hierarchy
 - Registered Module Directory
 - Module
 - Theme
 - Site
 - Global
 - Core

The “modules” Directory

- Lookup Hierarchy
 - **Registered Module Directory**
 - A pre-registered module directory path
 - We'll cover more in Modifying Mura's Modules under the Mura Rendering section

The “modules” Directory

- Lookup Hierarchy
 - **Module**
 - ../{module}/modules/
 - A nested “modules” directory nested within a known module

The “modules” Directory

- Lookup Hierarchy
 - **Theme**
 - ../{ThemeName}/modules/
 - Theme modules are only used when the specified theme is actively assigned to a site

The “modules” Directory

- Lookup Hierarchy
 - **Site**
 - {context}/sites/{SiteID}/modules/
 - Site Modules are shared across all themes within the specified site

The “modules” Directory

- Lookup Hierarchy
 - **Global**
 - {context}/modules/
 - Global Modules are shared across all sites under a single Mura instance

The “modules” Directory

- Lookup Hierarchy
 - **Core**
 - {context}/core/modules/v1/core_assets/modules/
 - These are the “core” modules or display objects which may be copied and pasted into any of the previous directories to be safely modified

The “modules” Directory

- Lookup Hierarchy
 - **Core**
 - {context}/core/modules/v1/core_assets/modules/
 - These are the “core” modules or display objects which may be copied and pasted into any of the previous directories to be safely modified
 - You should ***not*** edit the “core” modules directory

The “modules” Directory

```
  ▲ www
    ▷ admin
    ▷ config
    ▲ core
      ▷ appfc
      ▷ docker
      ▲ modules
        ▲ v1 ←
          ▷ calendar
          ▷ category_summary
          ▷ collection
          ▷ comments
          ▷ component
          ▷ container
          ▷ core_assets
          ▷ datacollection
          ▷ dataresponses
          ▷ deny
          ▷ editprofile
          ▷ embed
          ▷ event_reminder_form
          ▷ favorites
          ▷ feed
          ▷ feedslideshow
          ▷ folder
          ▷ form
          ▷ formbuilder
          ▷ forward_email
          ▷ gallery
          ▷ gotoFirstChild
          ▷ htmlhead
          ▷ login
          ▷ mailing_list
          ▷ mailing_list_master
          ▷ media
```

The "core" modules

The “resource_bundles” Directory

The “resource_bundles” Directory

- Mura utilizes resource bundles to internationalize various areas of the user interface, making the code locale-independent

The “resource_bundles” Directory

- Mura utilizes resource bundles to internationalize various areas of the user interface, making the code locale-independent
- Resource bundles are .properties files, located under specific directories within Mura

The “resource_bundles” Directory

- Mura utilizes resource bundles to internationalize various areas of the user interface, making the code locale-independent
- Resource bundles are .properties files, located under specific directories within Mura
- The files are named to indicate the language code, and country code

The “resource_bundles” Directory

- Mura utilizes resource bundles to internationalize various areas of the user interface, making the code locale-independent
- Resource bundles are .properties files, located under specific directories within Mura
- The files are named to indicate the language code, and country code
 - For example, `en_US.properties` for “English” and “United States”

The “resource_bundles” Directory

- The file itself is comprised of key-value pairs

The “resource_bundles” Directory

- The file itself is comprised of key-value pairs
- The keys remain the same throughout each of the .properties files, and the value is translated into the file’s designated language

The “resource_bundles” Directory

- The file itself is comprised of key-value pairs
- The keys remain the same throughout each of the .properties files, and the value is translated into the file’s designated language
- If Mura is searching for a specific key-value pair within a translation and cannot locate it, Mura will fall back to the English translation



en_US.properties x

```

1 #X-Generator: crowdin.com
2 #Created by JInto - www.guh-software.de
3 #Mon Apr 20 09:41:22 PDT 2009
4 calendar.agendaWeek=Agenda Week
5 calendar.agendaDay=Agenda Day
6 calendar.day=Day
7 calendar.week=Week
8 calendar.month=Month
9 calendar.year=Year
10 calendar.today=Today
11 calendar.monthLong=January,February,March,April,May,June,July
12 calendar.monthshort=Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sept,Oct,
13 calendar.summary.on.daily=Daily
14 calendar.summary.on.weekly=Every week on {1}
15 calendar.summary.on.week1=Every 1st week of the month on {1}
16 calendar.summary.on.week2=Every 2nd week of the month on {1}
17 calendar.summary.on.week3=Every 3rd week of the month on {1}
18 calendar.summary.on.week4=Every 4th week of the month on {1}
19 calendar.summary.on.weeklast=Every last week of the month on
20 calendar.summary.on.monthly=Every month on {1}
21 calendar.summary.on.yearly=Every year on {1}
22 calendar.summary.timespan=from {1} to {2}
23 calendar.summary.datespan=from {1} to {2}
24 calendar.summary.until=until {1}
25 calendar.weekdayShort=S,M,T,W,T,F,S
26 calendar.weekdaylong=Sunday,Monday,Tuesday,Wednesday,Thursday
27 calendar.loadingevents=Loading events...
28 calendar.firstdayoftheweek=First Day of the Week
29 calendar.close=Close
30 calendar.eventfetcherror=Warning\! An error occurred while fe
31 captcha.error=The ''Security Code'' entered was not correct.
32 captcha.instructions=Please enter the code above
33 captcha.securitycode=Security Code

```

es_ES.properties x

```

1 #X-Generator: crowdin.com
2 #Created by JInto - www.guh-software.de
3 #Mon Apr 20 09:41:22 PDT 2009
4 calendar.agendaWeek=Puntos de Agenda
5 calendar.agendaDay=Agenda d\u00eda
6 calendar.day=D\u00eda
7 calendar.week=Semana
8 calendar.month=Mes
9 calendar.year=A\u00f1o
10 calendar.today=Hoy
11 calendar.monthLong=Enero,Febrero,Marzo,Abril,Mayo,Junio,Julio
12 calendar.monthshort=Ene,Feb,Mar,Abr,May,Jun,Jul,Ago,Sept,Oct,I
13 calendar.summary.on.daily=Diaria
14 calendar.summary.on.weekly=Cada semana en {1}
15 calendar.summary.on.week1=Cada semana 1 del mes en {1}
16 calendar.summary.on.week2=Cada semana 2\u00aa del mes {1}
17 calendar.summary.on.week3=Cada semana 3 del mes en {1}
18 calendar.summary.on.week4=Cada semana 4\u00aa del mes {1}
19 calendar.summary.on.weeklast=Cada \u00faltima semana del mes
20 calendar.summary.on.monthly=Cada mes en {1}
21 calendar.summary.on.yearly=Cada a\u00f1o, el {1}
22 calendar.summary.timespan=de {1} a {2}
23 calendar.summary.datespan=de {1} a {2}
24 calendar.summary.until=hasta {1}
25 calendar.weekdayShort=L,M,Mi,J,V,S,D
26 calendar.weekdaylong=Domingo,Lunes,Martes,Mi\u00e9rcoles,Juev
27 calendar.loadingevents=Eventos de carga...
28 calendar.firstdayoftheweek=Primer d\u00eda de la semana
29 calendar.close=Cerrar
30 calendar.eventfetcherror=\u00a1Advertencia\! Se ha producido
31 captcha.error=El ''C\u00f3digo de Seguridad'' que ingres\u00f3
32 captcha.instructions=Por favor ingrese el C\u00f3digo de Segu
33 captcha.securitycode=C\u00f3digo de Seguridad

```

The “resource_bundles” Directory

- Lookup Hierarchy
 - Module
 - Content Types
 - Theme
 - Site
 - Global
 - Core

The “resource_bundles” Directory

- **Note:**
 - Admin-area resource bundles are located under
`{context}/core/mura/resourceBundle/resources/`

The “resource_bundles” Directory

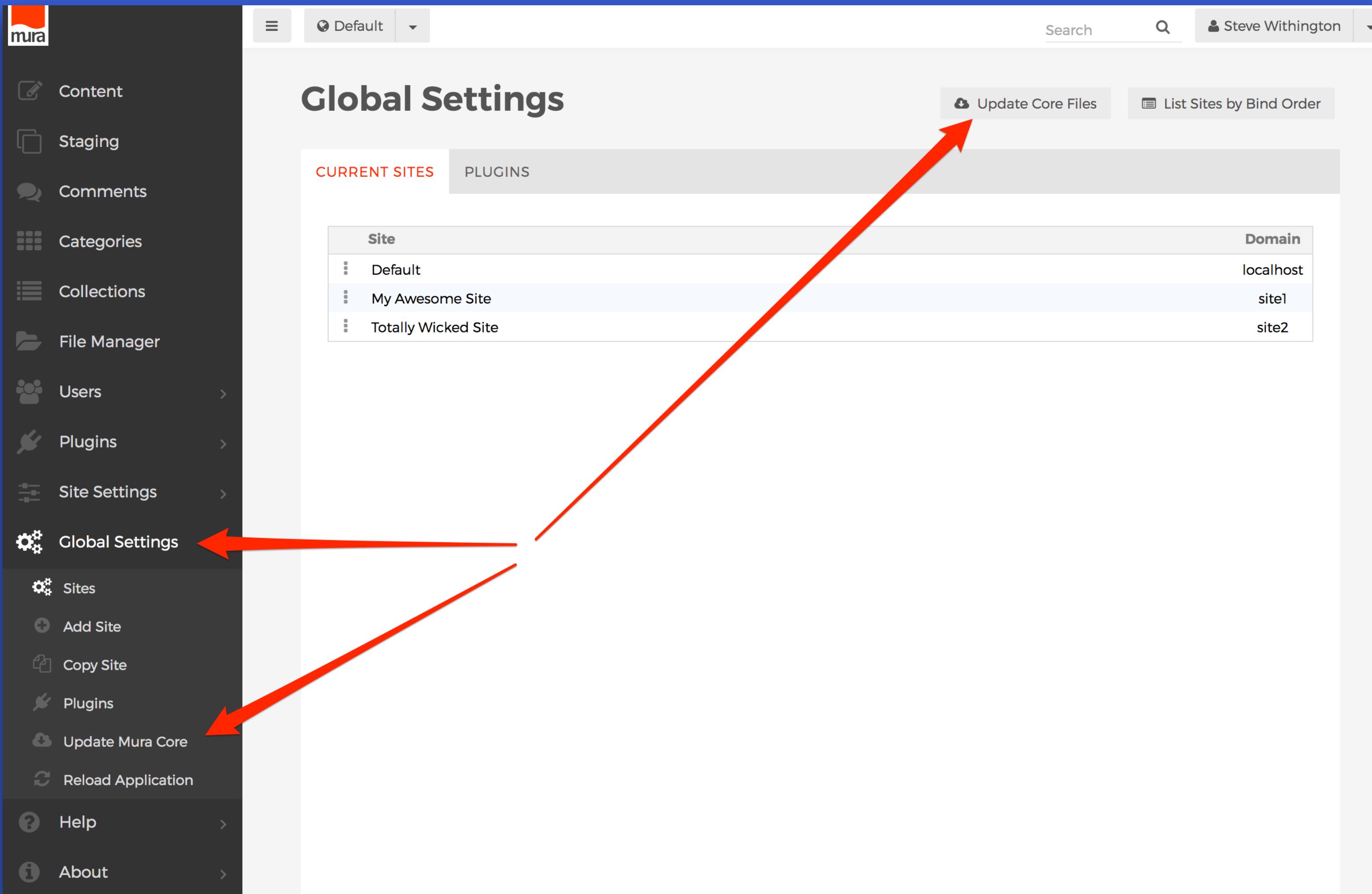
- **Note:**
 - Admin-area resource bundles are located under `{context}/core/mura/resourceBundle/resources/`
 - However, as of v7.1, many key-value pairs are not able to be overwritten using these techniques at this time.
 - Allowing for this option is under consideration for a future version.

The “resource_bundles” Directory

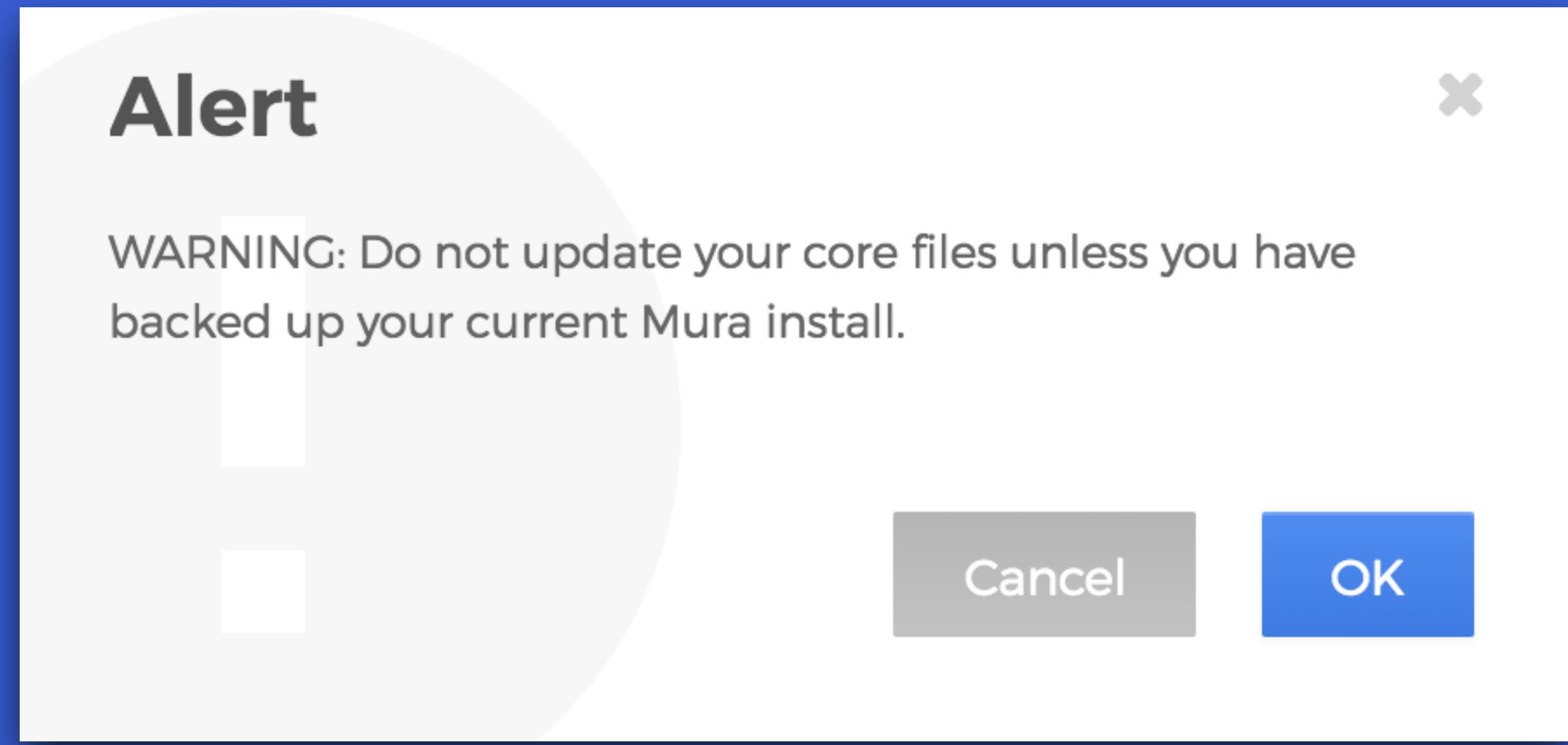
- Contribute at <https://crowdin.com/project/muracms>

How to Update Mura

How to Update Mura



How to Update Mura



How to Update Mura

The screenshot shows the Mura CMS interface. On the left is a dark sidebar with white icons and text, listing various site management options. The main area is titled "Global Settings". A prominent message box contains the text "Your core files have been updated to version 7.1.93." A large red arrow points from the bottom right towards this message box. The top right corner of the screen shows the user's name, "Steve Withington".

- Content
- Staging
- Categories
- Collections
- File Manager
- Users >
- Plugins >
- Site Settings >
- Global Settings >
 - Sites
 - Add Site
 - Copy Site
 - Plugins
 - Update Mura Core
 - Reload Application
- Help >
- About >

Default Search Steve Withington List Sites

Your core files have been updated to version 7.1.93.

How to Update Mura

The screenshot shows the Mura CMS Global Settings page. On the left is a dark sidebar with various navigation options. The main content area is titled "Global Settings" and contains a message: "Your core files have been updated to version 7.1.93." In the top right corner, there is a user profile for "Steve Withington" and a "List Sites" button. A large red arrow points from the bottom left towards the "List Sites" button.

- Content
- Staging
- Categories
- Collections
- File Manager
- Users >
- Plugins >
- Site Settings >
- Global Settings >
 - Sites
 - Add Site
 - Copy Site
 - Plugins
 - Update Mura Core
 - Reload Application
 - Help >
 - About >

Lab Exercise

Lab Exercise

- See **/3-core-developer/1-mura-directory-structure/instructions.md**

Summary

- The “config” Directory
- The “core” Directory
- The “sites” Directory
- The “themes” Directory
- The “modules” Directory
- The “resource_bundles” Directory
- How to Update Mura

Mura Scope

LET'S TALK WITH MURA

Mura Scope

- What is the Mura Scope?
- Custom Instance of the Mura Scope
- setCustomMuraScopeKey
- Lab Exercise

What is the Mura Scope?

- Similar to how CFML includes various scope types, Mura also includes its very own “Mura Scope” for accessing, and manipulating Mura variables, data, and objects.

What is the Mura Scope?

- mura
- \$
- m

What is the Mura Scope?

1. `<!-- Example from within .cfm templates -->`
2. `#m.content('title')#`
- 3.
4. `<!-- Example when using the [m] or [mura] tag -->`
5. `[m]m.content('title')[/m]`

What is the Mura Scope?

1. <!-- Inspecting the Mura Scope -->
2. <cfdump var="#m#">

Custom Instance of “m”

```
1. if ( !IsDefined('m') ) {  
2.     m = StructKeyExists(session, 'siteid')  
3.         ? application.serviceFactory.getBean('m').init(session.siteid)  
4.         : application.serviceFactory.getBean('m').init('default');  
5. }
```

Custom Instance of “m”

```
1. if ( !IsDefined('m') ) {  
2.  
3.     // Assumes you have a pre-defined event object named `yourEventObject`  
4.     m = application.serviceFactory.getBean('m').init(yourEventObject);  
5.  
6.     // OR  
7.  
8.     yourStruct = {  
9.         siteID = 'SomeSiteID'  
10.        , someKey = 'Some Value'  
11.    };  
12.  
13.    m = application.serviceFactory.getBean('m').init(yourStruct);  
14. }
```

setCustomMuraScopeKey()

```
1. m.setCustomMuraScopeKey(  
2.     name  
3.     , value  
4. )
```

setCustomMuraScopeKey()

```
1. public function onSiteRequestStart(m) {  
2.     m.setCustomMuraScopeKey('myCustomKey', 'This is my custom Mura Scope Key value!');  
3. }
```

setCustomMuraScopeKey()

```
1. <cfoutput>  
2.   <p>#m.myCustomKey#</p>  
3. </cfoutput>
```

setCustomMuraScopeKey()

```
1. | <cfoutput>  
2. |   <p>#m.myCustomKey#</p>  
3. | </cfoutput>
```

```
1. | <p>This is my custom Mura Scope Key value!</p>
```

exampleObject.cfc

```
1. component {  
2.  
3.     public function sayHello() {  
4.         return 'Hello from exampleObject.cfc!';  
5.     }  
6.  
7. }
```

setCustomMuraScopeKey()

```
1. public function onBeforeUserSave(m) {  
2.     var myObject = new exampleObject();  
3.  
4.     m.setCustomMuraScopeKey('myObject', myObject);  
5. }
```

setCustomMuraScopeKey()

```
1. <cfoutput>  
2.   #m.myObject.sayHello()#  
3. </cfoutput>
```

setCustomMuraScopeKey()

```
1. <cfoutput>  
2.   #m.myObject.sayHello()#  
3. </cfoutput>
```

```
1. Hello from exampleObject.cfc!
```

Lab Exercise

LET'S CHAT WITH MURA

Lab Exercise

- See `/3-core-developer/2-mura-scope/instructions.md`

Summary

- What is the Mura Scope?
- Custom Instance of the Mura Scope
- setCustomMuraScopeKey
- Lab Exercise

Mura Events

ADDING AND/OR REPLACING FUNCTIONALITY

Mura Events

- Lifecycle Events vs. Contextual Events
- Mura “event” Scope
- Event Hooks
- Event Lifecycles
- Event Handlers
- Mura’s Event Log
- Lab Exercise

Mura Events

- As a CFML application, Mura has CFML-specific application-wide settings, variables, and event handlers. In addition to the CFML-specific events (e.g., `onApplicationStart`, `onSessionStart`, `onRequestStart`, etc.), Mura has several of its own events, to allow developers the opportunity to add to, or in some cases, even replace, Mura's own business logic and functionality.

Lifecycle v. Contextual Events

Lifecycle v. Contextual Events

- “Lifecycle” events generally get triggered on every page request, and represent the execution of Mura’s normal request flow.

Lifecycle v. Contextual Events

- “Lifecycle” events generally get triggered on every page request, and represent the execution of Mura’s normal request flow.
- Mura consists of two unique lifecycles: the **Front-end Lifecycle**, and the **Admin Lifecycle**.”

Lifecycle v. Contextual Events

- “Lifecycle” events generally get triggered on every page request, and represent the execution of Mura’s normal request flow.
- Mura consists of two unique lifecycles: the **Front-end Lifecycle**, and the **Admin Lifecycle**.”
- “Contextual” events aren’t triggered on every page request, because they occur in response to the specific event(s) or action(s) that preceded them.

Lifecycle v. Contextual Events

- **Note:** Contextual events only contain data that is directly supplied to it by the Mura core. If you need access to the full event scope itself (e.g., during a front-end request or admin rendering event), it can be accessed via `m.getGlobalEvent()`.

Mura “event” Scope

Mura “event” Scope

- When working with events, keep CFML’s scopes in mind.
As noted in the Mura Scope section, in addition to the CFML scopes, Mura has its own scope called the Mura Scope.

Mura “event” Scope

- When working with events, keep CFML’s scopes in mind.
As noted in the Mura Scope section, in addition to the CFML scopes, Mura has its own scope called the Mura Scope.
- The Mura Scope has a special subscope called the “**event**” scope, and it’s created on every request.

Mura “event” Scope

- When working with events, keep CFML’s scopes in mind.

As noted in the Mura Scope section, in addition to the CFML scopes, Mura has its own scope called the Mura Scope.
- The Mura Scope has a special subscope called the “**event**” scope, and it’s created on every request.
 - `m.event()`

Mura “event” Scope

- Mura’s “event” scope wraps CFML’s **request**, **form**, and **URL** scopes.

Mura “event” Scope

- Mura’s “event” scope wraps CFML’s **request**, **form**, and **URL** scopes.
- Mura also injects additional data and helper methods into the “event” scope, and the availability of the data is dependent upon the context in which you may be accessing it.

Mura “event” Scope

- Mura’s “event” scope wraps CFML’s `request`, `form`, and `URL` scopes.
- Mura also injects additional data and helper methods into the “event” scope, and the availability of the data is dependent upon the context in which you may be accessing it.
- As you’ll see in the Mura Objects section, we can store our own data in Mura’s “event” scope too!

Getters

1. `m.event('attributeName')`
2. `m.event().get('attributeName')`
3. `m.event().get{AttributeName}()`
4. `m.event().getValue('attributeName')`

Setters

1. `m.event('attributeName', 'someValue')`
2. `m.event().set('attributeName', 'someValue')`
3. `m.event().set{AttributeName}('someValue')`
4. `m.event().setValue('attributeName', 'someValue')`

Event Hooks

Event Hooks

- In this section, we'll cover Mura's various event hooks/ methods, or trigger points, you can use to either add to, or even replace Mura's business logic and functionality.

Event Hooks

- Determining the point(s) in which you wish to add your logic.

Event Hooks

- Determining the point(s) in which you wish to add your logic.
- *When Mura is loading up?*

Event Hooks

- Determining the point(s) in which you wish to add your logic.
 - *When Mura is loading up?*
 - *During a normal, front-end request?*

Event Hooks

- Determining the point(s) in which you wish to add your logic.
 - *When Mura is loading up?*
 - *During a normal, front-end request?*
 - *When a content item is being updated?*

Event Hooks

- Determining the point(s) in which you wish to add your logic.
 - *When Mura is loading up?*
 - *During a normal, front-end request?*
 - *When a content item is being updated?*
 - *When a session begins?*

Event Hooks

- Once you've answered these questions, you can dig into the available event hooks, register your event handlers, and implement your custom application logic.

Event Hooks

- Once you've answered these questions, you can dig into the available event hooks, register your event handlers, and implement your custom application logic.
- Also, all events are automatically passed in the Mura (m) scope as an argument.

Event Hooks

```
1. public any function onSomeEvent(m) {  
2.     // do something here  
3. }
```

Event Hooks

```
1. public any function onSomeEvent(m) {  
2.     // obtain the `event` scope via the Mura Scope  
3.     var e = arguments.m.event();  
4. }
```

Event Hooks

```
1. public any function onSomeEvent(m) {  
2.     // access the global event scope  
3.     var globalEvent = arguments.m.getGlobalEvent();  
4. }
```

Event Hooks: Prefixes

- **standard{Event}**
 - All “**standard**” events are points where you can *replace* Mura’s business logic and functionality.

Event Hooks: Prefixes

- **standard{Event}**
 - All “**standard**” events are points where you can *replace* Mura’s business logic and functionality.
- {context}/core/mura/Handler/
standardEventsHandler.cfc

Event Hooks: Prefixes

- **on{Event}**
- All “on” events are points where you can *add to* Mura’s business logic and functionality.

Event Hooks: Prefixes

- **on{Event}**
 - All “on” events are points where you can *add to* Mura’s business logic and functionality.
 - These “on” events are not necessarily “defined” anywhere. These types of events are merely “announced” during the normal execution of a request.

Event Hooks: Prefixes

- **onGlobal{Event}**
 - All “**global**” events are those that occur throughout the entire instance of Mura, and are not bound to any specific site.

Event Hooks: Prefixes

- **onSite{Event}**
 - All “**site**” events occur site-wide, and are bound to the specific site they have been registered to.

Event Hooks: Prefixes

- **onBefore{Event}**
 - All “**before**” events are announced immediately *before* Mura is about to execute its normal functionality.

Event Hooks: Prefixes

- **onAfter{Event}**
 - All “**after**” events are announced immediately **after** Mura has executed its normal functionality.

Event Hooks

- CFML Application Events
- Standard Events
- App-Related Events
- Content-Related Events
- User-Related Events
- Custom Events

CFML Application Events

CFML Application Events

| CFML Method | Analogous Mura Event |
|--------------------|---|
| onApplicationStart | onApplicationLoad |
| onApplicationEnd | <i>Not implemented in Mura at this time</i> |
| onSessionStart | onGlobalSessionStart |
| onSessionEnd | onGlobalSessionEnd |
| onRequestStart | onGlobalRequestStart |
| onRequest | <i>Not implemented in Mura at this time</i> |
| onRequestEnd | onGlobalRequestEnd |
| onMissingTemplate | onGlobalMissingTemplate |
| onError | onGlobalError |

Standard Events

Standard Events

- Replacing Mura's business logic and functionality.

Standard Events

- Replacing Mura's business logic and functionality.
- Review {context}/core/mura/Handler/
standardEventsHandler.cfc

Standard Events

- Replacing Mura's business logic and functionality.
- Review {context}/core/mura/Handler/
standardEventsHandler.cfc
- [https://www.getmura.com/component-api/7.1/index.html?
mura/Handler/standardEventsHandler.html](https://www.getmura.com/component-api/7.1/index.html?mura/Handler/standardEventsHandler.html)

Standard Events

- Three primary suffixes for “standard” events:

Standard Events

- Three primary suffixes for “standard” events:
 - validator
 - handler
 - translator

Standard Events: Suffixes

- **validator**
 - Validators inspect the “event” or “request” itself, and forward the request/event to the appropriate handler, if necessary.

Standard Events: Suffixes

- **handler**
 - Handlers usually set values in the “global event”, or redirect the user somewhere.

Standard Events

- **translator**
 - Translators check to see if the requested `returnformat` is JSON, and if so, utilizes Mura's JSON API to handle the request.

App-Related Events

App-Related Events

- Events specifically related to the Mura application itself.

App-Related Events

- Events specifically related to the Mura application itself.
- See <http://docs.getmura.com/v7-1/mura-developers/mura-events/event-hooks/app-related-events/>

Content-Related Events

Content-Related Events

- Events specifically related to content items.

Content-Related Events

- Events specifically related to content items.
- See <http://docs.getmura.com/v7-1/mura-developers/mura-events/event-hooks/content-related-events/>

User-Related Events

User-Related Events

- Events specifically related to Mura users.

User-Related Events

- Events specifically related to Mura users.
- See <http://docs.getmura.com/v7-1/mura-developers/mura-events/event-hooks/user-related-events/>

Custom Events

Custom Events

- In addition to all of Mura's event hooks, developers may create their own, custom event hooks.

Custom Events

- In addition to all of Mura’s event hooks, developers may create their own, custom event hooks.
- The main decision you’ll want to make is whether you wish to “*announce*” an event, or “*render*” an event.

Custom Events

- Regardless of whether you choose to *announce* an event, or *render* an event, Mura will trigger your event as illustrated below, and in turn, any registered event handlers with these methods defined, will be executed.

1. | `on{YourEvent}`

Announcing Events

- When you “*announce*” an event, you’re essentially sending a notification throughout Mura that something in particular is about to occur, or has just occurred.

Announcing Events

```
1. m.announceEvent('YourEvent');  
2. // OR  
3. m.announceEvent('onYourEvent');
```

Rendering Events

- If you wish to “*render*” an event, in addition to sending a notification throughout Mura that something in particular is about to occur, or has just occurred, you are allowing for the possibility to render something to the browser.

Rendering Events

- If you wish to “*render*” an event, in addition to sending a notification throughout Mura that something in particular is about to occur, or has just occurred, you are allowing for the possibility to render something to the browser.
- In other words, your code should be checking for a string to return, and if nothing is returned, then you’ll most likely want to display something else by default.

Rendering Events

```
1. m.renderEvent('YourEvent');
```

Event Lifecycles

Event Lifecycles

- Lifecycle events are generally triggered on every page request, and represent the execution of Mura's normal request flow.

Event Lifecycles

- Mura consists primarily of two unique lifecycles:
 - The Admin Request Lifecycle
 - The Front-end Request Lifecycle

Event Lifecycles

- Mura consists primarily of two unique lifecycles:
 - The Admin Request Lifecycle
 - The Front-end Request Lifecycle
- During the flow of these lifecycles, other contextual events are triggered, and are dependent upon the requested action that precedes it.

Admin Request Lifecycle

Admin Request Lifecycle

- The administration area of Mura has a multitude of contextual events that could be triggered, depending on the requested action.

Admin Request Lifecycle

- The administration area of Mura has a multitude of contextual events that could be triggered, depending on the requested action.
- For example, a user could be attempting to create new content, updating a user, or deleting a category.

Admin Request Lifecycle

- The administration area of Mura has a multitude of contextual events that could be triggered, depending on the requested action.
- For example, a user could be attempting to create new content, updating a user, or deleting a category.
- Each of these actions could also trigger their own contextual events.

```
1.  onGlobalRequestStart
2.    onAdminRequestStart
3.
4.    onAdminHTMLHeadRender          // renders just before the closing </head> tag
5.
6.    onAdminMFAChallengeRender    // Multi-factor Auth area of login screen
7.
8.    onDashboardReplacement       // render a completely new dashboard
9.
10.   onDashboardPrimaryTop        // renders in the top of the dashboard
11.   onDashboardPrimaryBottom     // renders in the bottom of the dashboard
12.   onDashboardSidebarTop        // renders in the top of the dashboard sidebar
13.   onDashboardSidebarBottom      // renders in the bottom of the dashboard sidebar
14.
15.   onAdminNavMainRender         // render additional main nav menu items
16.   onFToolbarExtensionRender   // render additional front-end toolbar menu items
17.
18.   on{Type}SecondaryNavRender
19.   on{Type}{Subtype}SecondaryNavRender
20.
21.   on{Type}{Subtype}NewContentMenuRender
22.   onNewContentMenuRender
23.
24.   onAdminHTMLFootRender        // renders just before the closing </body> tag
25.
26.   onAdminRequestEnd
27.   onGlobalRequestEnd
```

Front-End Request Lifecycle

Front-End Request Lifecycle

- While there may be any number of contextual events that are triggered, dependent upon the requested action, a typical front-end request does consist of several events triggered in a specific order.

Front-End Request Lifecycle

- When it comes to the front-end request lifecycle, if you're most interested in the final string of code that will be returned to the browser, you'll want to be aware of

m.event('__MuraResponse__').

Front-End Request Lifecycle

- When it comes to the front-end request lifecycle, if you're most interested in the final string of code that will be returned to the browser, you'll want to be aware of

m.event('__MuraResponse__').

- *That's a double-underscore in front of, and after “MuraResponse”.*

Front-End Request Lifecycle

- Visit <http://docs.getmura.com/v7-1/mura-developers/mura-events/event-lifecycles/front-end-request-lifecycle/> to view a typical front-end request lifecycle.

Event Handlers

Event Handlers

- Mura event handlers are simply ColdFusion components (CFCs), or files save with the extension “**.cfc**”.

Event Handlers

- Mura event handlers are simply ColdFusion components (CFCs), or files save with the extension “**.cfc**”.
- They contain methods/functions, and may contain other variables and/or data.

Event Handlers

- Mura event handlers are simply ColdFusion components (CFCs), or files save with the extension “**.cfc**”.
- They contain methods/functions, and may contain other variables and/or data.
- Most contain one or more of Mura’s event hooks, and/or custom event hooks.

Event Handlers

- The event hooks are merely “announced” during specific points in the request.

Event Handlers

- The event hooks are merely “announced” during specific points in the request.
- Registered event handlers will be parsed when the event they’re registered for occurs, and if any event hooks are found, they will be executed.

Event Handlers

- The event hooks are merely “announced” during specific points in the request.
- Registered event handlers will be parsed when the event they’re registered for occurs, and if any event hooks are found, they will be executed.
- For example, if you have two registered handlers, and both contain a method listening for the same hook, both will execute, when event is announced.

Event Handlers

- Mura's standard events handler is located under {context}/core/mura/Handler/standardEventsHandler.cfc.

Event Handlers

- Mura's standard events handler is located under {context}/core/mura/Handler/standardEventsHandler.cfc.
- You may also view them via Mura's Component API at <http://www.getmura.com/component-api/7.1/index.html?mura/Handler/standardEventsHandler.html>

Event Handlers

- You may also have several Mura event handlers throughout your sites, themes, content types, display objects/modules, and plugins.

Event Handlers

- You may also have several Mura event handlers throughout your sites, themes, content types, display objects/modules, and plugins.
- In addition, you can choose to register your event handlers by convention, or explicitly register them via a known event handler.

Event Handler

- With the exception of the “Site” event handler, or the “Theme” event handler, you can name your event handler anything you want, as long as it has the “**.cfc**” file extension.

Event Handler

- With the exception of the “Site” event handler, or the “Theme” event handler, you can name your event handler anything you want, as long as it has the “**.cfc**” file extension.
- All event handlers should extend “**mura.cfobject**”.

Event Handler

- With the exception of the “Site” event handler, or the “Theme” event handler, you can name your event handler anything you want, as long as it has the “**.cfc**” file extension.
- All event handlers should extend “**mura.cfobject**”.
- By doing so, you allow yourself the ability to leverage Mura’s baked-in methods for its objects.

CFScript-based .CFC Example

```
1. component extends="mura.cfobject" {  
2.  
3.     public any function onSomeEvent(m) {  
4.         // Do something  
5.     }  
6.  
7. }
```

Tag-based .CFC Example

```
1. <cfcomponent extends="mura.cfobject" output="false">  
2.  
3.   <cffunction name="onSomeEvent" access="public" returntype="any">  
4.     <cfargument name="m" hint="Mura Scope">  
5.     <!-- Do something -->  
6.   </cffunction>  
7.  
8. </cfcomponent>
```

Plugin Event Handlers

Plugin Event Handlers

- Plugin event handlers should extend “**mura.plugin.pluginGenericEventHandler**”, instead of using the typical “**mura.cfobject**”.

Plugin Event Handlers

- Plugin event handlers should extend “**mura.plugin.pluginGenericEventHandler**”, instead of using the typical “**mura.cfobject**”.
- Doing so allows you access to the plugin’s own configuration information via “**variables.pluginConfig**”, as well as Mura’s configuration data via “**variables.configBean**”.

Plugin Event Handlers

- Plugin event handlers should extend “**mura.plugin.pluginGenericEventHandler**”, instead of using the typical “**mura.cfobject**”.
- Doing so allows you access to the plugin’s own configuration information via “**variables.pluginConfig**”, as well as Mura’s configuration data via “**variables.configBean**”.
- We’ll cover registration in the Plugins section.

Register By Convention

Register By Convention

- One way to register a Mura event handler is by convention.

Register By Convention

- One way to register a Mura event handler is by convention.
- This means, if you create a “**.cfc**” file, and place it in a “known” location, Mura will automatically register it, and any event hooks contained within it will be executed when announced.

Register By Convention

- The “Site” Event Handler

```
1. {context}/sites/{SiteID}/eventHandler.cfc
```

Register By Convention

- The “Site” Event Handler
 - Instantiated in every *front-end* request.

```
1. {context}/sites/{SiteID}/eventHandler.cfc
```

Register By Convention

- The “Site” Event Handler
 - Instantiated in every *front-end* request.
 - Methods/event hooks should be specific to front-end requests only.

```
1. {context}/sites/{SiteID}/eventHandler.cfc
```

Register By Convention

- The “Theme” Event Handler

```
1. {context}/sites/{SiteID}/themes/{ThemeName}/eventHandler.cfc  
2. // OR  
3. {context}/themes/{ThemeName}/eventHandler.cfc
```

Register By Convention

- The “Theme” Event Handler
 - Instantiated during **onApplicationLoad()**

```
1. {context}/sites/{SiteID}/themes/{ThemeName}/eventHandler.cfc  
2. // OR  
3. {context}/themes/{ThemeName}/eventHandler.cfc
```

Register By Convention

- The “Theme” Event Handler
 - Instantiated during **onApplicationLoad()**
 - Placed into the application scope

```
1. {context}/sites/{SiteID}/themes/{ThemeName}/eventHandler.cfc  
2. // OR  
3. {context}/themes/{ThemeName}/eventHandler.cfc
```

Register By Convention

- The “Theme” Event Handler
 - Instantiated during **onApplicationLoad()**
 - Placed into the **application** scope
 - You must reload Mura anytime you make a change

```
1. {context}/sites/{SiteID}/themes/{ThemeName}/eventHandler.cfc  
2. // OR  
3. {context}/themes/{ThemeName}/eventHandler.cfc
```

Register By Convention

- Content Type & Module Event Handlers

```
1.  ../model/handlers/{anyFilename}.cfc
```

Register By Convention

- Content Type & Module Event Handlers
- Currently, “content_types” and “module” directories may reside under a site and/or theme.

```
1. // Sites  
2. {context}/sites/{SiteID}/content_types/{Type}_{Subtype}/model/handlers/  
3. {context}/sites/{SiteID}/modules/{module}/model/handlers/  
4.  
5. // Themes  
6. ../themes/{ThemeName}/content_types/{Type}_{Subtype}/model/handlers/  
7. ../themes/{ThemeName}/modules/{module}/model/handlers/
```

Register By Convention

- Content Type & Module Event Handlers
- Currently, “content_types” and “module” directories may reside under a site and/or theme.
- And global “modules” too!
- {context}/modules/{module}/model/handlers/

```
1. // Sites
2. {context}/sites/{SiteID}/content_types/{Type}_{Subtype}/model/handlers/
3. {context}/sites/{SiteID}/modules/{module}/model/handlers/
4.
5. // Themes
6. ../themes/{ThemeName}/content_types/{Type}_{Subtype}/model/handlers/
7. ../themes/{ThemeName}/modules/{module}/model/handlers/
```

Explicitly Registered Handlers

Explicitly Registered Handlers

- Basic syntax

```
1. m.getBean('pluginManager').addEventHandler(  
2.   {handlerObject or path}  
3.   , {SiteID}  
4. );
```

Explicitly Registered Handlers

- Example

```
1. public any function onApplicationLoad(event, m) {
2.     // The customHandler.cfc file exists in the same directory as this file
3.     var myHandler = new customHandler();
4.
5.     // Register the custom handler
6.     arguments.m.getBean('pluginManager').addEventHandler(
7.         myHandler
8.         , arguments.event.get('siteid')
9.     );
10. }
```

Entity Event Handlers

Entity Event Handlers

- Introduced in Mura v7.1, developers may register handlers for specific beans/objects or entities such as content beans, user beans, or even your own custom entities/objects.

Entity Event Handlers

- Introduced in Mura v7.1, developers may register handlers for specific beans/objects or entities such as content beans, user beans, or even your own custom entities/objects.
- This allows developers the ability to create more focused business logic and code clarity.

Entity Event Handlers

- For example, if you registered an event handler for “onRenderStart”, your method will be invoked for each and every front-end page request, regardless of the content type, etc.

Entity Event Handlers

- For example, if you registered an event handler for “onRenderStart”, your method will be invoked for each and every front-end page request, regardless of the content type, etc.
- Using targeted logic allows you to specify not only the object and type, you could also target a very specific object by loading it up, and attaching a custom event handler to it.

Entity Event Handlers

- To target a specific bean/object, we begin by using the “onApplicationLoad” event of a registered event handler.

```
1. public any function onApplicationLoad(m) {  
2.     // This is where our code will be  
3. }
```

Entity Event Handlers

- To target a specific bean/object, we begin by using the “onApplicationLoad” event of a registered event handler.
- Then, we’ll target a specific bean/object, and use either the “on” or “addEventHandler” methods.

```
1. public any function onApplicationLoad(m) {  
2.     // This is where our code will be  
3. }
```

Entity Event Handlers

- `on`
 - A bean/object helper method to dynamically register an event handler.
Methods may be chained to attach multiple event handlers.

Entity Event Handlers

- on
 - A bean/object helper method to dynamically register an event handler.
Methods may be chained to attach multiple event handlers.

```
1. m.getBean('someBean').on( eventName, fn )
```

Entity Event Handlers

- **on**
 - A bean/object helper method to dynamically register an event handler.
Methods may be chained to attach multiple event handlers.

```
1. m.getBean('someBean').on( eventName, fn )
```

| Parameter | Type | Req/Opt | Description |
|-----------|----------|---------|---|
| eventName | string | Req | The name of the event you wish to register, without the " <code>on</code> " prefix. For example, if you wish to target the " <code>onRenderStart</code> " method, set this to " <code>renderStart</code> ". |
| fn | function | Req | A function, or name of a function containing the code you wish to execute. |

Entity Event Handlers

```
1. public any function onApplicationLoad(m) {  
2.     m.getBean('content')  
3.         .loadBy(title='Contact Us')  
4.         .on('renderStart', function(m) {  
5.             // do something  
6.         });  
7.     }  
8.  
9. }
```

Entity Event Handlers

```
1. public any function onApplicationLoad(m) {  
2.     //  
3.     m.getBean('user')  
4.         .loadByUsername='steve')  
5.         .on('userDelete', function(m) {  
6.             // do something  
7.         });  
8.     //  
9. }
```

Entity Event Handlers

```
1.  public any function onApplicationLoad(m) {  
2.    ...  
3.    m.getBean('configBean')  
4.      .on('onRenderEnd', function(m) {  
5.        // do something  
6.      });  
7.  
8.    // OR  
9.  
10.   m.globalConfig()  
11.     .on('onRenderEnd', function(m) {  
12.       // do something  
13.     });  
14.  
15. }
```

Entity Event Handlers

```
1. public any function onApplicationLoad(m) {  
2.     m.getBean('content')  
3.         .loadBy(title='Information Request Form')  
4.         .on('submitSave', function(m) {  
5.             // do something  
6.         })  
7.         .on('submitResponseRender', function(m) {  
8.             // do something  
9.         });  
10.    );  
11.  
12. }
```

Entity Event Handlers

```
1. public any function onApplicationLoad(m) {  
2.     //  
3.     m.getBean('configBean')  
4.         .on('beforeWidgetSave', function(m) {  
5.             // do something  
6.         })  
7.         .on('widgetSave', function(m) {  
8.             // do something  
9.         })  
10.        .on('widgetSave', function(m) {  
11.            // do something  
12.        });  
13.  
14. }
```

Entity Event Handlers

- `addEventHandler`
 - A bean/object helper method to dynamically register event handlers.

Entity Event Handlers

- `addEventHandler`
 - A bean/object helper method to dynamically register event handlers.

```
1. m.getBean('someBean').addEventHandler( component )
```

Entity Event Handlers

- `addEventHandler`
 - A bean/object helper method to dynamically register event handlers.

```
1. m.getBean('someBean').addEventHandler( component )
```

| Parameter | Type | Req/Opt | Description |
|-----------|--------|---------|--|
| component | struct | Req | A struct of key/value pairs of eventNames and functions to execute as event handlers for the specified eventName(s). |

Entity Event Handlers

```
1. public any function onApplicationLoad(m) {  
2.  
3.     m.getBean('configBean')  
4.         .addEventHandler{  
5.             beforeWidgetSave = function(m) {  
6.                 // do something  
7.             }  
8.             , widgetSave = function(m) {  
9.                 // do something  
10.            }  
11.            , afterWidgetSave = function(m) {  
12.                // do something  
13.            }  
14.        });  
15.  
16.    }
```

Entity Event Handlers

```
1.  public any function onApplicationLoad(m) {  
2.    ...  
3.    m.getBean('widget')  
4.      .loadBy(id='someID')  
5.      .addEventHandler({  
6.        beforeSave = function(m) {  
7.          // do something  
8.        }  
9.        , save = function(m) {  
10.          // do something  
11.        }  
12.        , afterSave = function(m) {  
13.          // do something  
14.        }  
15.      });  
16.    ...  
17.  }
```

Mura's Event Log

Mura's Event Log

- Mura has the capability to output an event log as a stack trace, including the time it took to execute, in milliseconds, directly to the browser.

Mura's Event Log

- Mura has the capability to output an event log as a stack trace, including the time it took to execute, in milliseconds, directly to the browser.
- This feature is quite useful for troubleshooting and debugging, as well as discovering which areas of your code may benefit from applying performance optimization techniques.

Enable/Disable Stack Trace

Enable/Disable Stack Trace

- To enable the stack trace, you must first be logged in to Mura as an Administrator.

Enable/Disable Stack Trace

- To enable the stack trace, you must first be logged in to Mura as an Administrator.
- Then, simply append “**/?showtrace=1**” to the URL, and reload your browser.

Enable/Disable Stack Trace

- To enable the stack trace, you must first be logged in to Mura as an Administrator.
- Then, simply append “**/?showtrace=1**” to the URL, and reload your browser.
- Once you do this, Mura will append a stack trace output to the end of your page.

Enable/Disable Stack Trace

Stack Trace

1. Reading config/settings.ini.cfm (1 | 1)
2. mura.Handler.standardEventsHandler.standardSetContentRendererHandler (1 | 12)
3. mura.Handler.standardEventsHandler.standardSetContentHandler (12 | 30)
4. mura.Handler.standardEventsHandler.standardSetAdTrackingHandler (0 | 18)
5. mura.Handler.standardEventsHandler.standardSetIsOnDisplayHandler (1 | 31)
6. mura.Handler.standardEventsHandler.standardDoActionsHandler (0 | 31)
7. mura.Handler.standardEventsHandler.standardSetPermissionsHandler (0 | 31)
8. mura.Handler.standardEventsHandler.standardSetLocaleHandler (0 | 31)
9. mura.Handler.standardEventsHandler.standardSetCommentPermissionsHandler (0 | 31)
10. mura.Handler.standardEventsHandler.standardDoResponseHandler (808 | 839)
11. muraWRM.sites.default.includes.themes.MuraBootstrap3.eventHandler.onRenderStart (0 | 32)
12. mura.Handler.standardEventsHandler.standardTranslationHandler (807 | 839)
13. mura.Translator.standardHTMLTranslator.translate (806 | 839)
14. /muraWRM/sites/default/includes/themes/MuraBootstrap3/templates/three_column.cfm (766 | 803)
15. contentRenderer.dspPrimaryNav (23 | 65)
16. Loading contentBean (0 | 70)
17. Loading contentBean (0 | 75)
18. /muraWRM/sites/default/includes/display_objects/nav/dsp_standard.cfm (4 | 87)
19. contentRenderer.dspStandardNav (4 | 87)
20. contentRenderer.dspSubNav (4 | 87)
21. /muraWRM/admin/core/views/carch/objectclass/object/meta.cfm (0 | 89)
22. /muraWRM/sites/default/includes/display_objects/collection/index.cfm (699 | 800)
23. /muraWRM/sites/default/includes/display_objects/feed/index.cfm (699 | 800)
24. Loading feedBean (0 | 102)
25. /muraWRM/admin/core/views/carch/objectclass/object/meta.cfm (0 | 802)
26. /muraWRM/sites/default/includes/themes/MuraBootstrap3/display_objects/examples/sampleModalWindow.cfm (0 | 803)
27. /muraWRM/sites/default/includes/display_objects/htmlhead/global.cfm (2 | 807)
28. /muraWRM/sites/default/includes/display_objects/htmlhead/prettify.cfm (0 | 807)
29. /muraWRM/sites/default/includes/display_objects/component/index.cfm (6 | 838)
30. The requested file 'extensions/dsp_Component_Default.cfm' could not be found. (0 | 838)
31. /muraWRM/admin/core/views/carch/objectclass/object/meta.cfm (0 | 839)

Total: 840 milliseconds

Enable/Disable Stack Trace

- Mura will then set a cookie, so you don't have to continually add it to each and every page you visit.

Enable/Disable Stack Trace

- Mura will then set a cookie, so you don't have to continually add it to each and every page you visit.
- If you wish to disable this feature, simply append “`/?showtrace=0`” to the URL, and the stack trace should disappear after reloading your browser.

Read/Use the Stack Trace

Read/Use the Stack Trace

- The stack trace displays information useful for both debugging or troubleshooting, as well as identifying areas of code execution that might benefit from using some performance optimization techniques, such as caching, or possibly even rewriting the code to run more efficiently.

Read/Use the Stack Trace

- Each item in the stack trace is listed in the order in which it was encountered during the execution of the request.

Read/Use the Stack Trace

- Each item in the stack trace is listed in the order in which it was encountered during the execution of the request.
- Some items in the list simply display informational messages, and don't necessarily point to any specific line of code, while others output the entire path to the file or method being parsed.

Read/Use the Stack Trace

- In the parenthesis to the right of each item in the list are two numbers.

Read/Use the Stack Trace

- In the parenthesis to the right of each item in the list are two numbers.
- The number on the left indicates the number of milliseconds it took the server to parse the specific file or method.

Read/Use the Stack Trace

- In the parenthesis to the right of each item in the list are two numbers.
- The number on the left indicates the number of milliseconds it took the server to parse the specific file or method.
- The number on the right reflects the point in the total request (in milliseconds) when the file or method finished executing.

Stack Trace

1. Reading config/settings.ini.cfm **(1 | 1)**
2. mura.Handler.standardEventsHandler.standardSetContentRendererHandler **(1 | 12)**
3. mura.Handler.standardEventsHandler.standardSetContentHandler **(12 | 30)**
4. mura.Handler.standardEventsHandler.standardSetAdTrackingHandler **(0 | 18)**
5. mura.Handler.standardEventsHandler.standardSetIsOnDisplayHandler **(1 | 31)**
6. mura.Handler.standardEventsHandler.standardDoActionsHandler **(0 | 31)**
7. mura.Handler.standardEventsHandler.standardSetPermissionsHandler **(0 | 31)**
8. mura.Handler.standardEventsHandler.standardSetLocaleHandler **(0 | 31)**
9. mura.Handler.standardEventsHandler.standardSetCommentPermissionsHandler **(0 | 31)**
10. mura.Handler.standardEventsHandler.standardDoResponseHandler **(808 | 839)**
11. muraWRM.sites.default.includes.themes.MuraBootstrap3.eventHandler.onRenderStart **(0 | 32)**
12. mura.Handler.standardEventsHandler.standardTranslationHandler **(807 | 839)**
13. mura.Translator.standardHTMLTranslator.translate **(806 | 839)**
14. **/muraWRM/sites/default/includes/themes/MuraBootstrap3/templates/three_column.cfm (766 | 803)**
15. contentRenderer.dspPrimaryNav **(23 | 65)**
16. Loading contentBean **(0 | 70)**
17. Loading contentBean **(0 | 75)**
18. /muraWRM/sites/default/includes/display_objects/nav/dsp_standard.cfm **(4 | 87)**
19. contentRenderer.dspStandardNav **(4 | 87)**
20. contentRenderer.dspSubNav **(4 | 87)**
21. /muraWRM//admin/core/views/carch/objectclass/object/meta.cfm **(0 | 89)**
22. /muraWRM/sites/default/includes/display_objects/collection/index.cfm **(699 | 800)**
23. /muraWRM/sites/default/includes/display_objects/feed/index.cfm **(699 | 800)**
24. Loading feedBean **(0 | 102)**
25. /muraWRM//admin/core/views/carch/objectclass/object/meta.cfm **(0 | 802)**
26. /muraWRM/sites/default/includes/themes/MuraBootstrap3/display_objects/examples/sampleModalWindow.cfm **(0 | 803)**
27. /muraWRM/sites/default/includes/display_objects/htmlhead/global.cfm **(2 | 807)**
28. /muraWRM/sites/default/includes/display_objects/htmlhead/prettyify.cfm **(0 | 807)**
29. /muraWRM/sites/default/includes/display_objects/component/index.cfm **(6 | 838)**
30. The requested file 'extensions/dsp_Component_Default.cfm' could not be found. **(0 | 838)**
31. /muraWRM//admin/core/views/carch/objectclass/object/meta.cfm **(0 | 839)**

Total: 840 milliseconds

Read/Use the Stack Trace

- While you may not necessarily have the ability to optimize some of Mura's core methods, you can see the “**three_column.cfm**” layout template is code you have direct control over.

Read/Use the Stack Trace

- While you may not necessarily have the ability to optimize some of Mura's core methods, you can see the “**three_column.cfm**” layout template is code you have direct control over.
- Maybe some of the code within the layout template could benefit from using a caching strategy, or some other form of performance optimization.

Custom Stack Trace Points

Custom Stack Trace Points

- As a Mura developer, you may add your own custom messages to the stack trace output.

Custom Stack Trace Points

- As a Mura developer, you may add your own custom messages to the stack trace output.
- This is useful for inspecting how long it takes for your custom code to execute, quickly and easily.

Custom Stack Trace Points

```
1. <!-- Place this either at the top of your file, or at the beginning of a block of code  
you wish to trace -->  
2. <cfset myTracePoint = $.initTracePoint('your filename, method name, or other description  
to identify this trace point goes here ... this is what will output in the Stack Trace') />  
3.  
4. <!-- file content or block of code goes here -->  
5.  
6. <!-- Place this either at the bottom of your file, or at the end of a block of code yo  
u wish to trace -->  
7. <cfset m.commitTracePoint(myTracePoint) />
```

Stack Trace

1. Reading config/settings.ini.cfm **(0 | 1)**
2. mura.Handler.standardEventsHandler.standardSetContentRendererHandler **(3 | 14)**
3. mura.Handler.standardEventsHandler.standardSetContentHandler **(9 | 24)**
4. mura.Handler.standardEventsHandler.standardSetAdTrackingHandler **(0 | 15)**
5. mura.Handler.standardEventsHandler.standardSetIsOnDisplayHandler **(1 | 25)**
6. mura.Handler.standardEventsHandler.standardDoActionsHandler **(0 | 25)**
7. mura.Handler.standardEventsHandler.standardSetPermissionsHandler **(0 | 25)**
8. mura.Handler.standardEventsHandler.standardSetLocaleHandler **(0 | 25)**
9. mura.Handler.standardEventsHandler.standardSetCommentPermissionsHandler **(0 | 25)**
10. mura.Handler.standardEventsHandler.standardDoResponseHandler **(818 | 843)**
11. muraWRM.sites.default.includes.themes.MuraBootstrap3.eventHandler.onRenderStart **(0 | 26)**
12. mura.Handler.standardEventsHandler.standardTranslationHandler **(817 | 843)**
13. mura.Translator.standardHTMLTranslator.translate **(816 | 843)**
14. /muraWRM/sites/default/includes/themes/MuraBootstrap3/templates/three_column.cfm **(785 | 816)**
15. contentRenderer.dspPrimaryNav **(17 | 54)**
16. Loading contentBean **(0 | 58)**
17. Loading contentBean **(0 | 62)**
18. /muraWRM/sites/default/includes/display_objects/nav/dsp_standard.cfm **(3 | 73)**
19. contentRenderer.dspStandardNav **(3 | 73)**
20. contentRenderer.dspSubNav **(3 | 73)**
21. /muraWRM//admin/core/views/carch/objectclass/object/meta.cfm **(0 | 74)**
22. /muraWRM/sites/default/includes/display_objects/collection/index.cfm **(732 | 814)**
23. /muraWRM/sites/default/includes/display_objects/feed/index.cfm **(732 | 814)**
24. Loading feedBean **(0 | 83)**
25. /muraWRM//admin/core/views/carch/objectclass/object/meta.cfm **(0 | 816)**
26. your filename, method name, or other description to identify this trace point goes here ... this is what will output in the Stack Trace **(0 | 816)**
27. /muraWRM/sites/default/includes/themes/MuraBootstrap3/display_objects/examples/sampleModalWindow.cfm **(0 | 816)**
28. /muraWRM/sites/default/includes/display_objects/htmlhead/global.cfm **(2 | 820)**
29. /muraWRM/sites/default/includes/display_objects/htmlhead/prettyify.cfm **(0 | 820)**
30. /muraWRM/sites/default/includes/display_objects/component/index.cfm **(5 | 842)**
31. The requested file 'extensions/dsp_Component_Default.cfm' could not be found. **(0 | 842)**
32. /muraWRM//admin/core/views/carch/objectclass/object/meta.cfm **(0 | 843)**

Lab Exercise

LET'S ADD AND REPLACE SOME BUSINESS LOGIC

Lab Exercise

- See `/3-core-developer/3-mura-events/instructions.md`

Summary

- Lifecycle Events vs. Contextual Events
- Mura “event” Scope
- Event Hooks
- Event Lifecycles
- Event Handlers
- Mura’s Event Log
- Lab Exercise

Mura Beans & Objects

REUSABLE SOFTWARE COMPONENTS

Mura Beans & Objects

- Introduction
- Common Bean Objects
- Mura Scope Objects
- Custom Objects
- Mura Iterators
- Lab Exercise

Mura Beans & Objects

Mura Beans & Objects

- The term “bean” is borrowed from the Java programming language. According to Wikipedia, JavaBeans are reusable software components.

Mura Beans & Objects

- The term “bean” is borrowed from the Java programming language. According to Wikipedia, JavaBeans are reusable software components.
- They are used to encapsulate many objects into a single object (the bean), so they can be passed around as a single bean object instead of multiple individual objects.

Mura Beans & Objects

- In CFML, a bean can be a ColdFusion Component (CFC). However there are a few characteristics the CFC must have to differentiate itself from other types of CFCs

Mura Beans & Objects

- In CFML, a bean can be a ColdFusion Component (CFC). However there are a few characteristics the CFC must have to differentiate itself from other types of CFCs
 - Serializable
 - A no-argument constructor
 - Allows access to properties using getter and setter methods

Mura Beans & Objects

- In CFML, a bean can be a ColdFusion Component (CFC). However there are a few characteristics the CFC must have to differentiate itself from other types of CFCs
 - Serializable
 - A no-argument constructor
 - Allows access to properties using getter and setter methods
- In short, beans hold information about the object or entity they represent, and allow developers the ability to quickly and easily access and manipulate some of its attributes

Mura Bean Objects

Mura Bean Objects

- In Mura, developers can use its software components without necessarily having to understand their inner workings.

Mura Bean Objects

- In Mura, developers can use its software components without necessarily having to understand their inner workings.
- Mura bean objects provide developers the ability to directly access and manipulate their associated records in the database, typically via the bean's associated helper methods.

Mura Bean Objects

- In Mura, developers can use its software components without necessarily having to understand their inner workings.
- Mura bean objects provide developers the ability to directly access and manipulate their associated records in the database, typically via the bean's associated helper methods.
- As you'll see in the Custom Objects section, developers may also create their own bean objects too.

Mura Bean Objects

- When you want to work a specific bean object, you will first want to load it by a specific attribute.

Mura Bean Objects

- When you want to work a specific bean object, you will first want to load it by a specific attribute.
- However, as each bean object is slightly different, the attributes you may load by will vary.

Loading Bean Objects

```
1. | someBean = m.getBean('someBean')  
2. |       .loadBy(someAttribute='someAttributeValue');
```

Loading Bean Objects

```
1. someBean = m.getBean('someBean')  
2.           .loadBy(someAttribute='someAttributeValue', siteid='SomeSiteID');
```

Getting Attributes

1. `someBean.get('attributeName');`
2. `someBean.get{AttributeName}();`
3. `someBean.getValue('attributeName');`

Setting Attributes

1. `someBean.set('attributeName', 'Some Value');`
2. `someBean.set{AttributeName}('Some Value');`
3. `someBean.setValue('attributeName', 'Some Value');`

Verifying Existence

```
1. // returns `true` if bean exists, or `false` if it does not  
2. someBean.exists();  
3.  
4. // returns `true` if bean is new, or `false` if bean already exists  
5. someBean.getIsNew();
```

Deleting Bean Objects

```
1. | someBean.delete();
```

Creating, Updating, & Saving

1.

```
someBean.save();
```

Error Handling

```
1. someBean.save();
2.
3. // If the bean has errors, then it did not save...
4. if ( someBean.hasErrors() ) {
5.
6.     // errors are returned as a struct
7.     errors = someBean.getErrors();
8.
9.     WriteOutput('<h3>Please review the following errors:</h3><ul>');
10.
11.    // Loop over the errors
12.    for ( e in errors ) {
13.        WriteOutput('<li>Attribute: #e# <br> Message: #errors[e]#</li>');
14.    }
15.
16.    WriteOutput('</ul>');
17. } else {
18.     WriteOutput('<h2>Success :: No Errors!</h2>');
19. }
```

Common Bean Objects

Common Bean Objects

- configBean
- site
- content
- user
- group
- comment
- category
- feed

Config Bean

Config Bean

- The **configBean** object includes instance-wide configuration data, primarily the data collected in the **settings.ini.cfm** file.

Loading the Bean

1.

```
<cfset configBean = m.getBean('configBean')>
```

Getters

1. configBean.get('attributeName')
2. configBean.get{AttributeName}()
3. configBean.getValue('attributeName')

Setters

1. configBean.set('attributeName', 'someValue')
2. configBean.set{AttributeName}('someValue')
3. configBean.setValue('attributeName', 'someValue')

Setters

- Setting any values of the **configBean** are only temporary, and last for the duration of the request. In other words, calling the **save()** method will not save any values.

```
1. configBean.set('attributeName', 'someValue')
2. configBean.set{AttributeName}('someValue')
3. configBean.setValue('attributeName', 'someValue')
```

Attributes

- For a listing of attributes, please visit <http://docs.getmura.com/v7-1/mura-developers/mura-beans-objects/common-bean-objects/config-bean/>

Site Bean

Site Bean

- The **site** bean/object includes site configuration data collected via the Site Settings section in the administration area, as well as other site-specific information.

Loading the Bean

```
1. | <cfset siteBean = m.getBean('site').loadBy(siteid=session.siteid)>
```

Getters

1. `siteBean.get('attributeName')`
2. `siteBean.get{AttributeName}()`
3. `siteBean.getValue('attributeName')`

Setters

1. siteBean.set('attributeName', 'someValue')
2. siteBean.set{AttributeName}('someValue')
3. siteBean.setValue('attributeName', 'someValue')

Attributes

- For a listing of attributes, please visit <http://docs.getmura.com/v7-1/mura-developers/mura-beans-objects/common-bean-objects/site-bean/>

Content Bean

Content Bean

- The **content** bean/object includes data about a Mura content item, such as the title, summary, body, etc. In addition, there are several content-specific helper methods included.

Content Bean

- The **content** bean/object includes data about a Mura content item, such as the title, summary, body, etc. In addition, there are several content-specific helper methods included.
- For example, you could get an iterator of any children of the content item, what specific categories the content item has been assigned to, etc.

Loading the Bean

- You can load a content bean/object by the following attributes:

`contentid, contenthistid, filename, remoteid, title,
urltitle`

```
1. <cfset contentBean = m.getBean('content').loadBy(title='Home')>
```

Getters

- The **content** object includes information about the content item itself.

```
1. contentBean.getAttributeName()  
2. contentBean.getAttributeName()  
3. contentBean.getValue('attributeName')
```

Setters

1. `contentBean.set('attributeName', 'someValue')`
2. `contentBean.set{AttributeName}('someValue')`
3. `contentBean.setValue('attributeName', 'someValue')`

Attributes

- For a listing of attributes, please visit <http://docs.getmura.com/v7-1/mura-developers/mura-beans-objects/common-bean-objects/content-bean/>

Helper Methods

```
1. <cfset it = contentBean.getCategoriesIterator()>
2. <!-- Loop over the iterator --->
3. <cfif it.hasNext()>
4.   <ul>
5.     <cfloop condition="it.hasNext()">
6.       <cfset item = it.next()>
7.       <li>
8.         #esapiEncode('html', item.get('name'))#
9.       </li>
10.      </cfloop>
11.    </ul>
12.  <cfelse>
13.    <!-- Content item has not been categorized --->
14.  </cfif>
```

Helper Methods

```
1. <cfset it = contentBean.getKidsCategoryIterator()>
2. <!-- Loop over the iterator -->
3. <cfif it.hasNext()>
4.   <ul>
5.     <cfloop condition="it.hasNext()">
6.       <cfset item = it.next()>
7.       <li>
8.         #esapiEncode('html', item.get('name'))#
9.       </li>
10.      </cfloop>
11.    </ul>
12.  <cfelse>
13.    <!-- No child categories exist -->
14.  </cfif>
```

Helper Methods

```
1. <cfset it = contentBean.getRelatedContentIterator()>
2. <!-- Loop over the iterator --->
3. <cfif it.hasNext()>
4.   <ul>
5.     <cfloop condition="it.hasNext()">
6.       <cfset item = it.next()>
7.       <li>
8.         <a href="#item.getUrl()#">
9.           #esapiEncode('html', item.getTitle())#
10.        </a>
11.      </li>
12.    </cfloop>
13.  </ul>
14. <cfelse>
15.   <!-- No related content items exist --->
16. </cfif>
```

Helper Methods

```
1. | <cfset it = contentBean.getRelatedContentIterator(name='Related Videos')>
   |
2. | <!-- Loop over the iterator -->
3. | <cfif it.hasNext()>
4. |   <ul>
5. |     <cfloop condition="it.hasNext()">
6. |       <cfset item = it.next()>
7. |       <li>
8. |         <a href="#item.getUrl()#">
9. |           #esapiEncode('html', item.getTitle())#
10. |         </a>
11. |       </li>
12. |     </cfloop>
13. |   </ul>
14. | <cfelse>
15. |   <!-- No related content items exist -->
16. | </cfif>
```

Helper Methods

```
1. <cfset it = contentBean.getCommentsIterator()>
2. <!-- Loop over the iterator --->
3. <cfif it.hasNext()>
4.   <ul>
5.     <cfloop condition="it.hasNext()">
6.       <cfset item = it.next()>
7.       <li>
8.         <div class="comments-datetime">
9.           #LSDateFormat(item.get('entered'))#
10.          #LSTimeFormat(item.get('entered'))#
11.        </div>
12.        <div class="comments-text">
13.          #esapiEncode('html', item.get('comments'))#
14.        </div>
15.        <div class="comments-author">
16.          #esapiEncode('html', item.get('name'))#
17.        </div>
18.      </li>
19.    </cfloop>
20.  </ul>
21. <cfelse>
22.   <!-- No comments exist --->
23. </cfif>
```

User Bean

User Bean

- The **user** bean/object includes user-specific data, and includes a number of helper methods.

User Bean

- The **user** bean/object includes user-specific data, and includes a number of helper methods.
- For example, you can obtain an iterator of the groups the user belongs to, etc.

User Bean

- **Note:** The **group** bean/object is essentially a **user** bean/object. The primary difference is that the **user** bean/object contains only user-specific information, and user-specific helper methods. Also, the “**type**” attribute will be “1” if a group, and “2” if a user.

Loading the Bean

- You can load a user bean/object by the following attributes:
userid, username, remoteid

```
1. <cfset userBean = m.getBean('user').loadByUsername='steve'>
```

Loading the Bean

- You can load a content bean/object by the following attributes:
userid, username, remoteid

```
1. | <cfset userBean = m.getBean('user').loadBy(username='steve')>
```

- An optional second argument, labeled “siteid” is necessary when working with a user assigned to a different site than the one you are current working with.

```
1. | <cfset userBean = m.getBean('user').loadBy(username='steve', siteid='default')>
```

Getters

1. `userBean.getAttributeName()`
2. `userBean.get{AttributeName}()`
3. `userBean.getValue('attributeName')`

Setters

1. `userBean.set('attributeName', 'someValue')`
2. `userBean.set{AttributeName}('someValue')`
3. `userBean.setValue('attributeName', 'someValue')`

Attributes

- For a listing of attributes, please visit <http://docs.getmura.com/v7-1/mura-developers/mura-beans-objects/common-bean-objects/user-bean/>

Helper Methods

```
1. <cfif userBean.isInGroup('admin')>
2.   <!-- User is a member of the 'admin' group -->
3. <cfelse>
4.   <!-- User is NOT a member of the 'admin' group -->
5. </cfif>
```

Helper Methods

```
1. <cfoutput>  
2. #esapiEncode('html', userBean.getFullName())#  
3. </cfoutput>
```

Helper Methods

```
1. <cfset it = userBean.getMembershipsIterator()>
2.
3. <h3>User Memberships</h3>
4. <!-- Loop over the iterator --->
5. <cfif it.hasNext()>
6.   <ul>
7.     <cfloop condition="it.hasNext()">
8.       <cfset item = it.next()>
9.       <li>
10.         #esapiEncode('html', item.get('groupname'))#
11.       </li>
12.     </cfloop>
13.   </ul>
14. <cfelse>
15.   <p>User does not belong to any groups.</p>
16. </cfif>
```

Helper Methods

```
1. | <cfset rsMemberships = userBean.getMembershipsQuery()>
```

Helper Methods

```
1. <cfset it = userBean.getInterestGroupsIterator()>
2.
3. <h3>User Interests</h3>
4. <!-- Loop over the iterator --->
5. <cfif it.hasNext()>
6.   <ul>
7.     <cfloop condition="it.hasNext()">
8.       <cfset item = it.next()>
9.       <li>
10.        #esapiEncode('html', item.get('name'))#
11.      </li>
12.    </cfloop>
13.  </ul>
14. <cfelse>
15.   <p>User does not have any interests.</p>
16. </cfif>
```

Helper Methods

```
1. | <cfset rsInterestGroups = userBean.getInterestGroupsQuery()>
```

Helper Methods

```
1. <cfset it = userBean.getAddressesIterator()>
2.
3. <h3>User Addresses</h3>
4. <!-- Loop over the iterator --->
5. <cfif it.hasNext()>
6.   <ul>
7.     <cfloop condition="it.hasNext()">
8.       <cfset item = it.next()>
9.       <li>
10.         #esapiEncode('html', item.get('addressname'))#
11.       </li>
12.     </cfloop>
13.   </ul>
14. <cfelse>
15.   <p>User does not have any addresses.</p>
16. </cfif>
```

Helper Methods

```
1. | <cfset rsUserAddresses = userBean.getAddressesQuery()>
```

Helper Methods

```
1. <cfif userBean.checkEmail('steve@blueriver.com')>
2.   <!-- Safe to use -->
3.   <cfset userBean.set('email', 'steve@blueriver.com').save()>
4. <cfelse>
5.   <!-- Email address matches another user's email address -->
6. </cfif>
```

Helper Methods

```
1. <cfif userBean.checkUsername('steve')>
2.   <!-- Safe to use -->
3.   <cfset userBean.set('username', 'steve').save()>
4. <cfelse>
5.   <!-- Username matches another user's username -->
6. </cfif>
```

Group Bean

Group Bean

- The **group** bean/object includes group-specific data, and includes a number of helper methods.

Group Bean

- The **group** bean/object includes group-specific data, and includes a number of helper methods.
- For example, you can obtain an iterator of the users who belong to the group, etc.

Group Bean

- **Note:** The **group** bean/object merely uses the **user** bean/object. However, it contains only group-specific information, and group-specific helper methods.

Loading the Bean

- You can load a group bean/object by the following attributes:
groupid, groupname, remoteid

```
1. | <cfset groupBean = m.getBean('group').loadBy(groupname='admin')>
```

Loading the Bean

- You can load a group bean/object by the following attributes:
groupid, groupname, remoteid

```
1. | <cfset groupBean = m.getBean('group').loadBy(groupname='admin')>
```

- An optional second argument, labeled “siteid” is necessary when working with a group assigned to a

```
1. | <cfset groupBean = m.getBean('group').loadBy(groupname='admin', siteid='default')>
```

Getters

1. `groupBean.get('attributeName')`
2. `groupBean.get{AttributeName}()`
3. `groupBean.getValue('attributeName')`

Setters

1. `groupBean.set('attributeName', 'someValue')`
2. `groupBean.set{AttributeName}('someValue')`
3. `groupBean.setValue('attributeName', 'someValue')`

Attributes

- For a listing of attributes, please visit <http://docs.getmura.com/v7-1/mura-developers/mura-beans-objects/common-bean-objects/group-bean/>

Helper Methods

```
1. <cfset it = groupBean.getMembersIterator()>
2.
3. <h3>Group Members</h3>
4. <!-- Loop over the iterator --->
5. <cfif it.hasNext()>
6.   <ul>
7.     <cfloop condition="it.hasNext()">
8.       <cfset user = it.next()>
9.       <li>
10.         #esapiEncode('html', user.getFullName())#
11.       </li>
12.     </cfloop>
13.   </ul>
14. <cfelse>
15.   <p>Group does not have any members.</p>
16. </cfif>
```

Helper Methods

```
1. | <cfset rsMembers = groupBean.getMembersQuery()>
```

Comment Bean

Comment Bean

- The **comment** bean/object includes comment-specific data, and includes helper methods such as whether the comment is approved, deleted, etc.

Loading the Bean

- You can load a comment bean/object by the following attributes:
commentid, remoteid

```
1. | <cfset commentBean = m.getBean('comment').loadBy(commentid='{UUID}')>
```

Getters

1. `commentBean.get('attributeName')`
2. `commentBean.get{AttributeName}()`
3. `commentBean.getValue('attributeName')`

Setters

1. `commentBean.set('attributeName', 'someValue')`
2. `commentBean.set{AttributeName}('someValue')`
3. `commentBean.setValue('attributeName', 'someValue')`

Attributes

- For a listing of attributes, please visit <http://docs.getmura.com/v7-1/mura-developers/mura-beans-objects/common-bean-objects/comment-bean/>

Category Bean

Category Bean

- The **category** bean/object includes category-specific data.

Category Bean

- The **category** bean/object includes category-specific data.
- This bean/object contains information about a category itself, and does not describe any relationships to content items themselves.

Loading the Bean

- You can load a category bean/object by the following attributes:
categoryid, name, filename, urltitle, remoteid

```
1. <cfset categoryBean = m.getBean('category').loadBy(name='Mura CMS')>
```

Getters

1. categoryBean.get('attributeName')
2. categoryBean.get{AttributeName}()
3. categoryBean.getValue('attributeName')

Setters

1. `categoryBean.set('attributeName', 'someValue')`
2. `categoryBean.set{AttributeName}('someValue')`
3. `categoryBean.setValue('attributeName', 'someValue')`

Attributes

- For a listing of attributes, please visit <http://docs.getmura.com/v7-1/mura-developers/mura-beans-objects/common-bean-objects/category-bean/>

Feed Bean

Feed Bean

- The **feed** bean/object allows developers to programmatically query Mura for other beans/objects.

Feed Bean

- The **feed** bean/object allows developers to programmatically query Mura for other beans/objects.
- Initially, many developers query for **content** beans/objects.

Feed Bean

- The **feed** bean/object allows developers to programmatically query Mura for other beans/objects.
- Initially, many developers query for **content** beans/objects.
- In essence, when you search for content items using this method, you're creating a **Collection** dynamically.

Feed Bean

- The **feed** bean/object allows developers to programmatically query Mura for other beans/objects.
- Initially, many developers query for **content** beans/objects.
- In essence, when you search for content items using this method, you're creating a **Collection** dynamically.
- However, you may also query for **user** beans/objects, and any other kind of Mura object or custom object!

Feed Bean

- **Note:** Once the **feed** bean/object has been populated, most developers will call either the **getIterator** or **getQuery** methods to work with the collection of beans/objects.

Loading the Bean

- The first way for obtaining a **feed** bean/object is by using the common **loadBy** method. When loading it this way, keep in mind you will only be working with **content** beans/objects.

Loading the Bean

- The first way for obtaining a **feed** bean/object is by using the common **loadBy** method. When loading it this way, keep in mind you will only be working with **content** beans/objects.
- If you wish to load an existing Collection created via Mura's Admin UI, you can load a feed bean/object by any of the following attributes: feedid, name, remoteid

```
1. | <cfset feedBean = m.getBean('feed').loadBy(name='{Collection Name Goes Here}')>
```

getFeed

- The primary method for obtaining a **feed** bean/object is to use your desired bean's/object's **getFeed** method.

getFeed

- The primary method for obtaining a **feed** bean/object is to use your desired bean's/object's **getFeed** method.
- You can either call **getFeed** on your desired bean/object itself, or pass in the name of the bean as an argument to the **getFeed** method.

Content Feed

```
1. feedBean = m.getBean('content').getFeed();  
2. // or  
3. feedBean = m.getFeed('content');
```

User Feed

```
1. feedBean = m.getBean('user').getFeed();  
2. // or  
3. feedBean = m.getFeed('user');
```

Custom Object Feed

```
1. feedBean = m.getBean('yourCustomObject').getFeed();  
2. // or  
3. feedBean = m.getFeed('yourCustomObject');
```

Key Feed Bean Methods

- where
- prop
- andProp
- orProp
- null
- isEQ
- isNEQ
- isLT
- isLTE
- isGT
- isGTE
- isIn
- isNotIn
- containsValue
- beginsWith
- endsWith
- openGrouping
- andOpenGrouping
- closeGrouping
- sort
- itemsPerPage
- maxItems
- includeHomePage
- showNavOnly
- showExcludeSearch
- isFeaturesOnly
- contentPools
- getIterator
- getQuery

Feed Bean

- **Note:** When using `m.getFeed('content')`, unless you specify otherwise, Mura will only search for content items that are active, approved, set to appear in navigation, not excluded from search results, currently on display, exclude the home page, and limited to the top 20 records.

Feed Bean

- **Note:** When using `m.getFeed('content')`, unless you specify otherwise, Mura will only search for content items that are active, approved, set to appear in navigation, not excluded from search results, currently on display, exclude the home page, and limited to the top 20 records.
- Also, when using `m.getFeed('user')`, by default, Mura will only search for Site Members (`isPublic=1`). To search for System Users, add “`.setIsPublic(0)`” to your method chain.

prop & addProp

```
1. feedBean = m.getFeed('content')
   .where()
   .prop('title')
   .isEQ('About Us')
   .andProp('siteid')
   .isEQ('default');

7.

8. // Analogous SQL
9. SELECT *
10. FROM tcontent
11. WHERE title = 'About Us'
12.   AND siteid = 'default';
```

orProp

```
1. feedBean = m.getFeed('content')
   .where()
   .prop('title')
   .isEQ('About Us')
   .orProp('title')
   .isEQ('News');

7.

8. // Analogous SQL
9. SELECT *
10. FROM tcontent
11. WHERE title = 'About Us'
12.      OR title = 'News';
```

sort

```
1. contentFeed = m.getFeed('content')
   .where()
   .prop('title')
   .isEQ('News')
   .orProp('title')
   .isEQ('Blog')
   .sort('title', 'desc');

8.

9. // Analogous SQL
10. SELECT *
11. FROM tcontent
12. WHERE title = 'News'
13.     OR title = 'Blog'
14. ORDER BY title DESC;
```

getIterator

```
1. <cfset it = feedBean.getIterator()>
2. <!-- Loop over the iterator --->
3. <cfif it.hasNext()>
4.   <ul>
5.     <cfloop condition="it.hasNext()">
6.       <cfset item = it.next()>
7.       <li>
8.         <cfdump var="#item.getAllValues()#">
9.       </li>
10.      </cfloop>
11.    </ul>
12.  <cfelse>
13.    <!-- Feed has no items --->
14.  </cfif>
```

getQuery

```
1. | rsObjects = feedBean.getQuery();
```

contentPools

```
1. feedBean = m.getFeed('content')
   .where()
   .prop('credits')
   .containsValue('Steve')
   .contentPools('site1,site2,site3');

6.

7. // Analogous SQL
8. SELECT *
9. FROM tcontent
10. WHERE credits LIKE '%Steve%'
11.   AND siteid IN ('site1','site2','site3');
```

Grouping Example

```
1. feedBean = m.getFeed('content')
2.         .where()
3.         .prop('credits')
4.         .containsValue('Steve')
5.         .openGrouping()
6.             .prop('title')
7.             .beginsWith('Why')
8.         .closeGrouping()
9.         .orOpenGrouping()
10.            .prop('title')
11.            .isEQ('News')
12.        .closeGrouping();
```

```
14. // Analogous SQL
15. SELECT *
16. FROM tcontent
17. WHERE credits LIKE '%Steve%'
18. AND (
19.     title LIKE 'Why%'
20. ) OR (
21.     title = 'News'
22. );
```

Searching for Super Users

```
1. feedBean = m.getFeed('user')
2.           .where()
3.           .prop('s2') // if s2=1, then they're a super user
4.           .isEQ(1)
5.           .setIsPublic(0);
6.
7. // Analogous SQL
8. SELECT *
9. FROM tusers
10. WHERE s2 = 1
11.   AND ispublic = 0;
```

Searching for Custom Objects

```
1. feedBean = m.getFeed('yourCustomObject')
2.           .where()
3.           .prop('someProperty')
4.           .isEQ('someValue');
5.
6. // Analogous SQL
7. SELECT *
8. FROM yourCustomObjectTable
9. WHERE someProperty = 'someValue';
```

Mura Scope Objects

Mura Scope Objects

- In addition to the CFML scopes, Mura has its own scope called the Mura Scope.

Mura Scope Objects

- In addition to the CFML scopes, Mura has its own scope called the Mura Scope.
- The Mura Scope has a number of subscope objects, that are essentially wrappers of some of Mura's most common bean objects.

Mura Scope Objects

- In addition to the CFML scopes, Mura has its own scope called the Mura Scope.
- The Mura Scope has a number of subscope objects, that are essentially wrappers of some of Mura's most common bean objects.
- In short, they are pre-populated beans, with their own special helper methods.

Mura Scope Objects

- m.globalConfig()
- m.siteConfig()
- m.content()
- m.currentUser()
- m.event() ~ *we've already talked about this one!*

Inspecting Available Attributes

```
1. <cfdump var="#m.{subscope}().getAllValues()#">
2.
3. <!-- For example --->
4. <cfdump var="#m.globalConfig().getAllValues()#">
5. <cfdump var="#m.siteConfig().getAllValues()#">
6. <cfdump var="#m.content().getAllValues()#">
7. <cfdump var="#m.currentUser().getAllValues()#">
8. <cfdump var="#m.event().getAllValues()#">
9. <!-- etc. --->
```

globalConfig

GLOBAL CONFIGURATION SCOPE

globalConfig

- The **globalConfig** is a wrapper of Mura's **configBean** object, and is a subscope object of the Mura Scope.

globalConfig

- The **globalConfig** is a wrapper of Mura's **configBean** object, and is a subscope object of the Mura Scope.
- It includes instance-wide configuration data, such as data collected in the **settings.ini.cfm** file.

Getters

1. `m.globalConfig('attributeName')`
2. `m.globalConfig().get('attributeName')`
3. `m.globalConfig().get{AttributeName}()`
4. `m.globalConfig().getValue('attributeName')`

Setters

- Setting any values with **globalconfig** are only temporary, and last for the duration of the request. In other words, calling the **save()** method will not save any values.

```
1. m.globalConfig('attributeName', 'someValue')
2. m.globalConfig().set('attributeName', 'someValue')
3. m.globalConfig().set{AttributeName}('someValue')
4. m.globalConfig().setValue('attributeName', 'someValue')
```

siteConfig

SITE CONFIGURATION SCOPE

siteConfig

- The **siteConfig** is a wrapper of Mura's **site** bean object, and is a subscope object of the Mura Scope.

siteConfig

- The **siteConfig** is a wrapper of Mura's **site** bean object, and is a subscope object of the Mura Scope.
- It includes site configuration data collected via the Site Settings section in the administration area, as well as other site-specific information.

siteConfig

- The **siteConfig** is a wrapper of Mura's **site** bean object, and is a subscope object of the Mura Scope.
- It includes site configuration data collected via the Site Settings section in the administration area, as well as other site-specific information.
- For example:
 - `m.siteConfig('themeAssetPath')`
 - `m.siteConfig('themeAssetMap')`

Getters

1. `m.siteConfig('attributeName')`
2. `m.siteConfig().get('attributeName')`
3. `m.siteConfig().get{AttributeName}()`
4. `m.siteConfig().getValue('attributeName')`

Setters

1. `m.siteConfig('attributeName', 'someValue')`
2. `m.siteConfig().set('attributeName', 'someValue')`
3. `m.siteConfig().set{AttributeName}('someValue')`
4. `m.siteConfig().setValue('attributeName', 'someValue')`

content

CONTENT SCOPE

content

- The **content** object is a wrapper of Mura's **content** bean object, and is a subscope object of the Mura Scope.

content

- The **content** object is a wrapper of Mura's **content** bean object, and is a subscope object of the Mura Scope.
- It wraps the content bean of the content item being served in the current request.

content

- The **content** object is a wrapper of Mura's **content** bean object, and is a subscope object of the Mura Scope.
- It wraps the content bean of the content item being served in the current request.
- It includes data about a Mura content item, such as the title, summary, body, etc.

content

- The **content** object is a wrapper of Mura's **content** bean object, and is a subscope object of the Mura Scope.
- It wraps the content bean of the content item being served in the current request.
- It includes data about a Mura content item, such as the title, summary, body, etc.
- In addition, there are several content-specific helper methods. For example, you could get an iterator of direct children of the content item, which categories it has been assigned to, etc.

Getters

1. `m.content('attributeName')`
2. `m.content().get('attributeName')`
3. `m.content().get{AttributeName}()`
4. `m.content().getValue('attributeName')`

Setters

1. `m.content('attributeName', 'someValue')`
2. `m.content().set('attributeName', 'someValue')`
3. `m.content().set{AttributeName}('someValue')`
4. `m.content().setValue('attributeName', 'someValue')`

currentUser

CURRENT USER SCOPE

currentUser

- The **currentUser** object is a wrapper of the **sessionUserFacade**, and is a subscope object of the Mura Scope.

currentUser

- The **currentUser** object is a wrapper of the **sessionUserFacade**, and is a subscope object of the Mura Scope.
- Under the hood, Mura allows for access to the **user** bean/object attributes, as well as some special session-specific data, such as whether the current user is logged in, whether the current user is a super user or admin user, etc.

Getters

1. `m.currentUser('attributeName')`
2. `m.currentUser().get('attributeName')`
3. `m.currentUser().get{AttributeName}()`
4. `m.currentUser().getValue('attributeName')`

Setters

1. `m.currentUser('attributeName', 'someValue')`
2. `m.currentUser().set('attributeName', 'someValue')`
3. `m.currentUser().set{AttributeName}('someValue')`
4. `m.currentUser().setValue('attributeName', 'someValue')`

Helper Methods

```
1. <cfif m.currentUser().isLoggedIn()>
2.   <!-- User IS logged in --->
3.   <h3>Hello, #esapiEncode('html', m.currentUser('fname'))#!</h3>
4. <cfelse>
5.   <!-- User is NOT logged in ---->
6.   <h3>Who are you?</h3>
7. </cfif>
```

Helper Methods

```
1. <cfif m.currentUser().isSuperUser()>
2.   <!-- User IS a 'super' user -->
3. <cfelse>
4.   <!-- User is NOT a 'super' user -->
5. </cfif>
```

Helper Methods

```
1. <cfif m.currentUser().isAdminUser()>
2.     <!-- User IS an 'admin' user -->
3. <cfelse>
4.     <!-- User is NOT an 'admin' user -->
5. </cfif>
```

Custom Objects

Custom Objects

- In addition to Mura's beans and objects, developers are able to create their own custom beans and objects.

Custom Objects

- In addition to Mura's beans and objects, developers are able to create their own custom beans and objects.
- We're going to walk through a couple of different processes for creating your own custom objects.

Custom Objects

- In addition to Mura's beans and objects, developers are able to create their own custom beans and objects.
- We're going to walk through a couple of different processes for creating your own custom objects.
- The path you ultimately choose, depends on whether you want to simply add some custom fields to one or more of Mura's common bean objects (i.e., content bean, user bean, etc.), or if you want to create a completely new, custom object altogether.

Custom Objects

- Class Extensions
 - If you're merely looking to customize one of Mura's common bean objects, then you'll want to use Class Extensions.

Custom Objects

- Class Extensions
 - If you're merely looking to customize one of Mura's common bean objects, then you'll want to use Class Extensions.
- Mura ORM
 - If you desire creating a completely new, custom object altogether, you'll definitely want to look at using Mura ORM.

Class Extensions

CUSTOM OBJECTS

Class Extensions

- Mura's Class Extensions allow you to customize some of its common bean objects such as content beans, user beans, group beans, and site beans.

Class Extensions

- Mura's Class Extensions allow you to customize some of its common bean objects such as content beans, user beans, group beans, and site beans.
- The term “class extension” itself is borrowed from the world of object-oriented programming (OOP).

Class Extensions

- Mura's Class Extensions allow you to customize some of its common bean objects such as content beans, user beans, group beans, and site beans.
- The term “class extension” itself is borrowed from the world of object-oriented programming (OOP).
- These “objects” are designed in hierarchies, commonly referred to as a “class” hierarchy.

Class Extensions

- Mura's Class Extensions allow you to customize some of its common bean objects such as content beans, user beans, group beans, and site beans.
- The term “class extension” itself is borrowed from the world of object-oriented programming (OOP).
- These “objects” are designed in hierarchies, commonly referred to as a “class” hierarchy.
- Hence, Mura uses it to define custom subclasses.

Class Extensions

- Class extensions allow you to create **Extended Attribute Sets** (think fieldsets, or groups of related form elements) to provide additional “attributes” (think form fields) to collect and store data.

Class Extensions

- Class extensions allow you to create **Extended Attribute Sets** (think fieldsets, or groups of related form elements) to provide additional “attributes” (think form fields) to collect and store data.
- In addition, you can create **Related Content Sets** to allow content managers the ability to associate content to something other than the default catch-all “Related Content” field, allowing you to create fields such as Related Videos, Related Files, Related Authors, and so on.

Class Extensions

- Also, class extensions provide an excellent way to allow developers the ability to target a certain “Type” and/or “Subtype” of a content item’s body area, so they can do things such as automatically alter the layout of the body area, and/or include additional data, etc.

Class Extensions

- Also, class extensions provide an excellent way to allow developers the ability to target a certain “Type” and/or “Subtype” of a content item’s body area, so they can do things such as automatically alter the layout of the body area, and/or include additional data, etc.
- Using this method means you wouldn’t necessarily have to create any Extended Attribute Sets or Related Content Sets either, unless that’s something you desire.

Types & Subtypes

CLASS EXTENSIONS

Types & Subtypes

- Each of the bean objects Mura exposes for customization include two very important attributes: “**Type**”, and “**Subtype**”.

Types & Subtypes

- Each of the bean objects Mura exposes for customization include two very important attributes: “**Type**”, and “**Subtype**”.
- While you can change the “**Subtype**” attribute of a bean object, you cannot alter the “**Type**” attribute itself.

Types & Subtypes

- When working with class extensions, you first target the specific “**Type**” for customization.

Types & Subtypes

- When working with class extensions, you first target the specific “**Type**” for customization.
- Then, you target the “default” **Subtype**, or create a new **Subtype** altogether.

Types & Subtypes

- When working with class extensions, you first target the specific “**Type**” for customization.
- Then, you target the “default” **Subtype**, or create a new **Subtype** altogether.
- These are often represented as **{Type} / {Subtype}** when creating new objects.

Types & Subtypes

- When working with class extensions, you first target the specific “**Type**” for customization.
- Then, you target the “default” **Subtype**, or create a new **Subtype** altogether.
- These are often represented as **{Type} / {Subtype}** when creating new objects.
- For example, “Folder/Locations”, “Page/Location”, “User/Employee”, etc.

Types & Subtypes

- While the options for customizations are essentially limitless, here's a few examples to help illustrate how you might use class extensions.

Types & Subtypes

- **User**
 - You could create a new "User" subtype, such as "Employee". Then, you might want to collect additional data about the employee such as "Emergency Contact Name", "Emergency Contact Mobile", etc.

Types & Subtypes

- **Folder**
 - You could create a new "Folder" subtype, such as "Locations".

Types & Subtypes

- **Folder**
 - You could create a new "Folder" subtype, such as "Locations".
 - Then, you could target any content items using "Folder/Locations" as the **Type** and **Subtype** in order to loop over any child content items and display them on a map.

Types & Subtypes

- **Folder**
 - You could create a new "Folder" subtype, such as "Locations".
 - Then, you could target any content items using "Folder/Locations" as the **Type** and **Subtype** in order to loop over any child content items and display them on a map.
 - You wouldn't necessarily have to add any Extended Attribute Sets or Related Content Sets either.

Types & Subtypes

- **Folder**
 - You could create a new "Folder" subtype, such as "Locations".
 - Then, you could target any content items using "Folder/Locations" as the **Type** and **Subtype** in order to loop over any child content items and display them on a map.
 - You wouldn't necessarily have to add any Extended Attribute Sets or Related Content Sets either.
 - You can also restrict the {Type}/{Subtype} of content that can be added as children.

Types & Subtypes

- **Page**
 - Along the same lines of the previous Folder example, you could create a new "Page" subtype, such as "Location".

Types & Subtypes

- **Page**
 - Along the same lines of the previous Folder example, you could create a new "Page" subtype, such as "Location".
 - Then, you could collect data about the location such as the Street, City, State, Phone Number, etc. and display the information entered into these fields on the page when it's being rendered.

Types & Subtypes

- **Page**
 - Along the same lines of the previous Folder example, you could create a new "Page" subtype, such as "Location".
 - Then, you could collect data about the location such as the Street, City, State, Phone Number, etc. and display the information entered into these fields on the page when it's being rendered.
 - In addition, you could use this data to geocode the location for use on map.

The “default” Subtype

The “default” Subtype

- By targeting “default” as the **Subtype**, any **Extended Attribute Sets** or **Related Content Sets** associated with the class extension will apply to *all* objects sharing the same **Type** attribute.

The “default” Subtype

- By targeting “default” as the **Subtype**, any **Extended Attribute Sets** or **Related Content Sets** associated with the class extension will apply to ***all*** objects sharing the same **Type** attribute.
- For example, if you create a new class extension and the **Type** equals “Page” and **Subtype** equals “Default”, then anything you create will apply to all “Page/Default” content items, as well as any other “Page/{Anything}” content items.

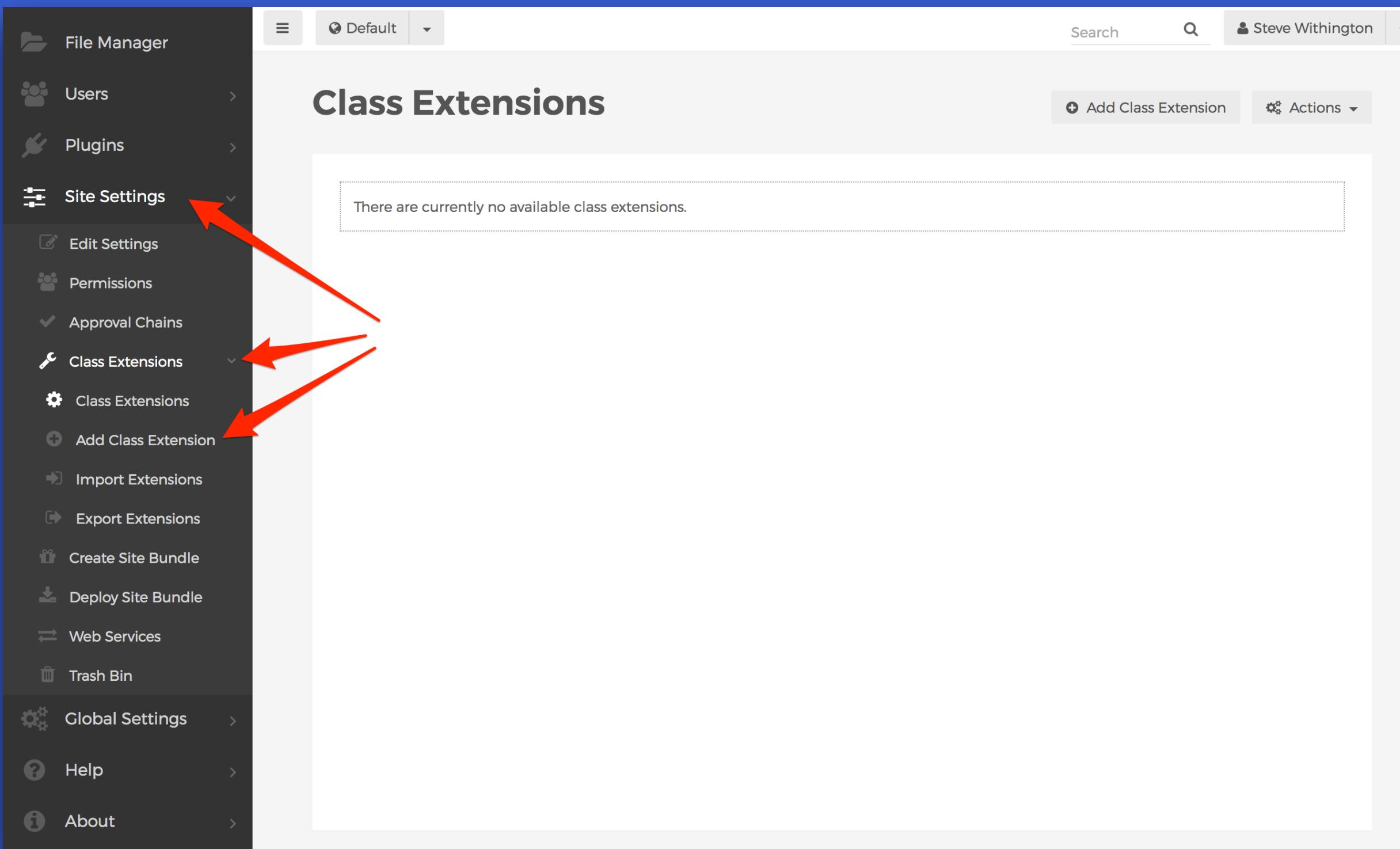
Extendable Bean Objects

- Page
- Folder
- File
- Calendar
- Link
- Component
- Form
- Base
- User (2)
- Group (1)
- Address
- Site

Define with Admin UI

CLASS EXTENSIONS

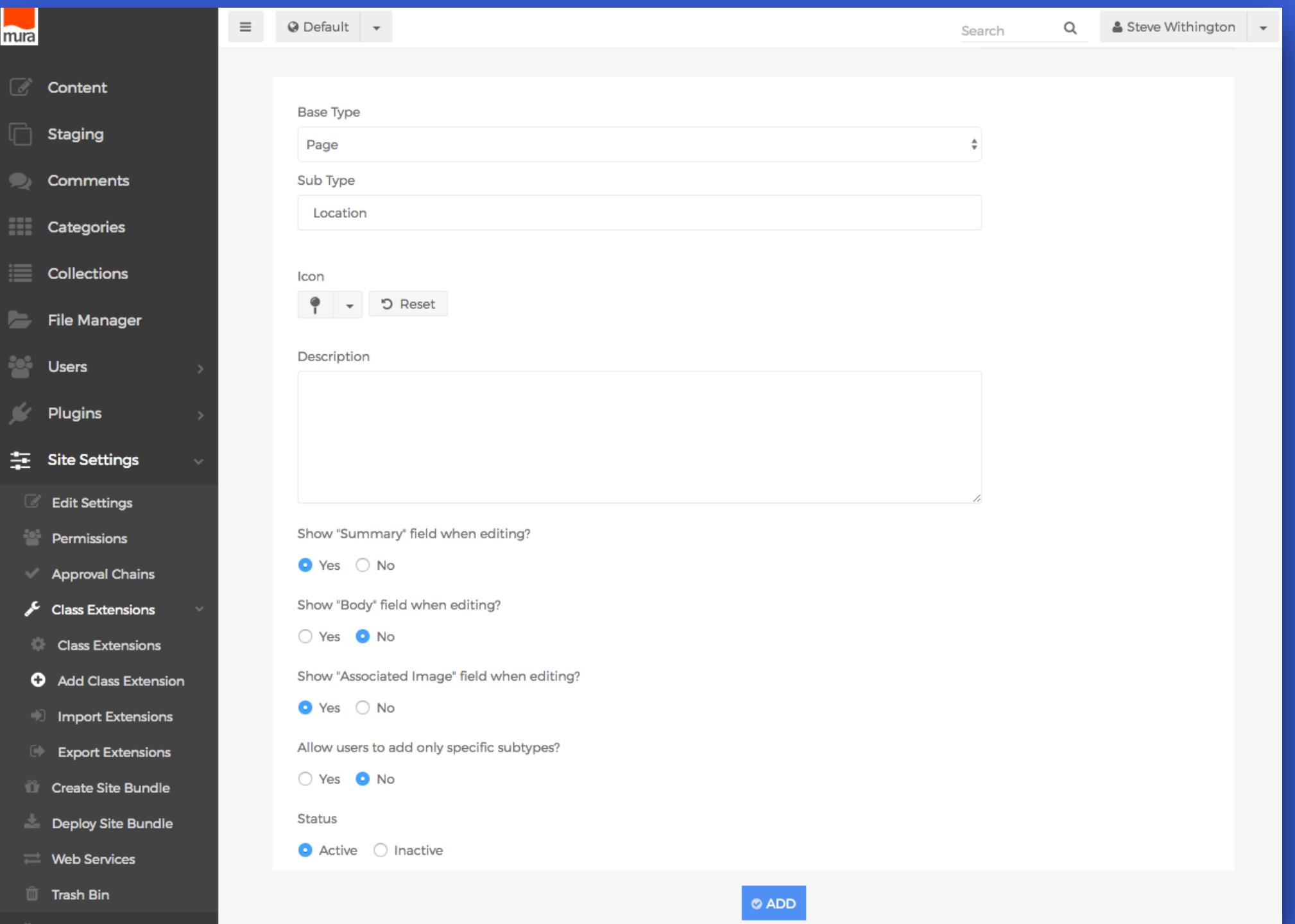
Define with Admin UI



Define with Admin UI

The screenshot shows the Mura CMS Admin UI with a dark sidebar and a light main content area. The sidebar includes links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings (with sub-links for Edit Settings, Permissions, Approval Chains, and Class Extensions), and actions for Add Class Extension, Import Extensions, and Export Extensions. The main content area has a header 'Add Class Extension' and a 'Back to Class Extension Overview' link. It contains fields for 'Base Type' (a dropdown menu currently set to 'Select'), 'Description' (a large text area), and 'Status' (radio buttons for Active and Inactive, with Active selected). A blue 'ADD' button is at the bottom right.

Define with Admin UI



Define with Admin UI

The screenshot shows the Mura CMS Admin UI with the following interface elements:

- Header:** Includes the Mura logo, a "Default" environment dropdown, a search bar, and a user profile for "Steve Withington".
- Left Sidebar:** A dark sidebar with white icons and text links:
 - Content
 - Staging
 - Comments
 - Categories
 - Collections
 - File Manager
 - Users
 - Plugins
 - Site Settings
 - Edit Settings
 - Permissions
 - Approval Chains
 - Class Extensions
 - Class Extensions
 - Add Class Extension
 - Import Extensions
 - Export Extensions
- Main Content Area:** The title "Class Extension" is displayed. Below it are several buttons: "Back to Class Extensions", "Edit Class Extension", "Export Class Extension", and a "Add" button with a dropdown arrow.
- Extended Attribute Sets Section:** This section contains two panels:
 - Page / Location:** A panel stating "There are currently no available attribute sets."
 - Extended Attribute Sets:** A panel stating "There are currently no related content sets."

Extended Attribute Sets

Extended Attribute Sets

- Extended Attribute Sets are essentially **<fieldset>** elements.

Extended Attribute Sets

- Extended Attribute Sets are essentially `<fieldset>` elements.
- In other words, they're a set of form controls, grouped under a common name.

Extended Attribute Sets

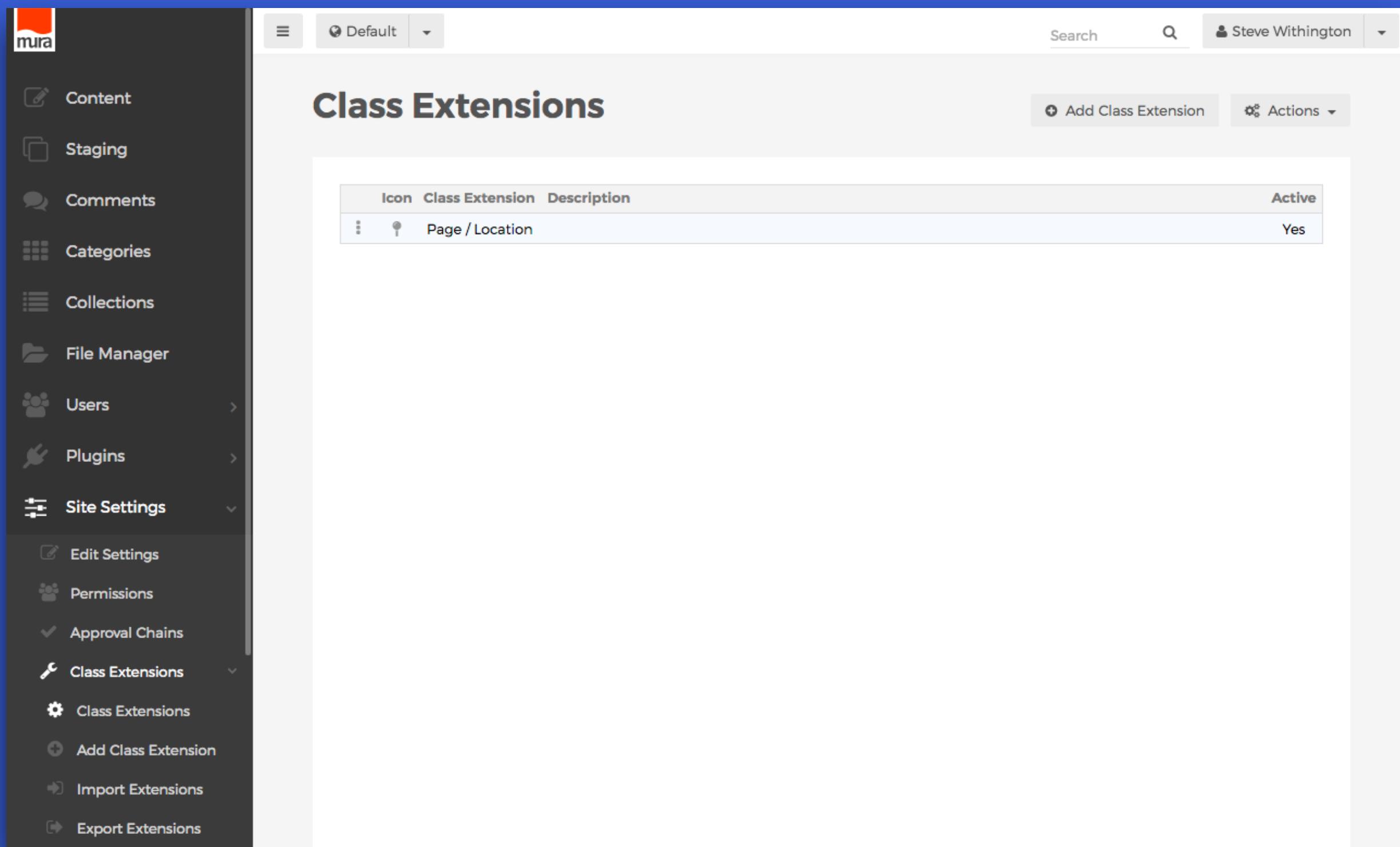
The screenshot shows the mura CMS interface with the following details:

- Left Sidebar (Dark Theme):**
 - Content
 - Staging
 - Comments
 - Categories
 - Collections
 - File Manager
 - Users
 - Plugins
 - Site Settings** (highlighted with a red arrow)
 - Class Extensions** (highlighted with a red arrow)
 - Add Class Extension
 - Import Extensions
 - Export Extensions
- Top Bar:** Default, Search, Steve Withington
- Main Content Area:**

Class Extensions

| Icon | Class Extension | Description | Active |
|-----------------|-----------------|-------------|--------|
| Page / Location | | | Yes |

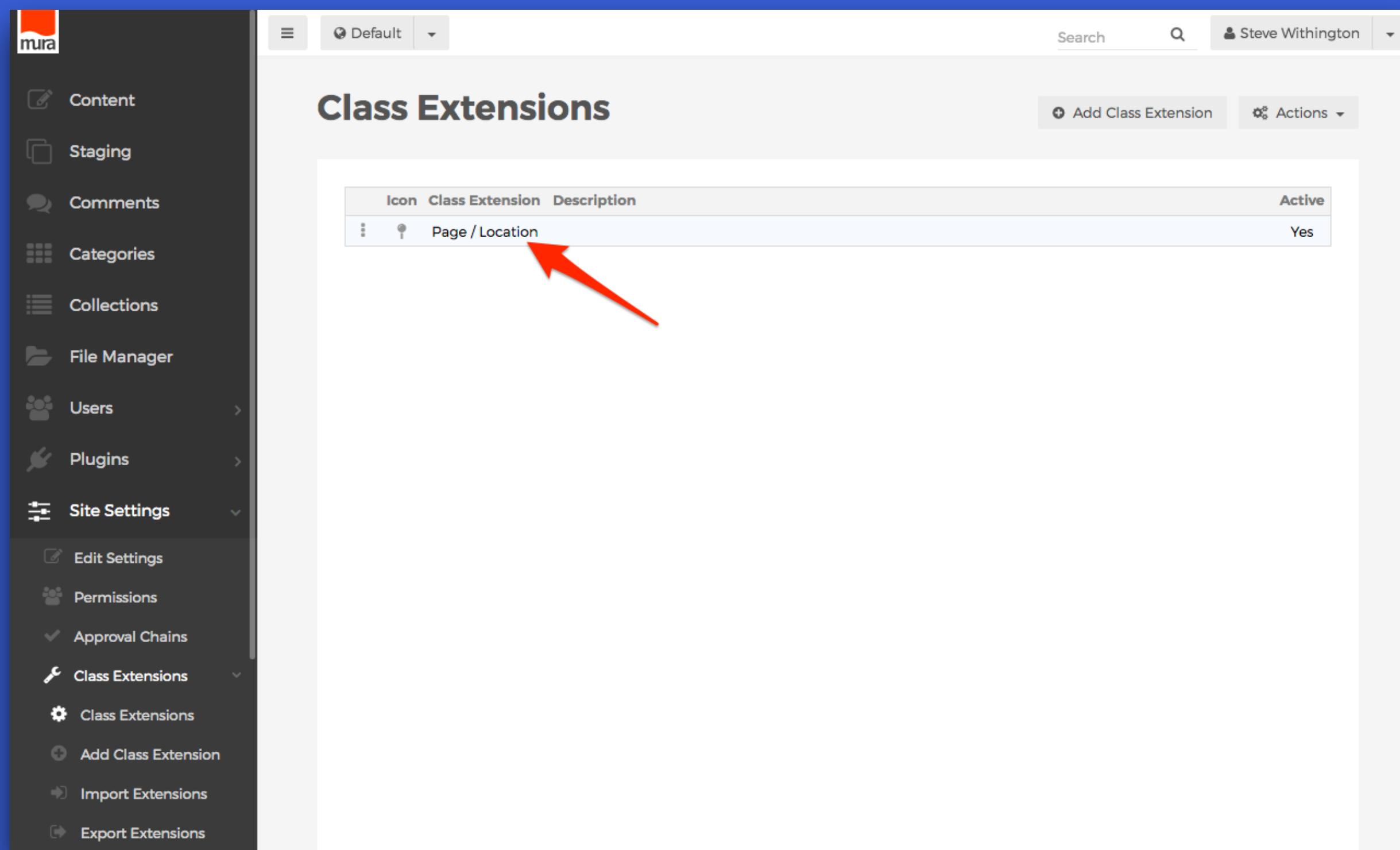
Extended Attribute Sets



The screenshot shows the 'Class Extensions' page in the Mura CMS interface. The left sidebar contains navigation links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, and Site Settings. Under Site Settings, 'Class Extensions' is listed under 'Edit Settings'. The main content area is titled 'Class Extensions' and displays a table with one row. The table columns are 'Icon', 'Class Extension', 'Description', and 'Active'. The single row shows an icon of a document with a gear, the class extension 'Page / Location', the description 'Page / Location', and the status 'Yes' under 'Active'.

| Icon | Class Extension | Description | Active |
|------|-----------------|-----------------|--------|
| 📄 | Page / Location | Page / Location | Yes |

Extended Attribute Sets



The screenshot shows the mura CMS interface with a dark sidebar on the left and a light-colored main content area. The sidebar contains various navigation items: Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, and Site Settings. Under Site Settings, there are sub-options: Edit Settings, Permissions, Approval Chains, Class Extensions, Add Class Extension, Import Extensions, and Export Extensions. The main content area is titled "Class Extensions". It features a search bar at the top right with the placeholder "Search" and a dropdown menu showing "Default". Below the search bar is a user profile for "Steve Withington". A red arrow points to the first row of a table. The table has columns: Icon, Class Extension, Description, and Active. The first row shows an icon of a page with a location pin, the class extension "Page / Location", the description "Page / Location", and the status "Active". There is also a "Yes" link under the Active column.

| Icon | Class Extension | Description | Active |
|---|-----------------|-----------------|---------------------|
|  | Page / Location | Page / Location | Yes |

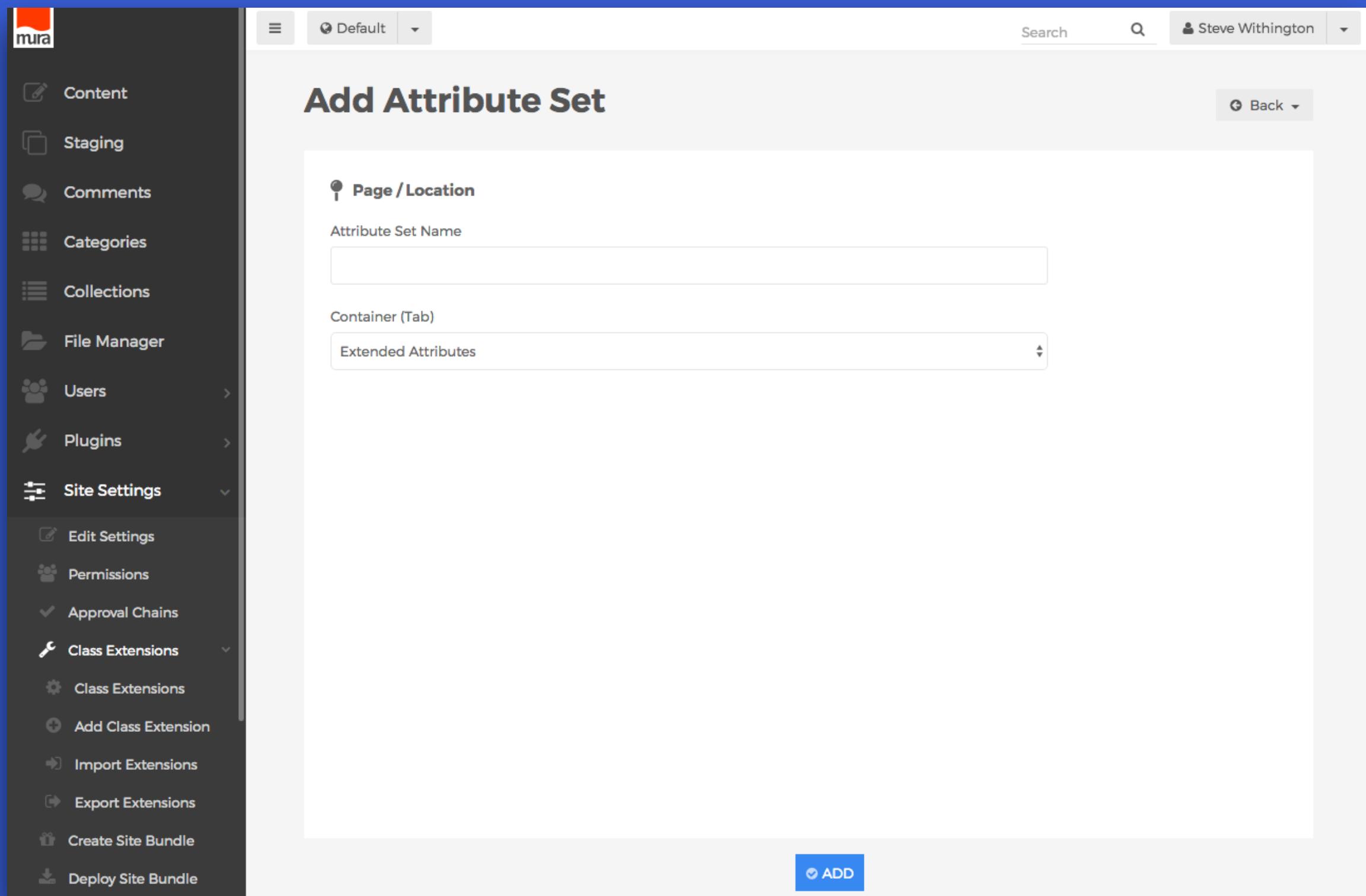
Extended Attribute Sets

The screenshot shows the Mura CMS interface. The left sidebar has a dark theme with white icons and text. It includes links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, and Site Settings. Under Site Settings, there are sub-links for Edit Settings, Permissions, Approval Chains, Class Extensions, Add Class Extension, Import Extensions, and Export Extensions. The main content area has a light gray background. At the top, it says "Class Extension" with buttons for Back to Class Extensions, Edit Class Extension, and Export Class Extension. There is also an "Add" button. Below this, under "Page / Location", there is a section titled "Extended Attribute Sets" which states "There are currently no available attribute sets." Another section below it states "There are currently no related content sets."

Extended Attribute Sets

The screenshot shows the 'Class Extension' page in the Mura CMS interface. The left sidebar contains navigation links such as Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, and Site Settings. Under Site Settings, 'Class Extensions' is expanded, showing 'Class Extensions', 'Add Class Extension', 'Import Extensions', and 'Export Extensions'. The main content area is titled 'Class Extension' and includes buttons for 'Back to Class Extensions', 'Edit Class Extension', and 'Export Class Extension'. A search bar and user profile are also present. A red arrow points to the 'Add' button in the top right corner of the content area, which has a dropdown menu open. The menu items are 'Add Attribute Set' and 'Add Related Content Set'. Below the menu, sections for 'Page / Location' and 'Extended Attribute Sets' are visible, both currently showing no available items.

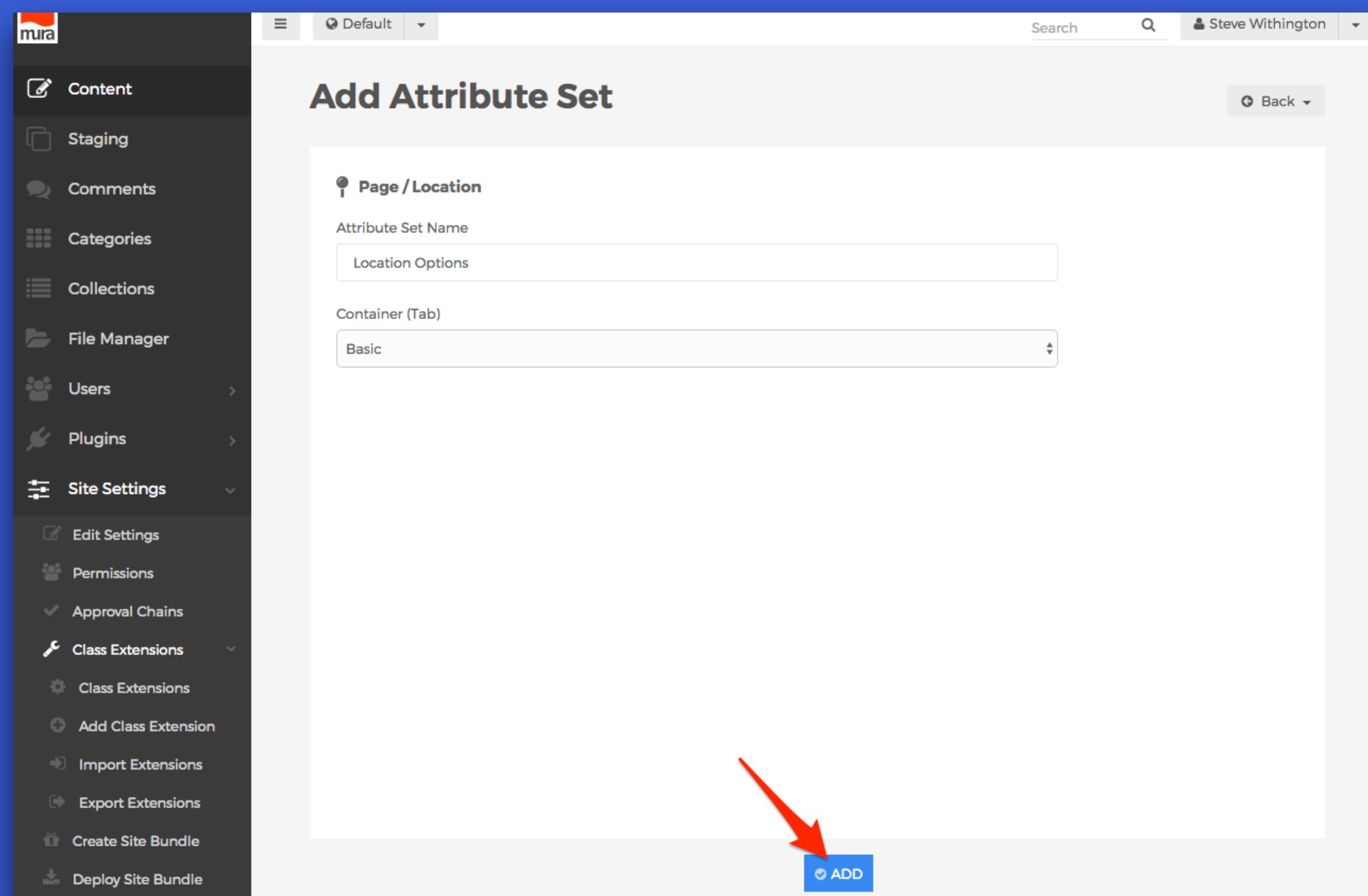
Extended Attribute Sets



The screenshot shows the Mura CMS interface for adding a new attribute set. The left sidebar contains navigation links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings (with sub-options like Edit Settings, Permissions, Approval Chains, Class Extensions, Add Class Extension, Import Extensions, Export Extensions, Create Site Bundle, and Deploy Site Bundle), and a search bar at the top.

The main content area is titled "Add Attribute Set". It has a "Page / Location" section with a "Container (Tab)" dropdown menu. The "Container (Tab)" dropdown is open, showing the option "Extended Attributes" highlighted. At the bottom right of the form is a blue "ADD" button.

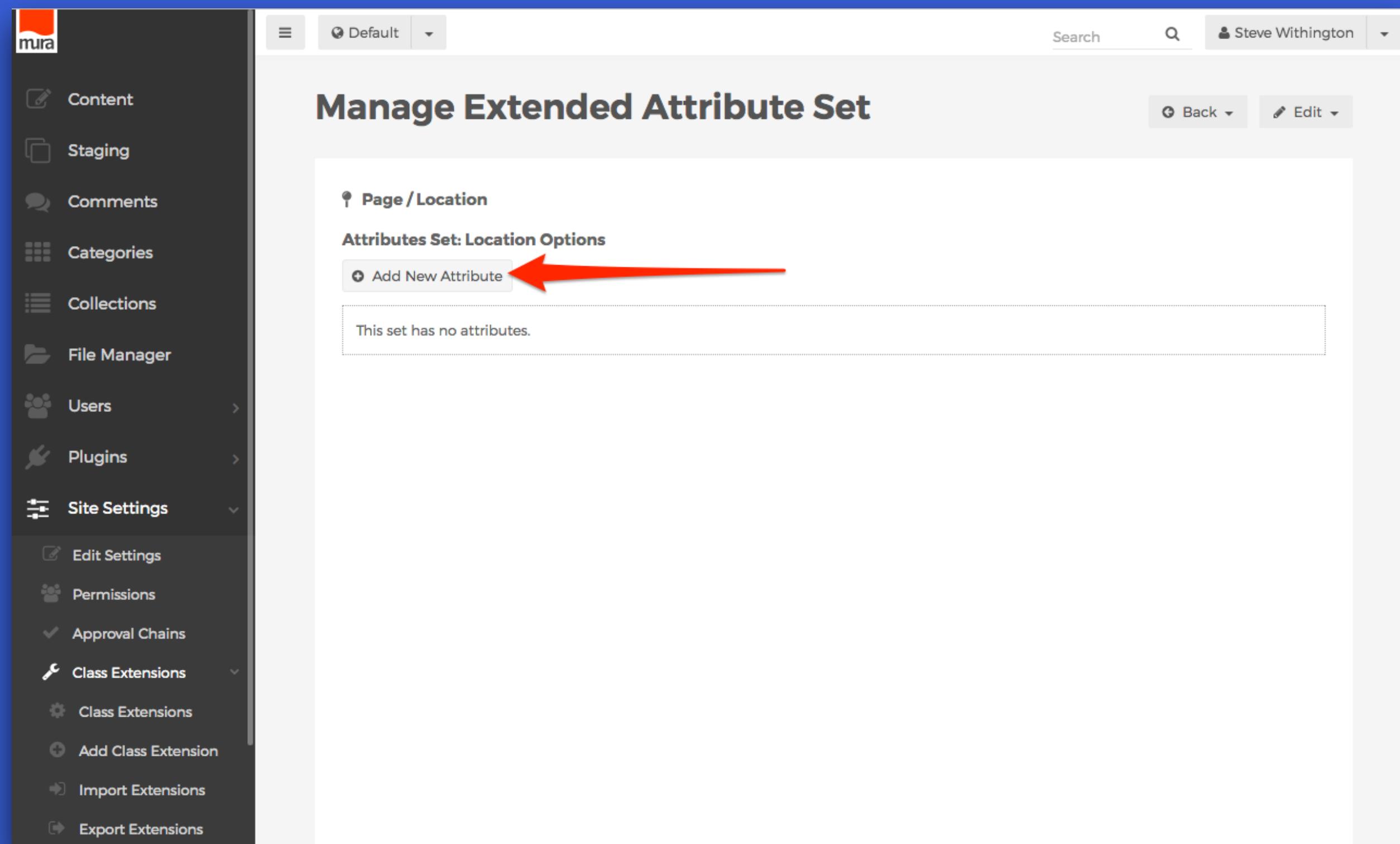
Extended Attribute Sets



Extended Attribute Sets

The screenshot shows the 'Manage Extended Attribute Set' interface in the Mura CMS. The left sidebar contains navigation links: Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, and Site Settings (which is expanded to show Edit Settings, Permissions, Approval Chains, Class Extensions, Import Extensions, and Export Extensions). The main content area is titled 'Manage Extended Attribute Set' and shows 'Attributes Set: Location Options'. It features a button to 'Add New Attribute' and a message stating 'This set has no attributes.'

Extended Attribute Sets



The screenshot shows the 'Manage Extended Attribute Set' interface in the Mura CMS. On the left is a dark sidebar with various navigation items: Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, and Site Settings (which is expanded to show Edit Settings, Permissions, Approval Chains, Class Extensions, and Import/Export options). The main content area is titled 'Manage Extended Attribute Set' and has a sub-section titled 'Attributes Set: Location Options'. Within this section, there is a button labeled '+ Add New Attribute' with a red arrow pointing to it. Below the button, a message states 'This set has no attributes.'

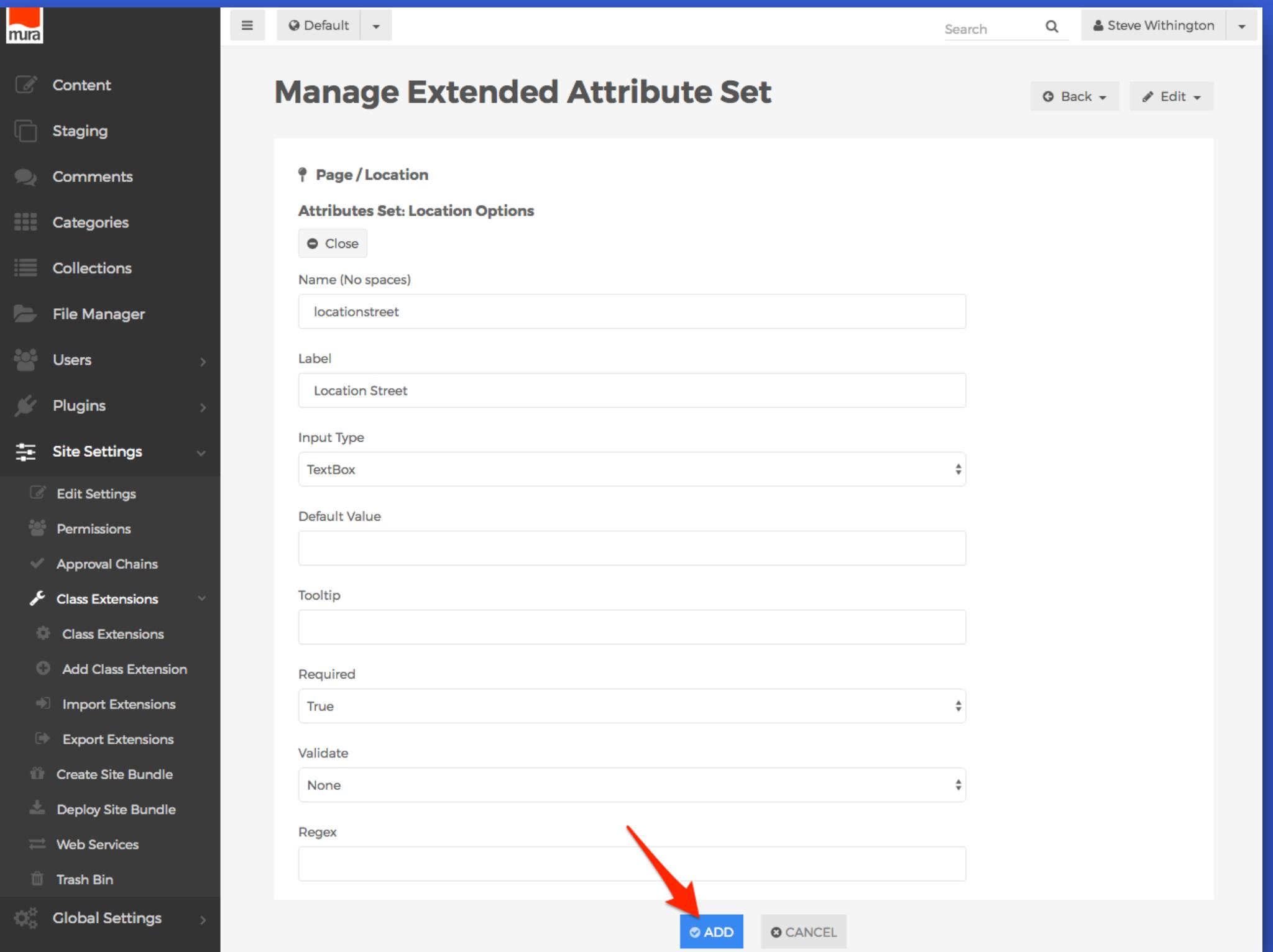
Extended Attribute Sets

The screenshot shows the 'Manage Extended Attribute Set' dialog box in the Mura CMS interface. The dialog is titled 'Manage Extended Attribute Set' and is located under the 'Page / Location' section of the 'Attributes Set: Location Options'. The dialog contains several input fields:

- Name (No spaces): A text input field.
- Label: A text input field.
- Input Type: A dropdown menu set to 'TextBox'.
- Default Value: An empty text input field.
- Tooltip: An empty text input field.
- Required: A dropdown menu set to 'False'.
- Validate: A dropdown menu set to 'None'.
- Regex: An empty text input field.

At the bottom of the dialog are two buttons: a blue 'ADD' button and a grey 'CANCEL' button.

Extended Attribute Sets



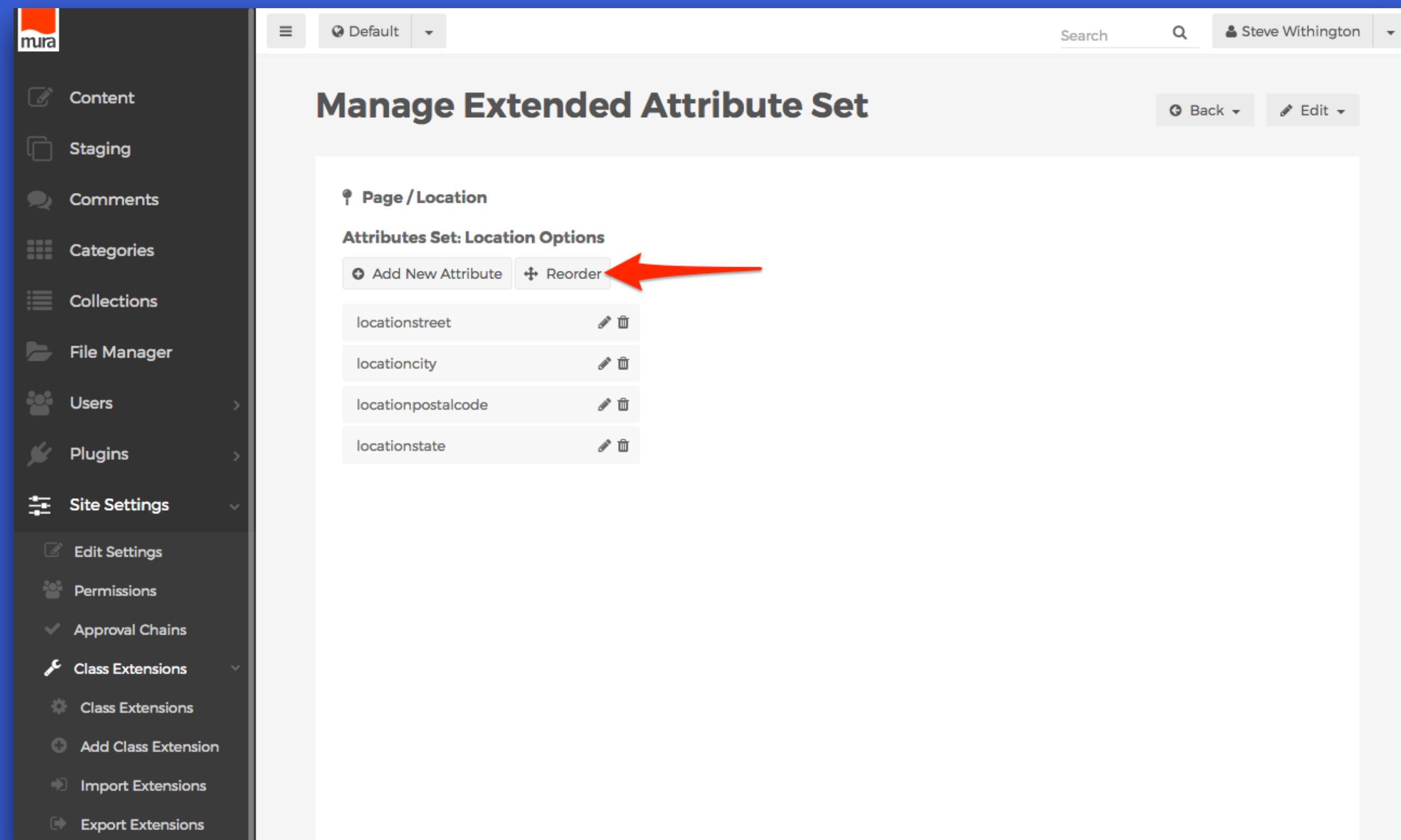
The screenshot shows the 'Manage Extended Attribute Set' dialog in the Mura CMS interface. The left sidebar contains navigation links such as Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings (with sub-options like Edit Settings, Permissions, Approval Chains, Class Extensions, Add Class Extension, Import Extensions, Export Extensions, Create Site Bundle, Deploy Site Bundle, Web Services, and Trash Bin), and Global Settings.

The main dialog title is 'Manage Extended Attribute Set' under the 'Page / Location' section. It is specifically configured for 'Attributes Set: Location Options'. The configuration fields include:

- Name (No spaces): locationstreet
- Label: Location Street
- Input Type: TextBox
- Default Value: (empty)
- Tooltip: (empty)
- Required: True
- Validate: None
- Regex: (empty)

A red arrow points to the blue 'ADD' button at the bottom of the dialog.

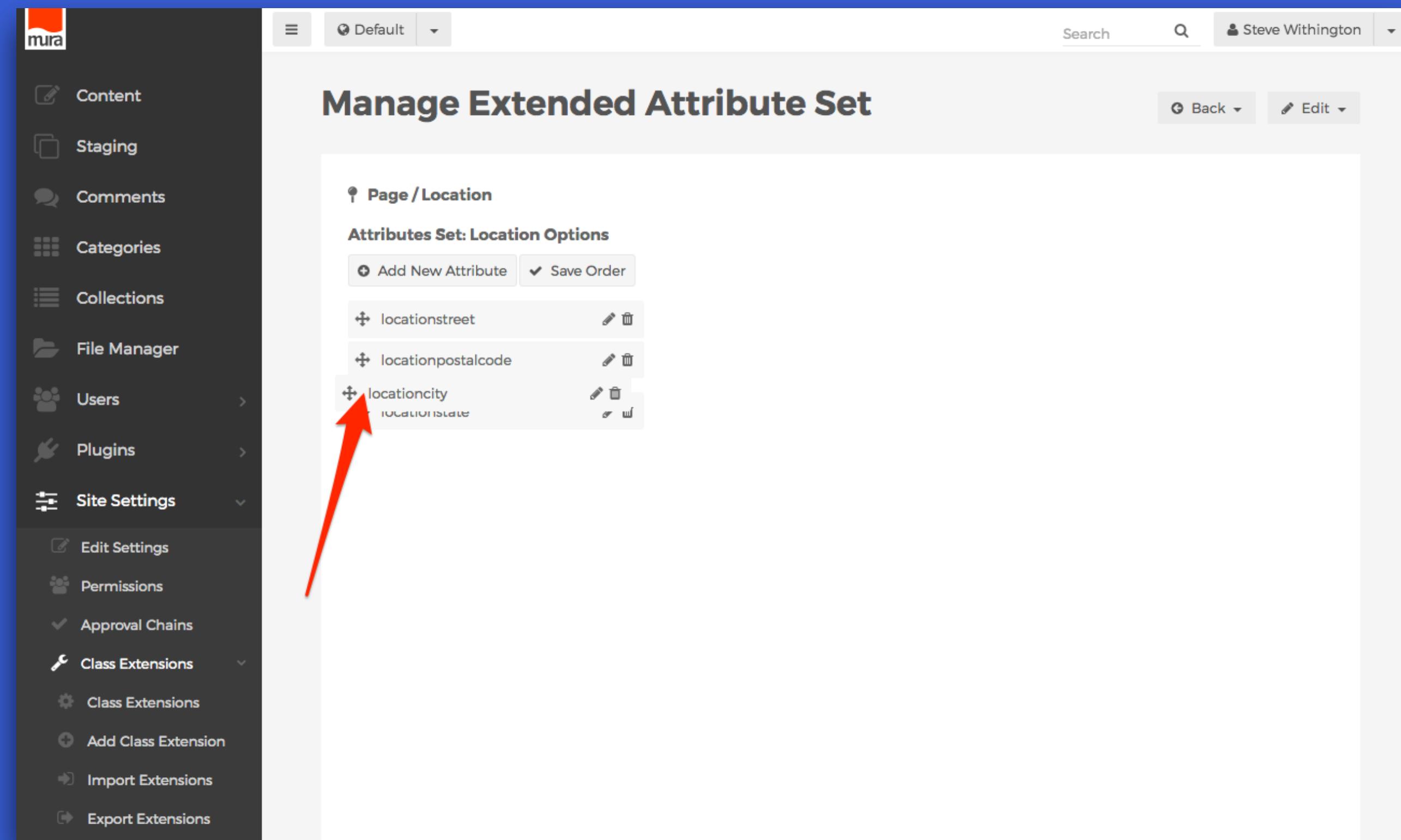
Extended Attribute Sets



The screenshot shows the 'Manage Extended Attribute Set' interface in the Mura CMS. On the left is a dark sidebar with various navigation items: Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings (with sub-options like Edit Settings, Permissions, Approval Chains, Class Extensions, Add Class Extension, Import Extensions, and Export Extensions), and a bottom section for Class Extensions.

The main content area is titled 'Manage Extended Attribute Set' and has a sub-section titled 'Attributes Set: Location Options'. It contains four items: 'locationstreet', 'locationcity', 'locationpostalcode', and 'locationstate', each with edit and delete icons. At the top of this list are two buttons: 'Add New Attribute' and 'Reorder'. A red arrow points to the 'Reorder' button.

Extended Attribute Sets



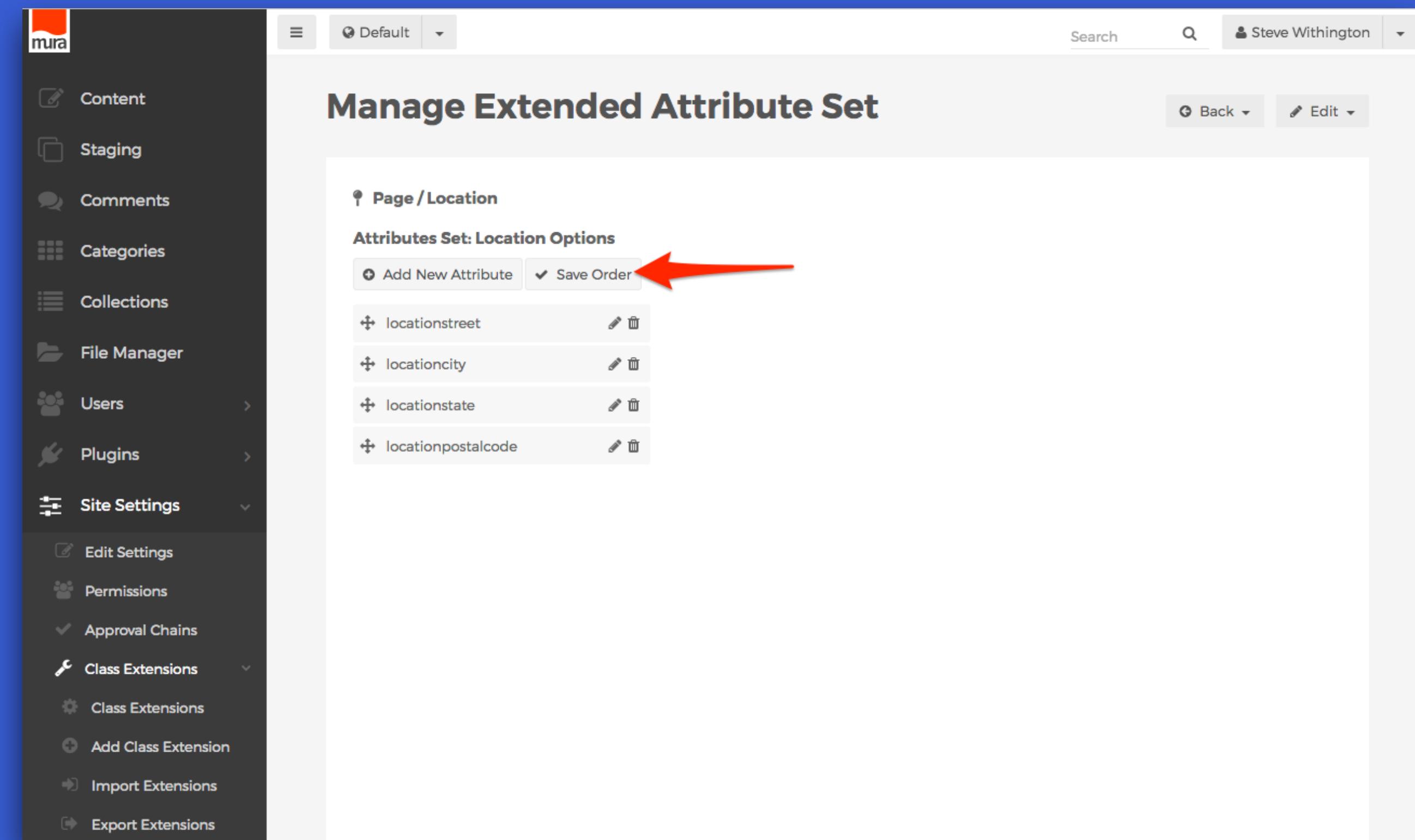
The screenshot shows the 'Manage Extended Attribute Set' page in the Mura CMS. The left sidebar contains navigation links such as Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings (with sub-options like Edit Settings, Permissions, Approval Chains, Class Extensions, Import Extensions, and Export Extensions), and a mura logo.

The main content area is titled 'Manage Extended Attribute Set' and shows the 'Attributes Set: Location Options' section. It includes a 'Page / Location' header, a toolbar with 'Add New Attribute' and 'Save Order' buttons, and a list of attributes:

- locationstreet
- locationpostalcode
- locationcity
- locationstate

A red arrow points to the 'locationcity' attribute entry. The interface also features standard navigation buttons like Back and Edit.

Extended Attribute Sets



The screenshot shows the 'Manage Extended Attribute Set' interface in the Mura CMS. The left sidebar contains navigation links such as Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings (with sub-options like Edit Settings, Permissions, Approval Chains, Class Extensions, Import Extensions, and Export Extensions), and mura. The main content area is titled 'Manage Extended Attribute Set' and shows a section for 'Attributes Set: Location Options'. It lists four attributes: locationstreet, locationcity, locationstate, and locationpostalcode. Each attribute has a plus sign icon, a pencil icon for editing, and a trash can icon for deletion. At the top of this section, there are two buttons: 'Add New Attribute' and 'Save Order'. A red arrow points to the 'Save Order' button. The top right of the screen shows a user profile for 'Steve Withington'.

Related Content Sets

Related Content Sets

- Related Content Sets allow content managers the ability to associate content to something other than the default catch-all “Related Content” field, allowing you to create fields such as Related Videos, Related Files, Related Authors, and so on.

Related Content Sets

The screenshot shows the Mura CMS interface with a dark sidebar and a light main content area. The sidebar contains the following navigation items:

- Content
- Staging
- Comments
- Categories
- Collections
- File Manager
- Users
- Plugins
- Site Settings
 - Edit Settings
 - Permissions
 - Approval Chains
 - Class Extensions
 - Class Extensions
 - Add Class Extension
 - Import Extensions
 - Export Extensions

Three red arrows point from the text "Related Content Sets" in the previous slide to the "Class Extensions" menu item in the sidebar.

The main content area is titled "Class Extensions". It features a search bar, a user dropdown, and two buttons: "Add Class Extension" and "Actions". A table displays one entry:

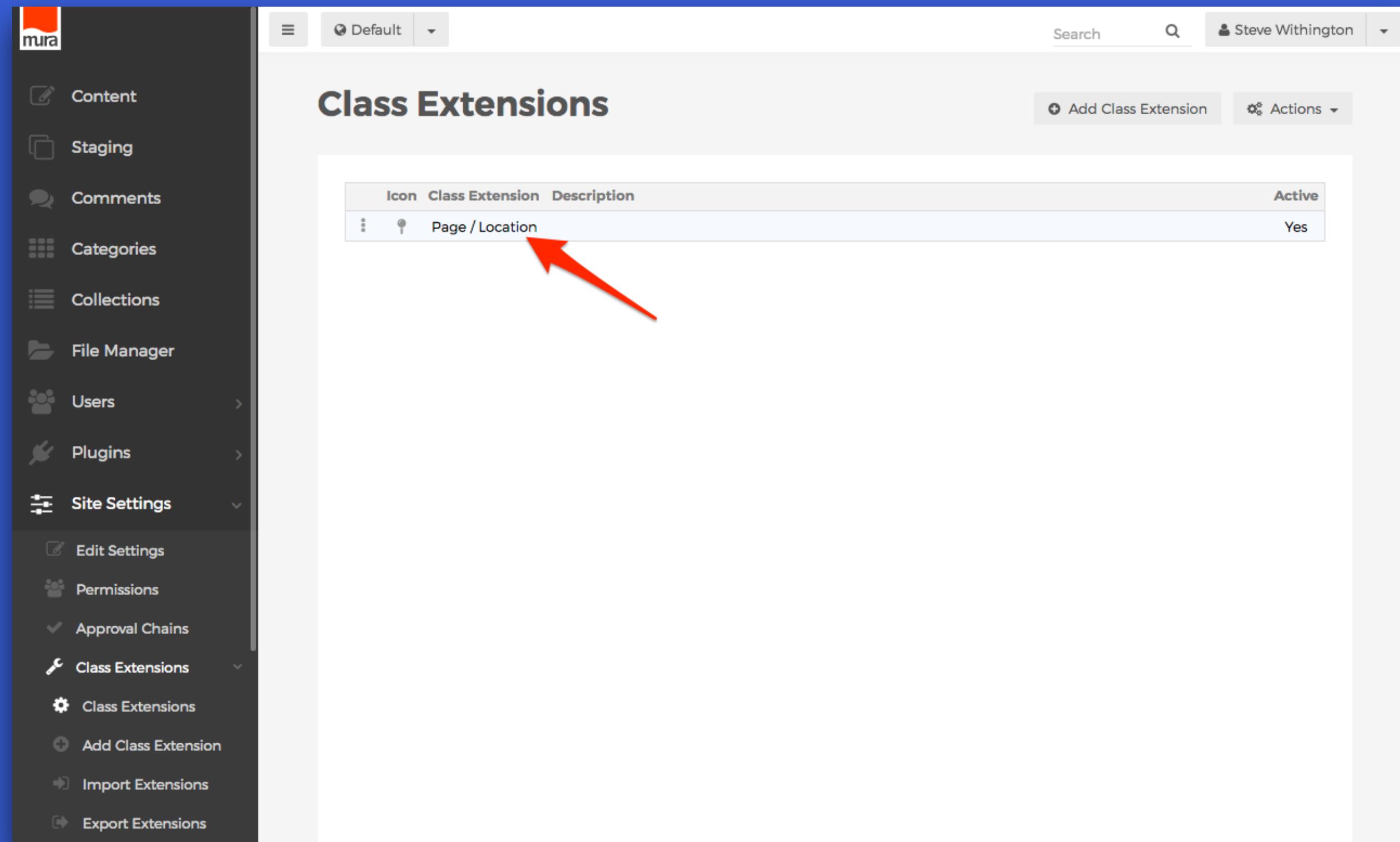
| Icon | Class Extension | Description | Active |
|------|-----------------|-------------|--------|
| • | Page / Location | | Yes |

Related Content Sets

The screenshot shows the Mura CMS interface with a dark sidebar and a light main content area. The sidebar includes links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, and Site Settings (with sub-links for Edit Settings, Permissions, Approval Chains, and Class Extensions). The main content area is titled "Class Extensions" and displays a table with one row. The table columns are Icon, Class Extension, Description, and Active. The single row contains an icon of a person, the text "Page / Location", and the status "Yes". There are "Add Class Extension" and "Actions" buttons at the top right of the table.

| Icon | Class Extension | Description | Active |
|-------------|-----------------|-------------|--------|
| Person icon | Page / Location | | Yes |

Related Content Sets



The screenshot shows the Mura CMS interface with the 'Class Extensions' page open. The left sidebar contains navigation links such as Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings (with sub-options like Edit Settings, Permissions, Approval Chains, Class Extensions, Add Class Extension, Import Extensions, and Export Extensions), and a search bar at the top.

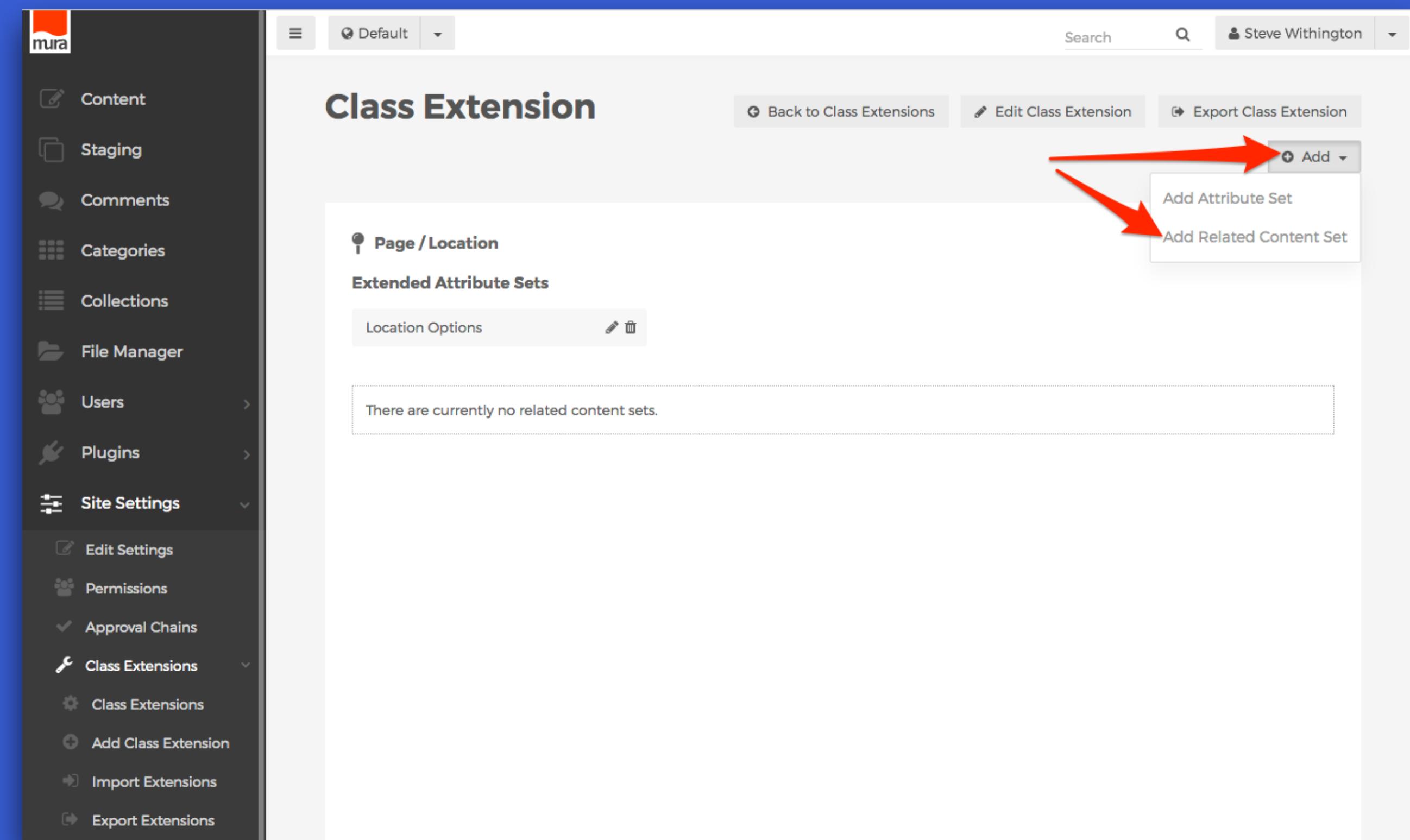
The main content area is titled 'Class Extensions' and displays a table with one row. The table columns are 'Icon', 'Class Extension', 'Description', and 'Active'. The single row shows an icon of a page, the class extension 'Page / Location', a description 'Yes', and the status 'Active'. A red arrow points to the 'Page / Location' column.

| Icon | Class Extension | Description | Active |
|-----------|-----------------|-------------|--------|
| Page icon | Page / Location | Yes | Yes |

Related Content Sets

The screenshot shows the Mura CMS interface for managing class extensions. The left sidebar contains navigation links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, and Site Settings (with sub-options for Edit Settings, Permissions, Approval Chains, Class Extensions, Add Class Extension, Import Extensions, and Export Extensions). The main content area is titled "Class Extension" and includes buttons for Back to Class Extensions, Edit Class Extension, Export Class Extension, and Add. A section titled "Page / Location" indicates there are currently no available attribute sets. Another section titled "Extended Attribute Sets" also indicates there are currently no related content sets.

Related Content Sets



The screenshot shows the mura CMS interface with a dark sidebar and a light main content area. The sidebar includes links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, and Site Settings (with sub-links for Edit Settings, Permissions, Approval Chains, Class Extensions, Import Extensions, and Export Extensions). The main content area is titled "Class Extension". It features a "Page / Location" section with a "Location Options" button and a message stating "There are currently no related content sets." At the top right, there is a "Back to Class Extensions" link, an "Edit Class Extension" button, and an "Export Class Extension" button. A red arrow points to a dropdown menu labeled "Add" which contains two options: "Add Attribute Set" and "Add Related Content Set".

Related Content Sets

The screenshot shows the Mura CMS interface with a dark sidebar on the left and a light-colored main content area. The sidebar contains the following navigation items:

- Content
- Staging
- Comments
- Categories
- Collections
- File Manager
- Users
- Plugins
- Site Settings
 - Edit Settings
 - Permissions
 - Approval Chains
 - Class Extensions
 - Class Extensions
 - Add Class Extension
 - Import Extensions
 - Export Extensions

The main content area has a title "Add Related Content Set" and two buttons: "Back to Class Extensions" and "Back to Extension Overview". Below the title, there is a section titled "Page / Location" with a "Related Content Set Name" input field. Underneath it, there is a question "Allow users to add only specific subtypes?" with two radio button options: "Yes" (unchecked) and "No" (checked).

Related Content Sets

The screenshot shows the 'Add Related Content Set' dialog box in the Mura CMS interface. The left sidebar contains navigation links such as Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings, Edit Settings, Permissions, Approval Chains, Class Extensions, Add Class Extension, Import Extensions, Export Extensions, Create Site Bundle, Deploy Site Bundle, Web Services, Trash Bin, and Global Settings. The main dialog box has a title 'Add Related Content Set' and sections for 'Page / Location' (with fields for 'Related Content Set Name' and 'Related Locations'), 'Allow users to add only specific subtypes?' (with radio buttons for 'Yes' and 'No'), and 'Select Subtypes' (listing options like Page/Default, Page/Location, Folder/Default, Calendar/Default, Gallery/Default, File/Default, and Link/Default, where 'Page/Location' is selected). A red arrow points to the 'ADD' button at the bottom right of the dialog.

Related Content Sets

The screenshot shows the Mura CMS interface for creating content. The left sidebar includes links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings, Global Settings, Help, and About. The main area is titled "Create Content" with the sub-titles "Update: 05/25/2017 11:56 AM Status: Draft Type: Page". The breadcrumb navigation shows Home > About Us > Locations. The top navigation bar includes Default, Search, and User Steve Withington. The tabs at the top of the content area are BASIC, PUBLISHING, LAYOUT, CATEGORIES, TAGS, RELATED CONTENT (which is highlighted in red), and ADVANCED. Below these tabs, the "Where is the Related Content?" section has two options: "In this site" (selected) and "On another site". Under "In this site", there is a search bar with "headquarters" and an Advanced Search link. The "Search Results (1-1 of 1)" section shows one result: Home > About Us > Locations > Headquarters. The "Related Locations" section is highlighted with a red box and contains the instruction "Add Related Content by dragging items here". A "Limited to:" dropdown is located next to this section. Below it, the "Default" section also has the same "Add Related Content by dragging items here" instruction. At the bottom of the page are buttons for SAVE DRAFT, PREVIEW, SAVE TO CHANGE SET, and PUBLISH (which is highlighted in blue).

Define with XML

CLASS EXTENSIONS

Define with XML

- In addition to using the administrator's UI, Mura can parse a special XML file to create Class Extensions, as well as create custom image sizes, and more.

Define with XML

- In addition to using the administrator's UI, Mura can parse a special XML file to create Class Extensions, as well as create custom image sizes, and more.
- Whenever Mura experiences an application reload, it scans for the “**config.xml.cfm**” file, and if found, will attempt to parse the file to create any pre-defined Class Extensions and custom image sizes.

Define with XML

- In addition to using the administrator's UI, Mura can parse a special XML file to create Class Extensions, as well as create custom image sizes, and more.
- Whenever Mura experiences an application reload, it scans for the “**config.xml.cfm**” file, and if found, will attempt to parse the file to create any pre-defined Class Extensions and custom image sizes.
- If the file is located under a module or plugin directory, Mura obtains other settings and info too.

Define with XML

- The “**config.xml.cfm**” file will be discovered in any site or theme directory, module directory, content type directory, or plugin directory.

Define with XML

- You can nest additional module directories and content type directories within each other, and Mura will automatically search for the “**config.xml.cfm**” file in those too.

config.xml.cfm

- Visit <http://docs.getmura.com/v7-1/mura-developers/mura-beans-objects/custom-objects/class-extensions/define-with-xml/elements-of-the-config-xml-cfm-file/> for details about the file itself.

Export Definitions via UI

CLASS EXTENSIONS

Export Definitions via UI

- Mura is able to export Class Extensions as an XML file via the back-end administration UI.

Export Definitions via UI

- Mura is able to export Class Extensions as an XML file via the back-end administration UI.
- This is the recommended method for creating XML Class Extension definitions, as opposed to writing the XML by hand.

Export Definitions via UI

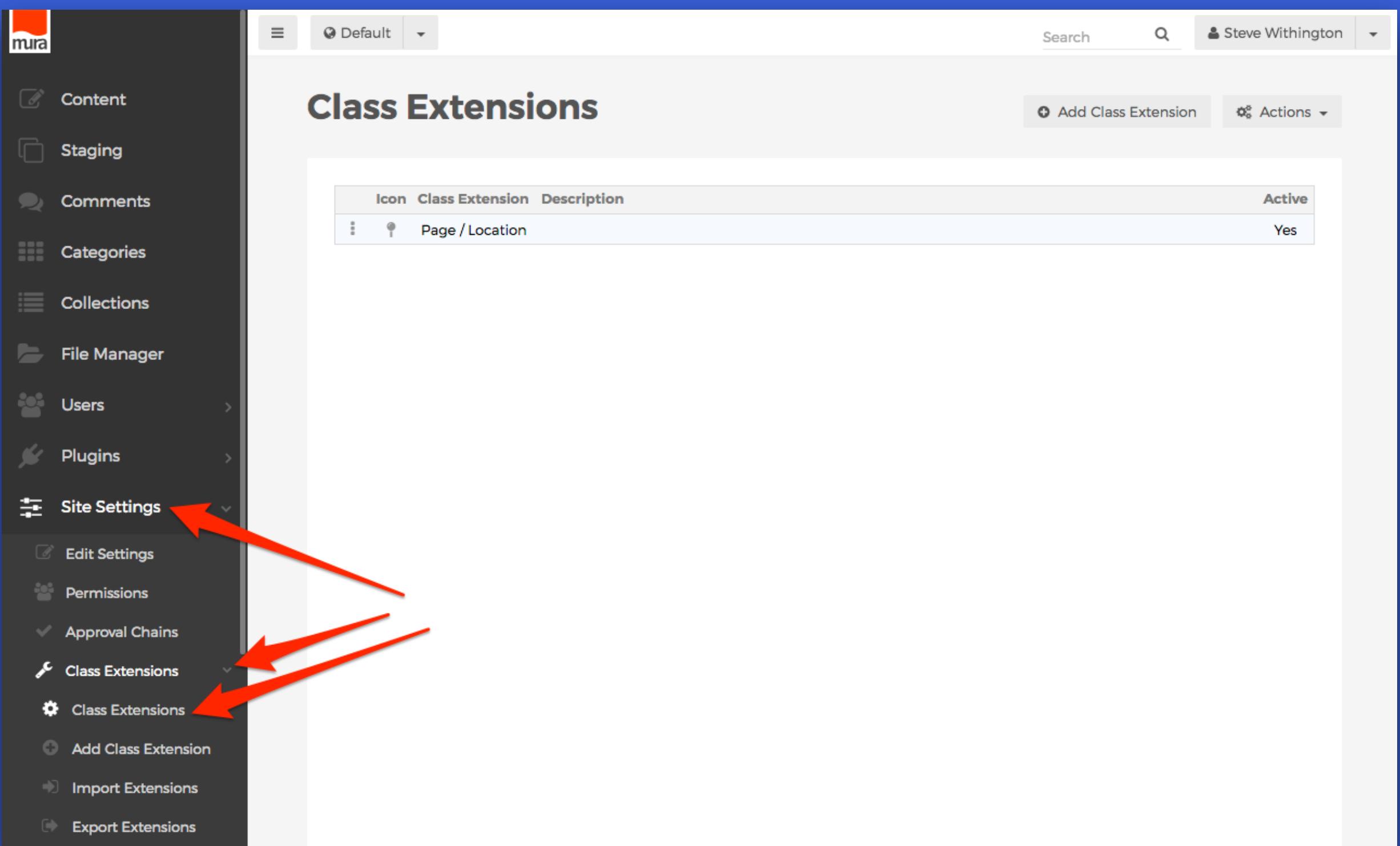
- Mura is able to export Class Extensions as an XML file via the back-end administration UI.
- This is the recommended method for creating XML Class Extension definitions, as opposed to writing the XML by hand.
- By using this method, you're able to reduce the likelihood of typos or introducing errors when Mura attempts to parse your XML file(s).

Export Definitions via UI

- You can choose to export class extensions individually, or you can select multiple class extensions to export as a single XML file.

Export Single Extension

Export Single Extension

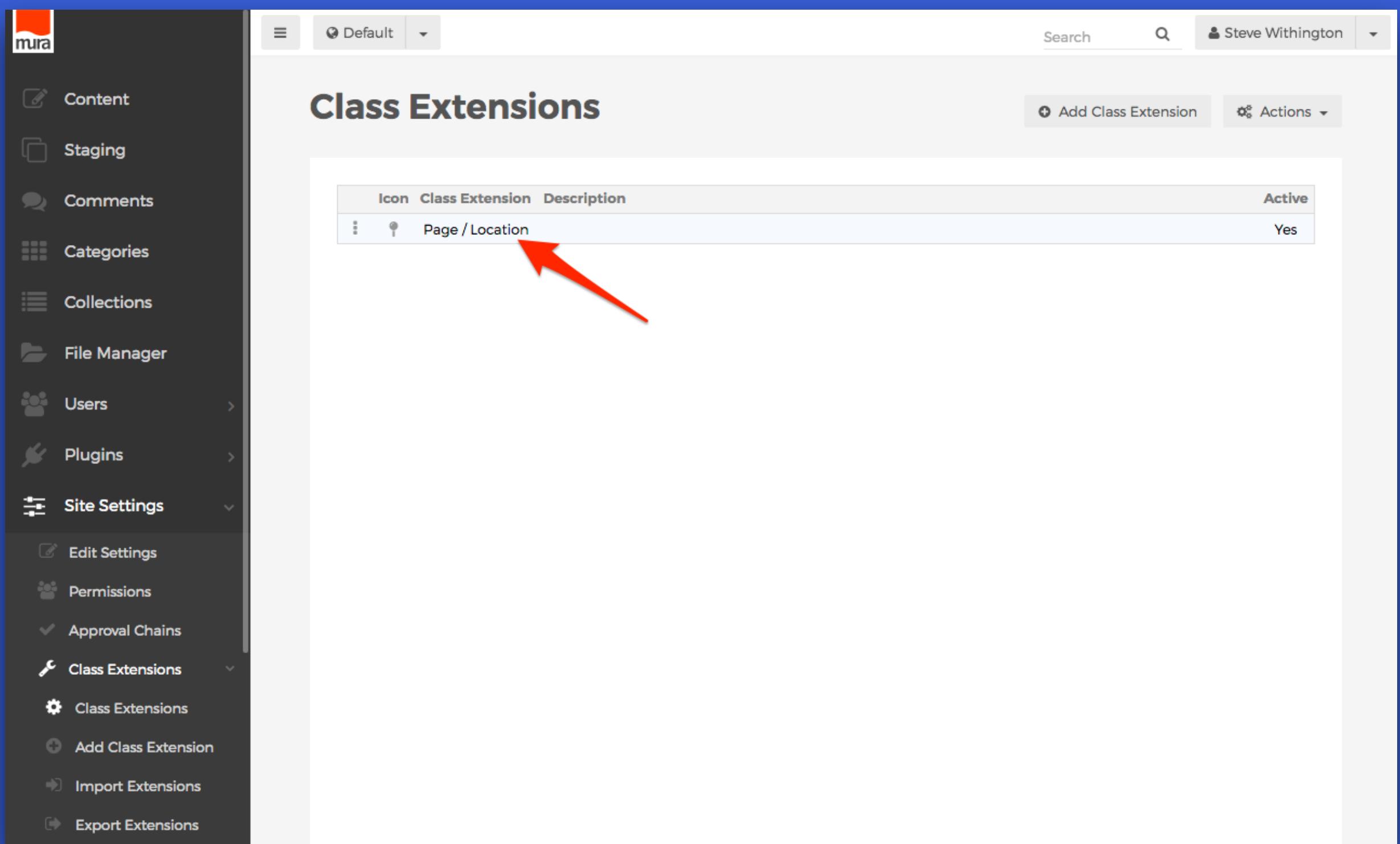


Export Single Extension

The screenshot shows the Mura CMS interface with a dark sidebar and a light main content area. The sidebar contains links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings (with sub-links for Edit Settings, Permissions, Approval Chains, and Class Extensions), Add Class Extension, Import Extensions, and Export Extensions. The main content area has a header with 'Default' and a search bar. Below the header is a table titled 'Class Extensions' with columns for Icon, Class Extension, Description, and Active status. A single row is listed: 'Page / Location' with 'Yes' in the Active column. There are 'Add Class Extension' and 'Actions' buttons at the top right of the table.

| Icon | Class Extension | Description | Active |
|------|-----------------|-------------|--------|
| key | Page / Location | | Yes |

Export Single Extension



The screenshot shows the Mura CMS interface. On the left is a dark sidebar with various navigation items: Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings (with sub-options like Edit Settings, Permissions, Approval Chains, Class Extensions, and Import/Export), Add Class Extension, Import Extensions, and Export Extensions. The 'Class Extensions' item under Site Settings is currently selected. The main content area is titled 'Class Extensions' and displays a table with one row. The table has columns for Icon, Class Extension, Description, and Active status. The single row contains an icon of a location pin, the text 'Page / Location', and the word 'Yes' under 'Active'. A red arrow points to the 'Page / Location' text in the table.

| Icon | Class Extension | Description | Active |
|-------------------|-----------------|-------------|--------|
| Location pin icon | Page / Location | | Yes |

Export Single Extension

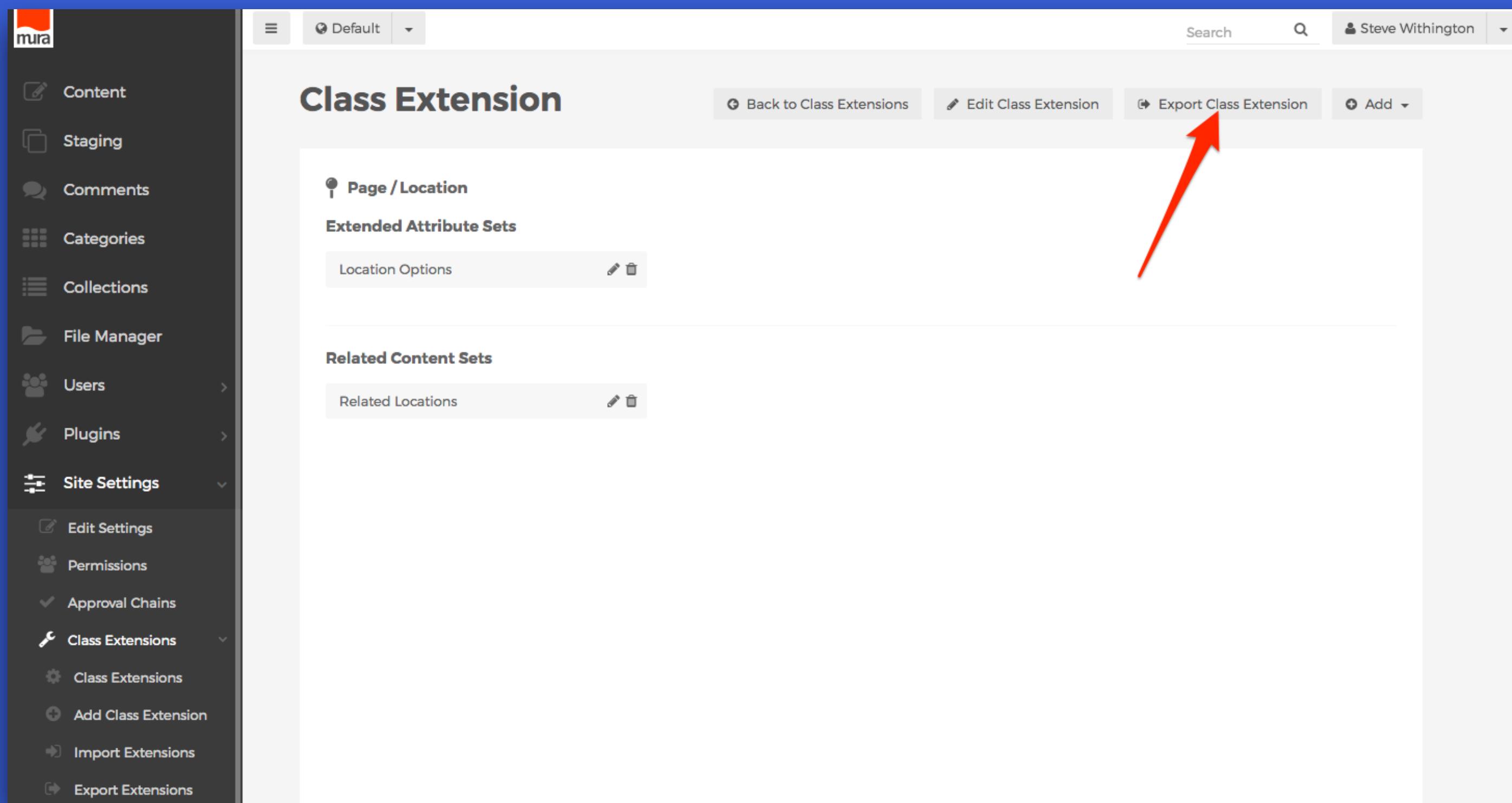
The screenshot shows the Mura CMS interface with a dark sidebar and a light main content area. The sidebar contains the following navigation items:

- Content
- Staging
- Comments
- Categories
- Collections
- File Manager
- Users
- Plugins
- Site Settings
 - Edit Settings
 - Permissions
 - Approval Chains
 - Class Extensions
- Class Extensions
 - Class Extensions
 - Add Class Extension
 - Import Extensions
 - Export Extensions

The main content area has a header "Class Extension" with a "Default" dropdown, a search bar, and a user dropdown for "Steve Withington". Below the header are buttons for "Back to Class Extensions", "Edit Class Extension", "Export Class Extension", and "Add". The main content is divided into sections:

- Page / Location**: Contains a "Location Options" section with edit and delete icons.
- Extended Attribute Sets**: Contains a "Related Locations" section with edit and delete icons.
- Related Content Sets**: Contains a "Related Locations" section with edit and delete icons.

Export Single Extension



Export Single Extension

The screenshot shows the Mura CMS interface with a dark sidebar on the left and a light-colored main content area. The sidebar includes links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, and Site Settings (with sub-links for Edit Settings, Permissions, Approval Chains, Class Extensions, Add Class Extension, Import Extensions, Export Extensions, Create Site Bundle, Deploy Site Bundle, Web Services, and Trash Bin). The main content area has a header with 'Default' and a search bar. Below the header is a title 'Export Class Extensions' with a 'Back to Class Extensions' link. A note says 'Hover over the code, and press 'CTRL/CMD + C' to copy it to your clipboard.' Below this is a large text area containing XML code for a class extension. At the bottom of the main area is a blue 'DOWNLOAD' button.

```
<?xml version="1.0" encoding="UTF-8"?>
<mura>
    <extensions>
        <extension adminonly="0" availablesubtypes="" basekeyfield="contentHistID" basetable="tcontent" datatable="tclassextnddata" description="" hasassocfile="1" hasbody="0" hasconfigurator="0" hassummary="1" iconclass="ml-map-pin" subtype="Location" type="Page">
            <attribute adminonly="0" defaultvalues="" hint="" label="Location Street" message="Please enter a Location Street" name="locationstreet" optionlabellist="" optionlist="" orderno="1" regex="" required="true" type="TextBox" validation="" />
            <attribute adminonly="0" defaultvalues="" hint="" label="Location City" message="Please enter a Location City" name="locationcity" optionlabellist="" optionlist="" orderno="2" regex="" required="true" type="TextBox" validation="" />
            <attribute adminonly="0" defaultvalues="" hint="" label="Location State" message="Please enter a Location State" name="locationstate" optionlabellist="" optionlist="" orderno="3" regex="" required="true" type="TextBox" validation="" />
            <attribute adminonly="0" defaultvalues="" hint="" label="Location Postal Code" message="" name="locationpostalcode" optionlabellist="" optionlist="" orderno="4" regex="" required="false" type="TextBox" validation="" />
        </attribute>
    </extensions>
</mura>
```

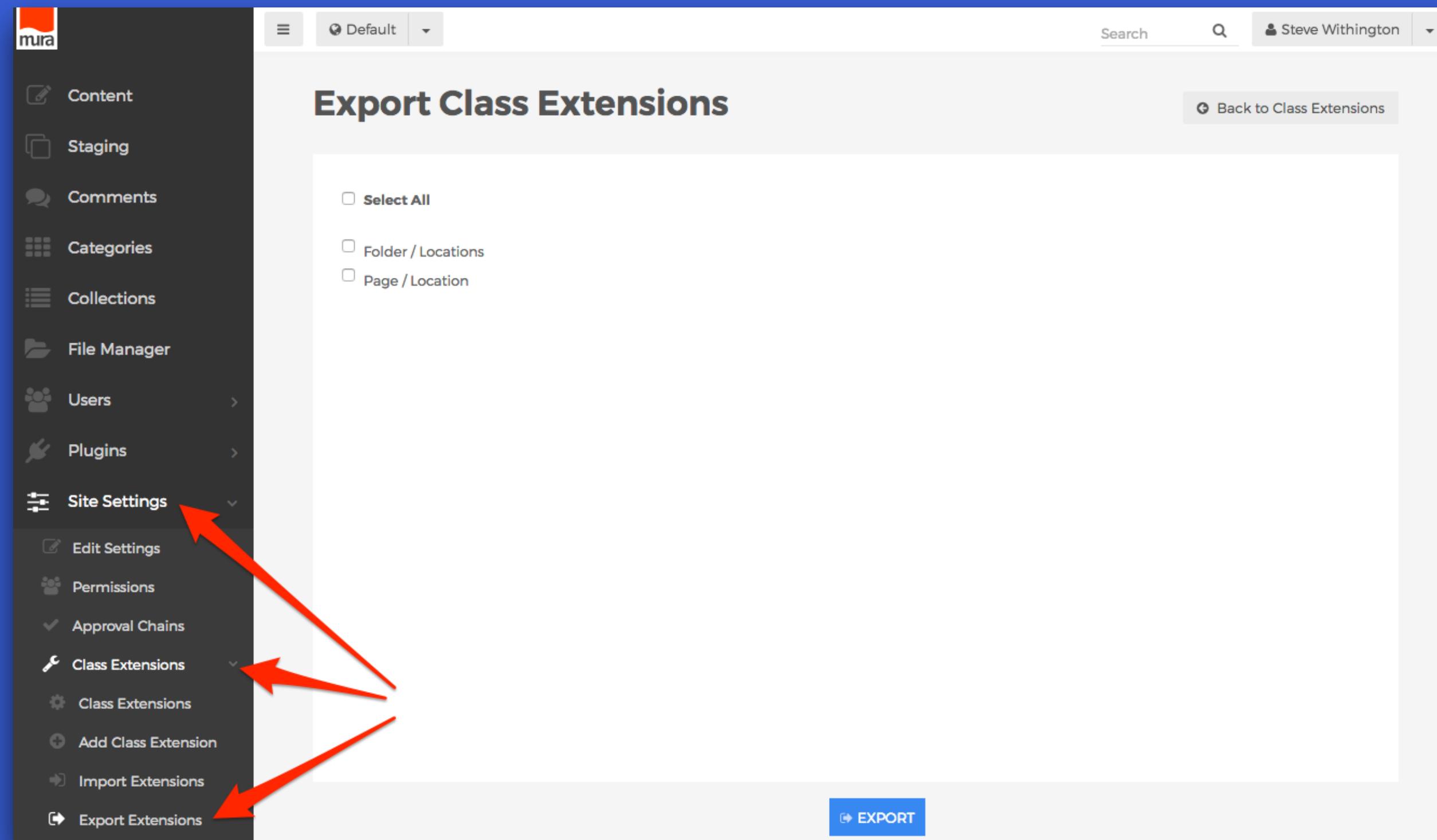
Export Single Extension

The screenshot shows the Mura CMS interface with a dark sidebar on the left and a light-colored main content area. The sidebar includes links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, and Site Settings (with sub-links for Edit Settings, Permissions, Approval Chains, Class Extensions, Add Class Extension, Import Extensions, Export Extensions, Create Site Bundle, Deploy Site Bundle, Web Services, and Trash Bin). The main content area has a header 'Export Class Extensions' with a 'Back to Class Extensions' link. Below the header is a text box containing XML code. A red arrow points to a blue 'DOWNLOAD' button at the bottom of the page.

```
<?xml version="1.0" encoding="UTF-8"?>
<mura>
    <extensions>
        <extension adminonly="0" availablesubtypes="" basekeyfield="contentHistID" basetable="tcontent" datatable="tclassextnddata" description="" hasassocfile="1" hasbody="0" hasconfigurator="0" hassummary="1" iconclass="ml-map-pin" subtype="Location" type="Page">
            <attribute adminonly="0" defaultvalue="" hint="" label="Location Street" message="Please enter a Location Street" name="locationstreet" optionlabellist="" optionlist="" ordeno="1" regex="" required="true" type="TextBox" validation="" />
            <attribute adminonly="0" defaultvalue="" hint="" label="Location City" message="Please enter a Location City" name="locationcity" optionlabellist="" optionlist="" ordeno="2" regex="" required="true" type="TextBox" validation="" />
            <attribute adminonly="0" defaultvalue="" hint="" label="Location State" message="Please enter a Location State" name="locationstate" optionlabellist="" optionlist="" ordeno="3" regex="" required="true" type="TextBox" validation="" />
            <attribute adminonly="0" defaultvalue="" hint="" label="Location Postal Code" message="" name="locationpostalcode" optionlabellist="" optionlist="" ordeno="4" regex="" required="false" type="TextBox" validation="" />
        </attribute>
    </extensions>
</mura>
```

Export Multiple Extensions

Export Multiple Extensions



Export Multiple Extensions

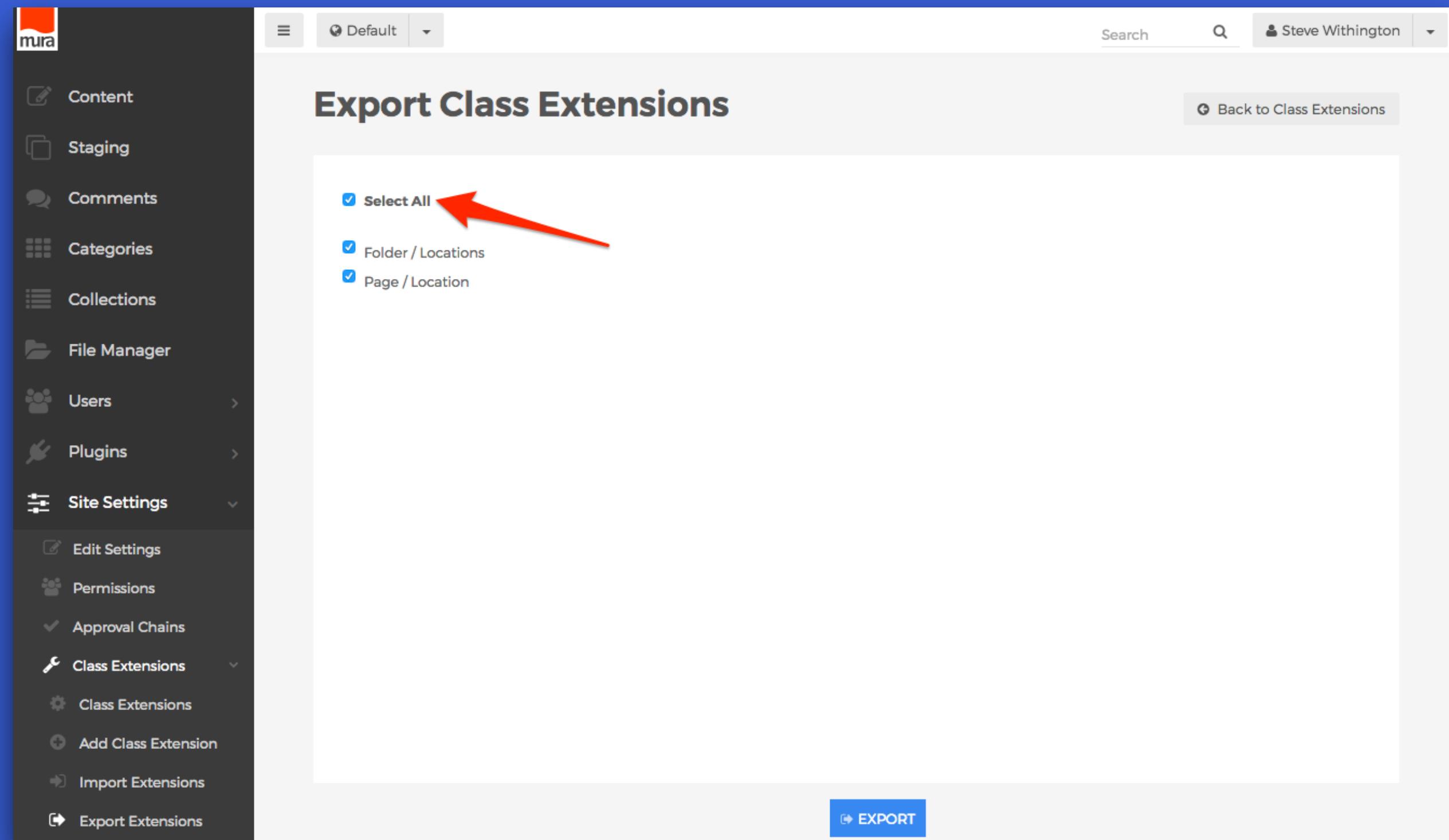
The screenshot shows the Mura CMS interface with the following details:

- Left Sidebar:** Contains links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, and Site Settings (with sub-options like Edit Settings, Permissions, Approval Chains, and Class Extensions).
- Current Page:** "Class Extensions" (under Site Settings).
- Actions Menu:** An open dropdown menu from the "Actions" button in the top right corner. It includes options for "Add Class Extension", "Export", "Import", and "Delete". The "Export" option is highlighted with a red arrow.
- Table:** A list of Class Extensions with columns for Icon, Class Extension, and Description. It shows two entries: "Folder / Locations" and "Page / Location".

Export Multiple Extensions

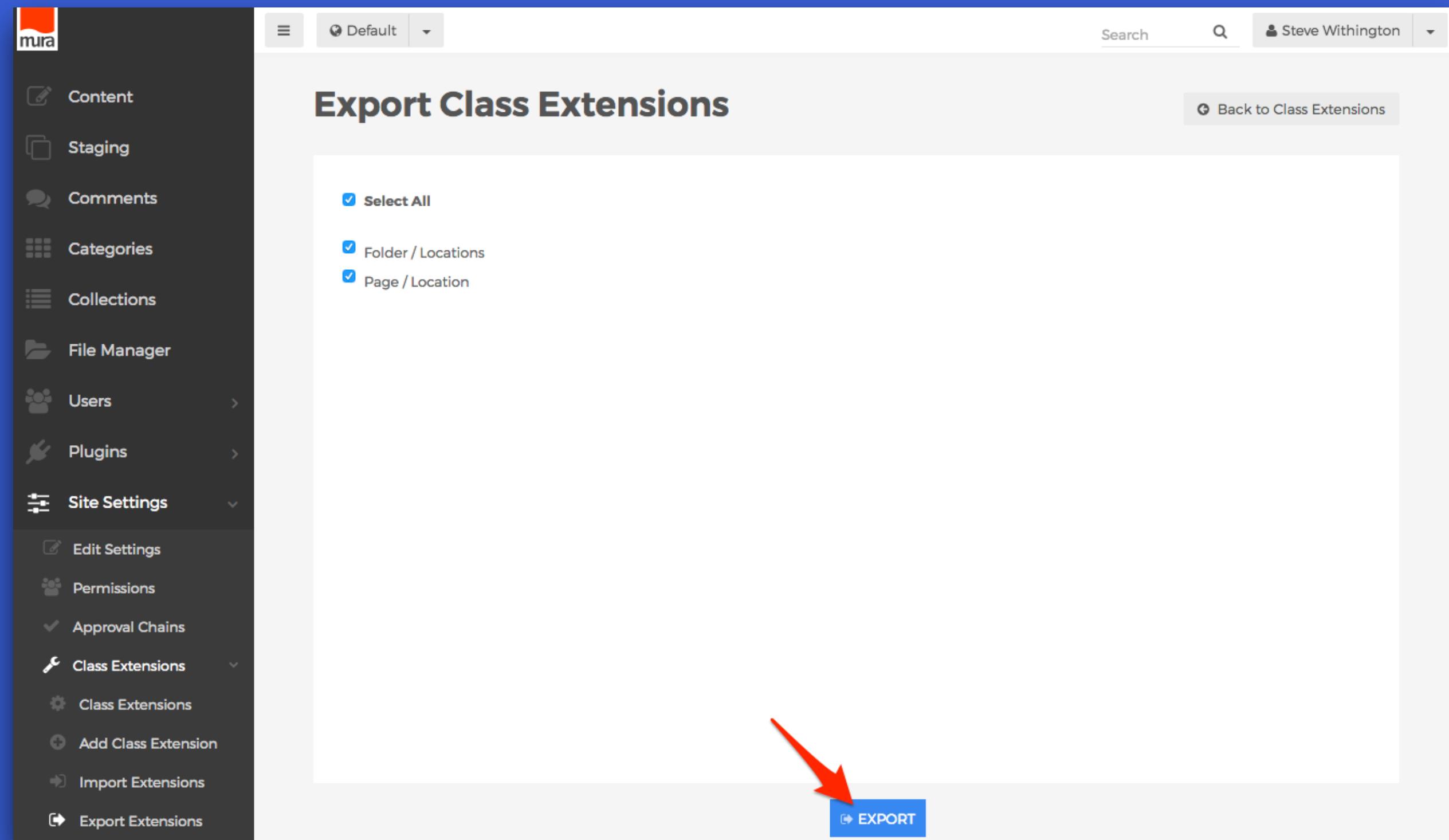
The screenshot shows the Mura CMS interface with a dark sidebar and a light main content area. The sidebar includes links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings (with sub-links for Edit Settings, Permissions, Approval Chains, and Class Extensions), and extensions for Import and Export. The main content area has a title 'Export Class Extensions' and a sub-link 'Back to Class Extensions'. It features a checkbox for 'Select All' and two other checkboxes for 'Folder / Locations' and 'Page / Location'. At the bottom right is a blue 'EXPORT' button.

Export Multiple Extensions



The screenshot shows the 'Export Class Extensions' interface in the Mura CMS. On the left is a dark sidebar with various site management options like Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, and Site Settings. Under Site Settings, 'Class Extensions' is expanded, showing 'Class Extensions', 'Add Class Extension', 'Import Extensions', and 'Export Extensions'. The main panel is titled 'Export Class Extensions' and contains three checkboxes: 'Select All' (which has a red arrow pointing to it), 'Folder / Locations', and 'Page / Location'. At the bottom right is a blue 'EXPORT' button.

Export Multiple Extensions



Export Multiple Extensions

The screenshot shows the Mura CMS interface with a dark sidebar on the left and a light-colored main content area. The sidebar contains links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings (with sub-links for Edit Settings, Permissions, Approval Chains, and Class Extensions), Add Class Extension, Import Extensions, and Export Extensions. The main content area has a header 'Default' and a search bar. Below the header is a title 'Export Class Extensions' with a 'Back to Class Extensions' link. A note says 'Hover over the code, and press 'CTRL/CMD + C' to copy it to your clipboard.' Below this is a large text area containing XML code for class extensions. At the bottom of the main area is a blue 'DOWNLOAD' button.

```
<?xml version="1.0" encoding="UTF-8"?>
<mura>
    <extensions>
        <extension adminonly="0" availablesubtypes="Page/Location" basekeyfield="contentHistID" basetable="tcontent"
            datatable="tclassextenddata" description="" hasassocfile="0" hasbody="0" hasconfigurator="0" hassummary="0" iconclass="mi-map-o"
            subtype="Locations" type="Folder"/>
        <extension adminonly="0" availablesubtypes="" basekeyfield="contentHistID" basetable="tcontent" datatable="tclassextenddata"
            description="" hasassocfile="1" hasbody="0" hasconfigurator="0" hassummary="1" iconclass="mi-map-pin" subtype="Location" type="Page">
            <attributeset categoryid="" container="Basic" name="Location Options" orderno="1">
                <attribute adminonly="0" defaultvalues="" hint="" label="Location Street" message="Please enter a Location Street"
                    name="locationstreet" optionlabellist="" optionlist="" orderno="1" regex="" required="true" type="TextBox" validation="" />
                <attribute adminonly="0" defaultvalues="" hint="" label="Location City" message="Please enter a Location City"
                    name="locationcity" optionlabellist="" optionlist="" orderno="2" regex="" required="true" type="TextBox" validation="" />
                <attribute adminonly="0" defaultvalue="" hint="" label="Location State" message="Please enter a Location State"
                    name="locationstate" optionlabellist="" optionlist="" orderno="3" regex="" required="true" type="TextBox" validation="" />
                <attribute adminonly="0" defaultvalue="" hint="" label="Location Postal Code" message="" name="locationpostalcode"
                    optionlabellist="" optionlist="" orderno="4" regex="" required="false" type="TextBox" validation="" />
            </attributeset>
        </extension>
    </extensions>
```

Export Multiple Extensions

The screenshot shows the Mura CMS interface with a dark sidebar on the left containing various site management options like Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings, Edit Settings, Permissions, Approval Chains, and Class Extensions. The Class Extensions option is expanded, showing sub-options: Add Class Extension, Import Extensions, and Export Extensions. The main content area is titled 'Export Class Extensions'. It displays an XML code block with instructions to copy it to clipboard by hovering and pressing 'CTRL/CMD + C'. A red arrow points to a blue 'DOWNLOAD' button at the bottom of the XML code area.

```
<?xml version="1.0" encoding="UTF-8"?>
<mura>
    <extensions>
        <extension adminonly="0" availablesubtypes="Page/Location" basekeyfield="contentHistID" basetable="tcontent"
            datatable="tclassextenddata" description="" hasassocfile="0" hasbody="0" hasconfigurator="0" hassummary="0" iconclass="mi-map-o"
            subtype="Locations" type="Folder"/>
        <extension adminonly="0" availablesubtypes="" basekeyfield="contentHistID" basetable="tcontent" datatable="tclassextenddata"
            description="" hasassocfile="1" hasbody="0" hasconfigurator="0" hassummary="1" iconclass="mi-map-pin" subtype="Location" type="Page">
            <attributeset categoryid="" container="Basic" name="Location Options" orderno="1">
                <attribute adminonly="0" defaultvalues="" hint="" label="Location Street" message="Please enter a Location Street"
                    name="locationstreet" optionlabellist="" optionlist="" orderno="1" regex="" required="true" type="TextBox" validation="" />
                <attribute adminonly="0" defaultvalues="" hint="" label="Location City" message="Please enter a Location City"
                    name="locationcity" optionlabellist="" optionlist="" orderno="2" regex="" required="true" type="TextBox" validation="" />
                <attribute adminonly="0" defaultvalue="" hint="" label="Location State" message="Please enter a Location State"
                    name="locationstate" optionlabellist="" optionlist="" orderno="3" regex="" required="true" type="TextBox" validation="" />
                <attribute adminonly="0" defaultvalue="" hint="" label="Location Postal Code" message="" name="locationpostalcode"
                    optionlabellist="" optionlist="" orderno="4" regex="" required="false" type="TextBox" validation="" />
            </attributeset>
        </extension>
    </extensions>
```

Download

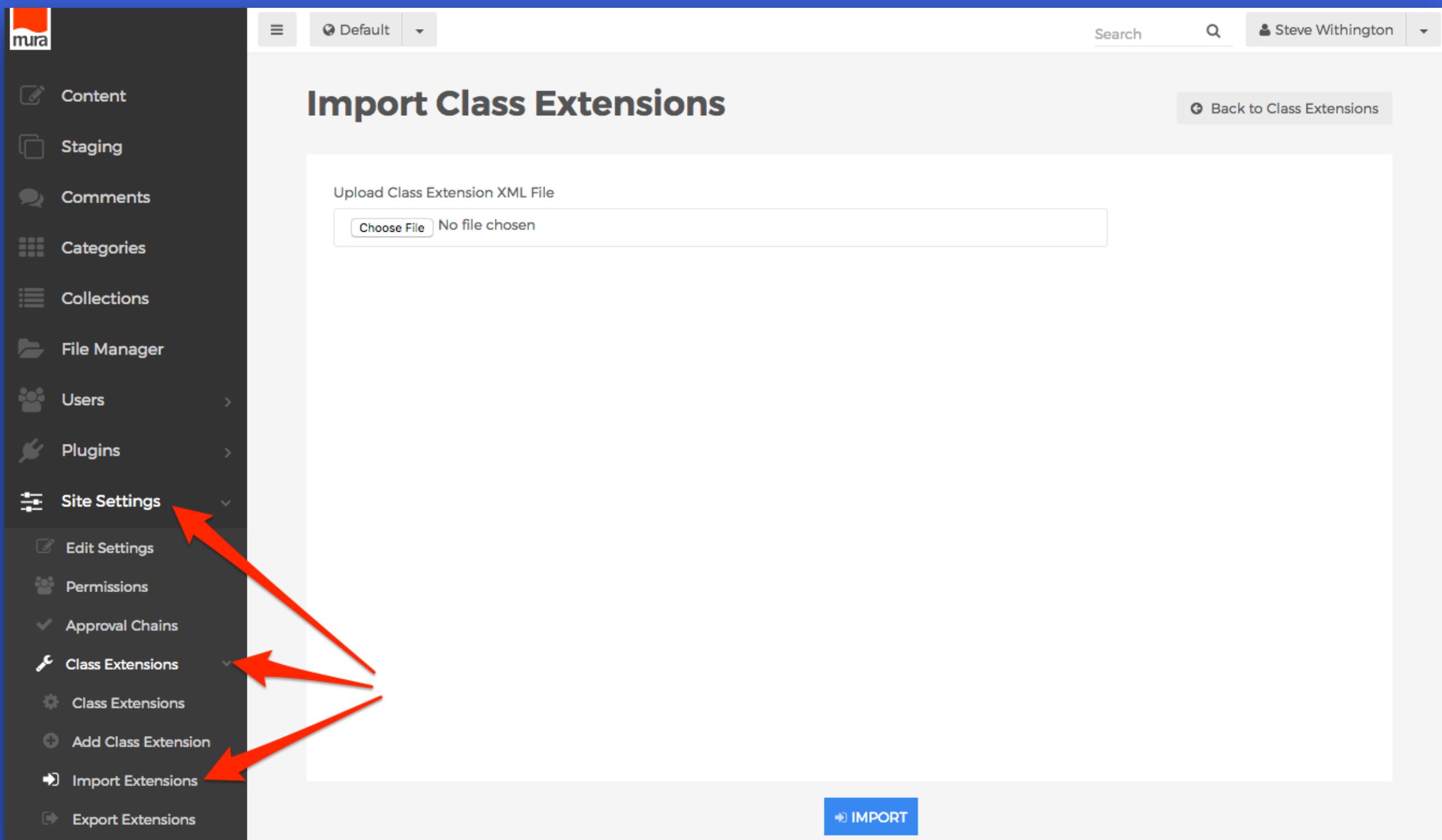
Import Definitions via UI

CLASS EXTENSIONS

Import Definitions via UI

- To import Class Extensions, you will first need a properly formatted XML file, preferably exported via Mura's admin UI.

Import Definitions via UI



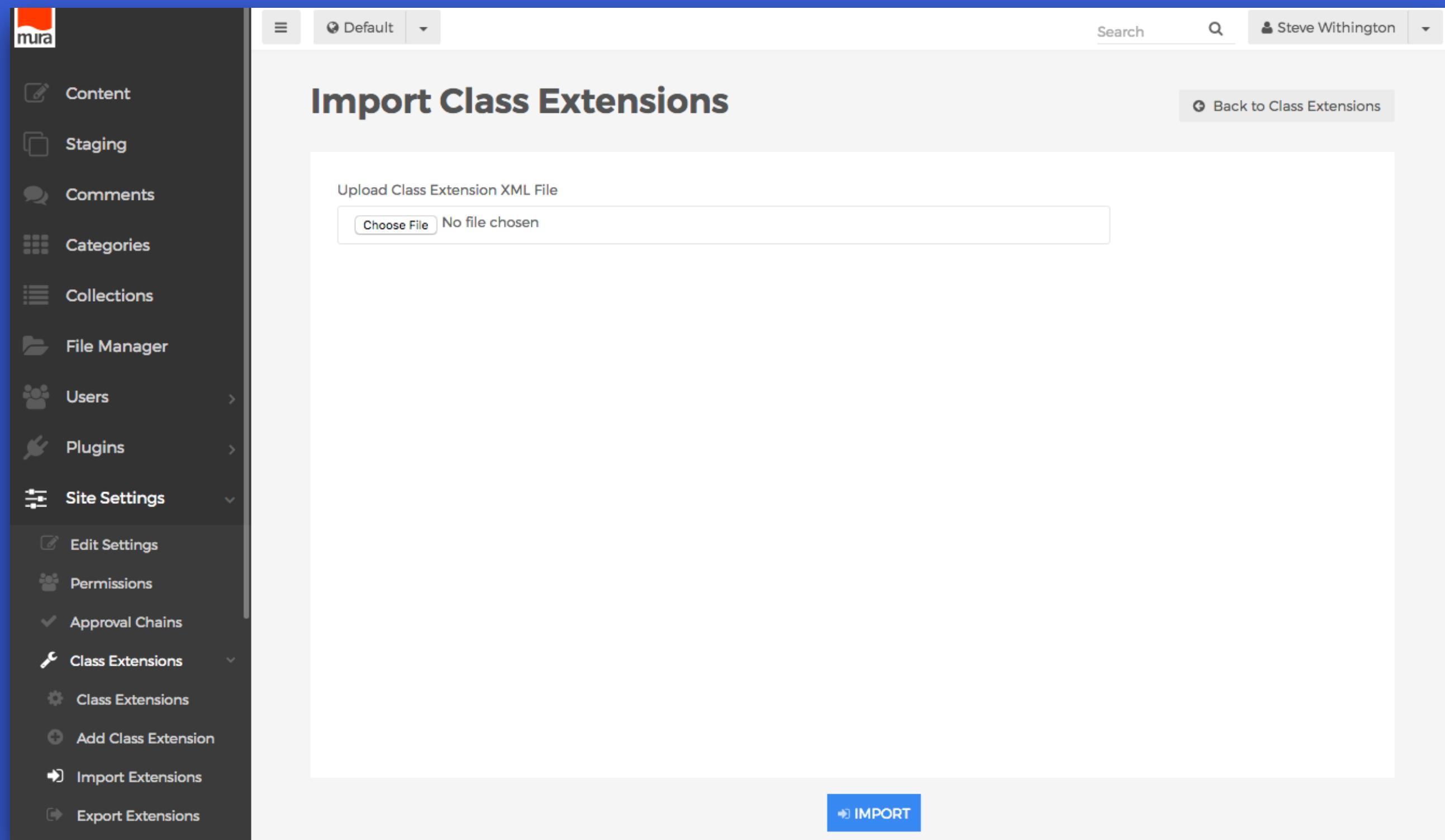
Import Definitions via UI

The screenshot shows the Mura CMS interface for managing Class Extensions. On the left is a dark sidebar with various site management options. The main area is titled "Class Extensions" and displays a table with two rows:

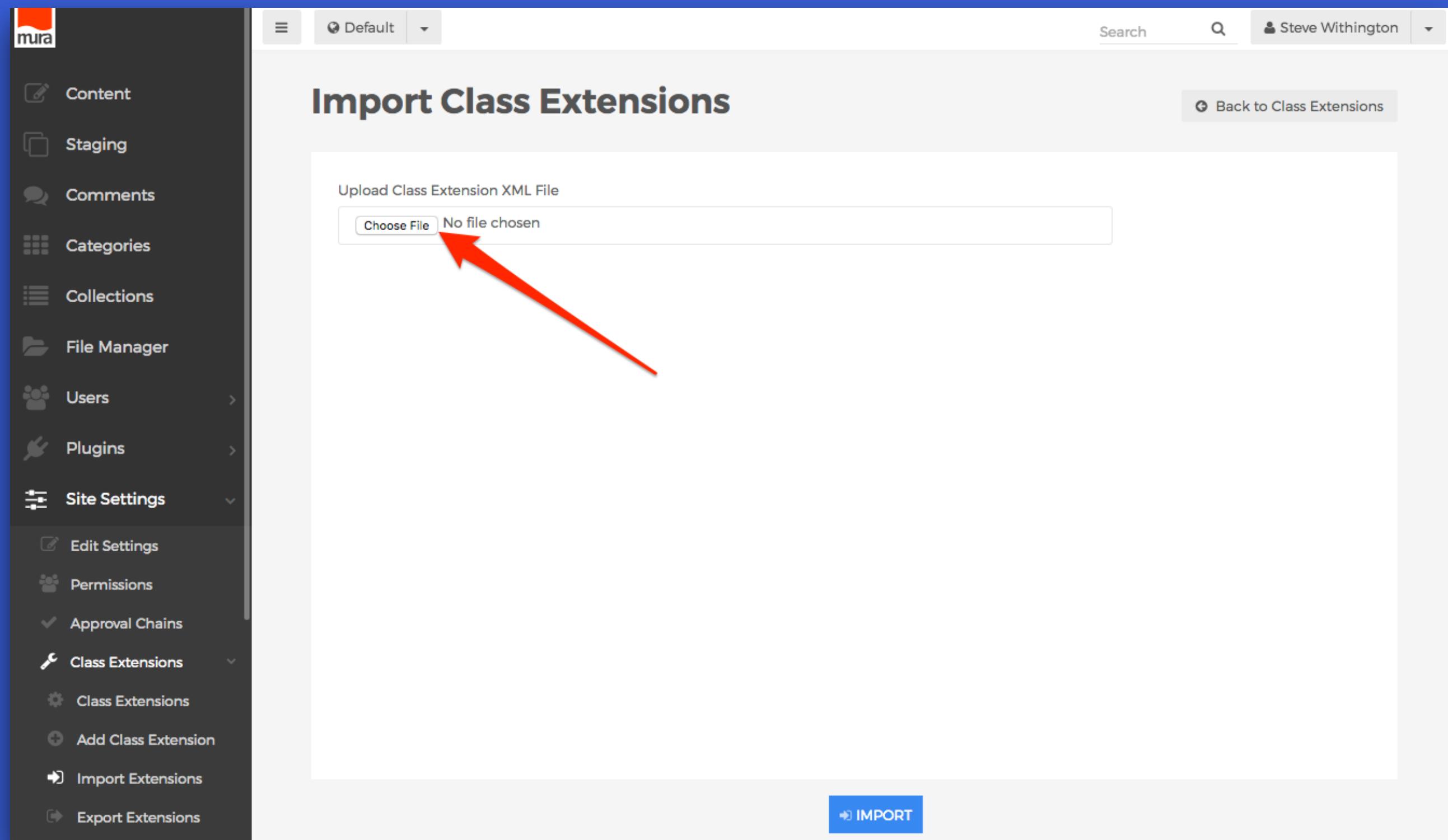
| Icon | Class Extension | Description |
|-------------|--------------------|-------------|
| Folder icon | Folder / Locations | |
| Page icon | Page / Location | |

In the top right corner, there is a dropdown menu labeled "Actions" with a submenu open. The submenu includes "Export" and "Import". A red arrow points from the text "Import Definitions via UI" in the image description to the "Import" option in the dropdown menu.

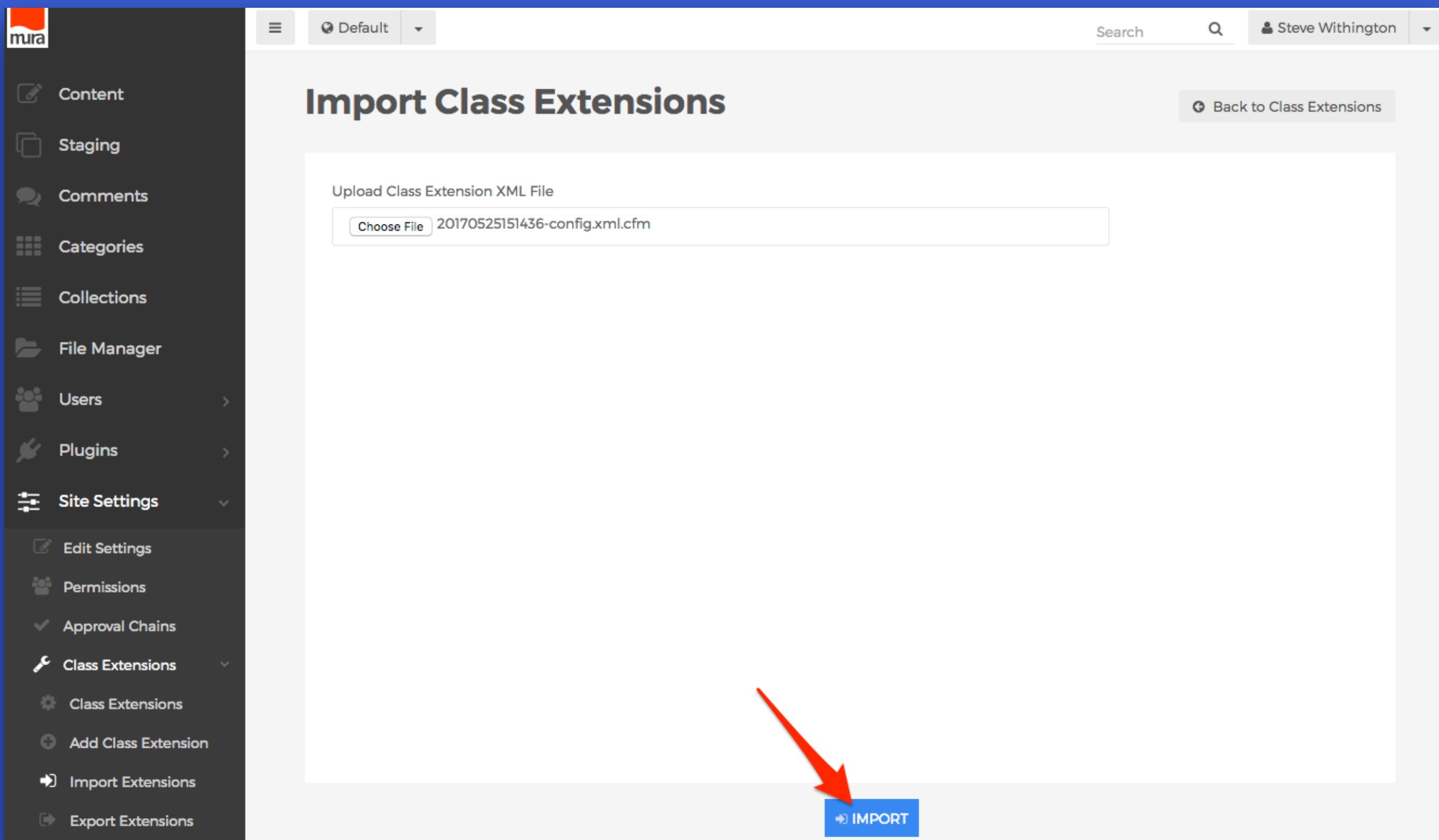
Import Definitions via UI



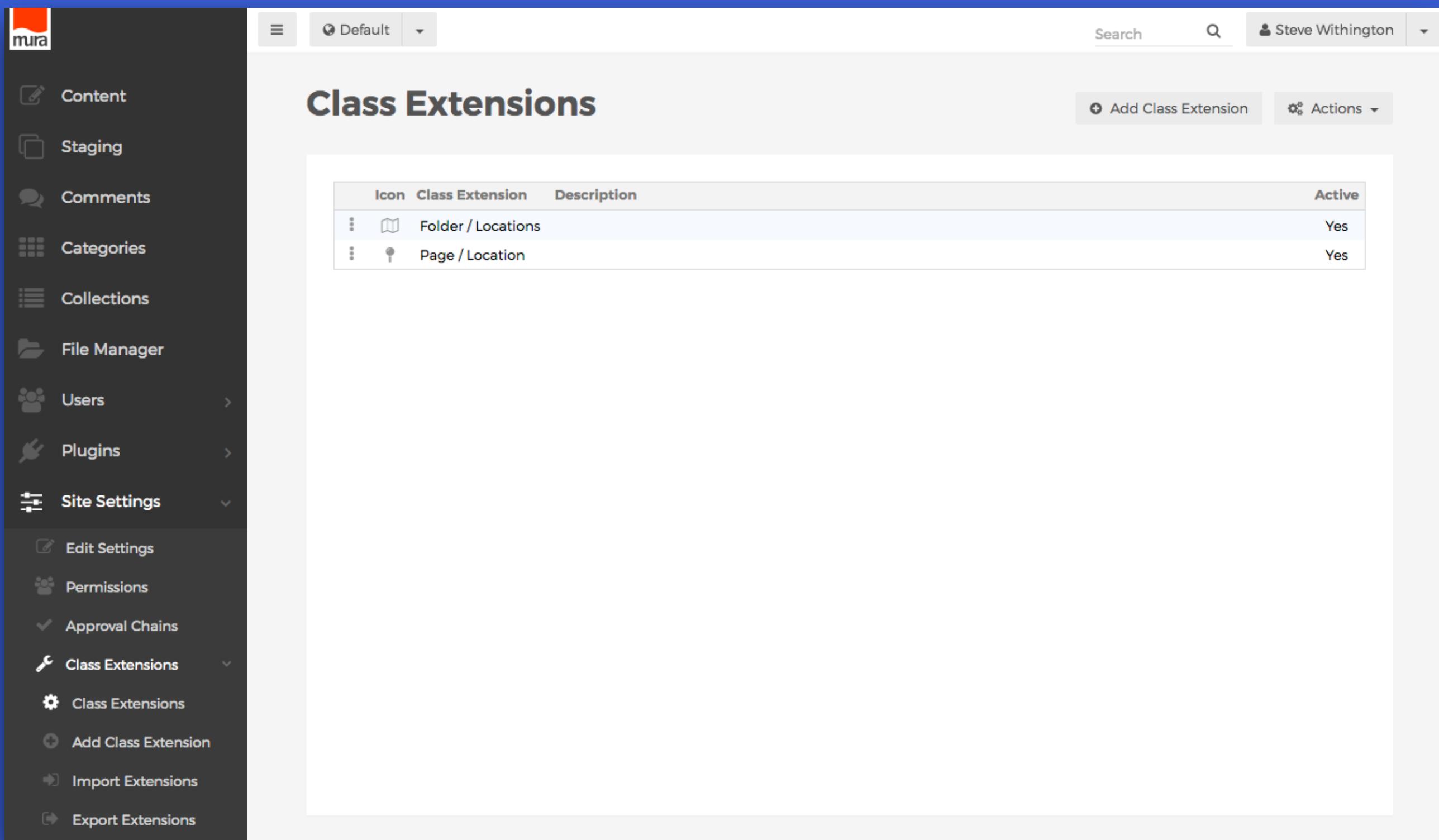
Import Definitions via UI



Import Definitions via UI



Import Definitions via UI



The screenshot shows the Mura CMS interface with the 'Class Extensions' page open. The left sidebar contains navigation links such as Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, and Site Settings. Under Site Settings, there are options for Edit Settings, Permissions, Approval Chains, Class Extensions, Add Class Extension, Import Extensions, and Export Extensions. The main content area is titled 'Class Extensions' and displays a table with two rows:

| Icon | Class Extension | Description | Active |
|-------------|--------------------|-------------|--------|
| Folder icon | Folder / Locations | | Yes |
| Page icon | Page / Location | | Yes |

Displaying Attributes

CLASS EXTENSIONS

Displaying Attributes

- There are a few ways to display attributes of Class Extensions, and it really depends on where you're attempting to display them.

Displaying Attributes

- There are a few ways to display attributes of Class Extensions, and it really depends on where you're attempting to display them.
- The most important thing to keep in mind is that you'll need to know your attribute's name property, in order to display it.

Displaying Attributes

- In short, a Class Extension's attribute is treated as a property of the bean/object it's associated with.

Displaying Attributes

- In short, a Class Extension's attribute is treated as a property of the bean/object it's associated with.
- For example, if a Class Extension's “**Base Type**” is “**Content**”, then you could use the same getter and setter methods when working with any other attribute of the Content Bean.

Basic Syntax

1. `someBean.get('attributeName')`
2. `someBean.get{AttributeName}()`
3. `someBean.getValue('attributeName')`

Via Layout Template

```
1. <cfoutput>
2.   <h2>
3.     #esapiEncode('html', m.content('attributeName'))#
4.   </h2>
5. </cfoutput>
```

Via HTML Editor

1.

```
[m]esapiEncode('html', m.content('attributeName')) [/m]
```

Via Content Bean

```
1. contentBean = m.getBean('content').loadBy(title='Home');  
2. // assumes we have a custom attribute labeled `locationstreet`  
3. street = contentBean.get('locationstreet');
```

Via Iterator

```
1.  <!-- get an iterator of the content item's children -->
2.  <cfset it = m.content().getKidsIterator()>
3.  <ul>
4.  <cfloop condition="it.hasNext()">
5.      <cfset item = it.next()>
6.      <li>
7.          #esapiEncode('html', item.getAttributeName())#
8.      </li>
9.  </cfloop>
10. </ul>
```

Making Attributes Editable

CLASS EXTENSIONS

Making Attributes Editable

- When displaying an extended attribute in a layout template, you may want to make the attribute editable by content managers when they're using front-end editing features.

Making Attributes Editable

- When displaying an extended attribute in a layout template, you may want to make the attribute editable by content managers when they're using front-end editing features.
- This is accomplished by using
“m.renderEditableAttribute”.

Making Attributes Editable

- When displaying an extended attribute in a layout template, you may want to make the attribute editable by content managers when they're using front-end editing features.
- This is accomplished by using **“m.renderEditableAttribute”**.
- **Note:** This feature only works on extended attributes with its **Input Type** set to “**TextBox**” or “**HTMLEditor**”.

Function Syntax

```
1. m.renderEditableAttribute(  
2.   attribute  
3.   , type  
4.   , required  
5.   , validation  
6.   , message  
7.   , label  
8.   , value  
9.   , enableMuraTag  
10. )
```

Example

```
1. <cfoutput>
2.   <div>
3.     #m.renderEditableAttribute(
4.       attribute='attributeName'
5.       type='HTMLEditor'
6.       label='Attribute Label'
7.     )#
8.   </div>
9. </cfoutput>
```

Mura ORM

OBJECT-RELATIONAL MAPPING (ORM)

Mura ORM

- Before we proceed through this section, it's important to have at least a basic understanding of object-oriented programming (OOP) and its principles.

Mura ORM

- Before we proceed through this section, it's important to have at least a basic understanding of object-oriented programming (OOP) and its principles.
- If you're unfamiliar with OOP, there's a plethora of resources, available both online and offline.

Mura ORM

- Before we proceed through this section, it's important to have at least a basic understanding of object-oriented programming (OOP) and its principles.
- If you're unfamiliar with OOP, there's a plethora of resources, available both online and offline.
- Additionally, having a solid understanding of relational database management systems (RDBMSs) is extremely helpful.

Object-Relational Mapping

- Since relational database management systems (RDBMSs) don't store objects directly, object-relational mapping (ORM), attempts to bridge the word to object-oriented programming (OOP).

Object-Relational Mapping

- According to Wikipedia, ORM addresses the main issue for OOP developers who work with RDBMSs, by “translating the logical representation of objects into an atomized form capable of being stored in a database, while preserving the properties of objects and their relationships so they can be reloaded as objects when needed.”

Object-Relational Mapping

- According to Wikipedia, ORM addresses the main issue for OOP developers who work with RDBMSs, by “translating the logical representation of objects into an atomized form capable of being stored in a database, while preserving the properties of objects and their relationships so they can be reloaded as objects when needed.”
- In other words, ORM essentially creates a “virtual object database” that can be used by programmers.

What is Mura ORM?

What is Mura ORM?

- Mura ORM is essentially a “virtual object database” that can be used by Mura developers.

What is Mura ORM?

- Mura ORM is essentially a “virtual object database” that can be used by Mura developers.
- It allows developers the ability to create their own objects, and work with them in the same ways Mura developers can work with Mura’s bean objects.

What is Mura ORM?

- Mura ORM is essentially a “virtual object database” that can be used by Mura developers.
- It allows developers the ability to create their own objects, and work with them in the same ways Mura developers can work with Mura’s bean objects.
- This means developers don’t have to worry about what kind of database Mura is running on, or write a ton of SQL to perform many of the most mundane tasks such as CRUD statements, etc.

What is Mura ORM?

- While comparable in nature to ColdFusion ORM, Mura ORM is not exactly the same.

What is Mura ORM?

- While comparable in nature to ColdFusion ORM, Mura ORM is not exactly the same.
- However, if you've worked with ColdFusion ORM before, you'll definitely find many similarities with Mura ORM.

Mura ORM Configuration

Mura ORM Configuration

- Mura employs a “convention over configuration” design paradigm for registering ORM objects/entities.

Mura ORM Configuration

- Mura employs a “convention over configuration” design paradigm for registering ORM objects/entities.
- So, instead of creating configuration files to describe the details of where your objects/entities reside, and how to instantiate them, Mura uses a convention-based approach.

Mura ORM Configuration

- Mura employs a “convention over configuration” design paradigm for registering ORM objects/entities.
- So, instead of creating configuration files to describe the details of where your objects/entities reside, and how to instantiate them, Mura uses a convention-based approach.
- In other words, by placing your objects/entities in specifically named directories, you don’t have to manually or explicitly register your entities.

Model Directory

- Under the hood, Mura simply leverages Di/1, a simple convention-based dependency injection (inversion of control) framework.

Model Directory

- As such, any .CFC discovered under any “**../model**” directories, will automatically get registered with Di/1.

Model Directory

- You may also register any directory you want by using
“`m.globalConfig().registerModelDir({dir})`”.

Model Directory

- You may also register any directory you want by using
“m.globalConfig().registerModelDir({dir})”.
- The “**dir**” parameter is a logical path to a CFML directory.

Model Directory

- You may also register any directory you want by using “`m.globalConfig().registerModelDir({dir})`”.
- The “**dir**” parameter is a logical path to a CFML directory.
- This is typically done in the **onApplicationLoad** event.

Model Directory

- You may also register any directory you want by using “`m.globalConfig().registerModelDir({dir})`”.
- The “`dir`” parameter is a logical path to a CFML directory.
- This is typically done in the `onApplicationLoad` event.
- This is useful for plugins, since Mura does not automatically scan for a “model” directory in plugins, since some plugin developers may be using their own bean factory, and do not need Mura to auto-register them.

Model Directory

```
1. public any function onApplicationLoad(m) {  
2.     // This is how you could register a 'model' directory in a plugin  
3.     arguments.m.globalConfig().registerModelDir('/pluginName/path/to/your/model/');  
4. }
```

Model Directory

- **Note:** All registered objects/entities are essentially “global” in the sense that they’ll all use the same underlying database tables. So, you’ll want to avoid describing the same entities in multiple sites.

The “beans” Directory

MURA ORM CONFIGURATION

The “beans” Directory

- Any .CFC's discovered under “**.../model/beans/**“ will be registered as “transient” objects.

The “beans” Directory

- Any .CFC's discovered under “`.../model/beans/`“ will be registered as “transient” objects.
 - **Transient** objects exist only for a request and then are discarded. In other words, you get a new copy every time.

The “beans” Directory

- Any .CFC's discovered under “**.../model/beans/**” will be registered as “transient” objects.
 - **Transient** objects exist only for a request and then are discarded. In other words, you get a new copy every time.
 - **Singleton** objects are shared among all threads and requests. In other words, there's only one of them, and you're not getting a new copy every time. These are great for “service” type objects.

The “beans” Directory

- The examples below show you how you could register multiple transient objects.
 - `../model/beans/person.cfc`
 - `../model/beans/personaddress.cfc`
 - `../model/beans/personphonenumbers.cfc`

The “handlers” Directory

MURA ORM CONFIGURATION

The “handlers” Directory

- Any .CFC's discovered under a “`../handlers/`“ directory within a “**model**” directory, will be registered as custom event handlers.

The “handlers” Directory

- The examples below show how you could register multiple custom event handlers, in various locations.
 - `../model/handlers/myhandler.cfc`
 - `../model/services/handlers/myhandler.cfc`

Registration

MURA ORM CONFIGURATION

Registration

- In order for Mura to discover these directories, and any subsequent changes you make to the directories or files contained within them, you'll need to reload Mura.

Registration

- In order for Mura to discover these directories, and any subsequent changes you make to the directories or files contained within them, you'll need to reload Mura.
- Also, the first time you reload, and anytime you add new properties to your entities, you'll need to append “**&applydbupdates**” to the URL in order for Mura to parse your objects/entities and make the appropriate changes to the database schema.

Mura ORM Entity

Mura ORM Entity

- In a nutshell, a Mura ORM entity is described by creating a ColdFusion component (.CFC).

Mura ORM Entity

- Mura leverages some of the standard attributes of a Mura ORM component, and also utilizes some custom attributes.
 - bundleable
 - cachename
 - datasource
 - dbtype
 - discriminatorcolumn
 - discriminatorvalue
 - **extends***
(mura.bean.beanORM)
 - **entityname***
 - manageschema
 - orderby
 - public
 - readonly
 - **table***
 - usetrash

Mura ORM Entity

```
1. component
2.   extends="mura.bean.beanORM"
3.   entityname="something"
4.   table="somethings"
5.   bundleable="true"
6.   displayname="SomethingBean"
7.   public=true
8.   orderby="{someproperty}" {
9.
10.   // Properties & Methods
11.
12. }
```

Mura ORM Component Props

- To describe the various properties/attributes of Mura ORM entities, use CFML properties. The following lists the available attributes for Mura ORM component properties.

- cascade
- cfc
- comparable
- datatype
- default
- fieldtype
- fkcolumn
- length
- loadkey
- message
- **name***
- nullable
- orderby
- persistant
- required
- singularname
- validate

Mura ORM Component Props

- If a property has a “**validate**” attribute, you may include additional attributes for the property validation.

- `regex`
- `minvalue`
- `maxvalue`
- `minlength`
- `maxlength`
- `inlist`
- `method`
- `lte`
- `lt`
- `gte`
- `gt`
- `eq`
- `neq`

Mura ORM Component Props

- The example below illustrates how to use a validation attribute, when a property has a “**validate**” attribute.

```
1. | property name="year" datatype="int" validate="numeric" minvalue="1900" maxvalue="2017";
```

Related Entity Methods

- If the property is a related entity, and the “**fieldtype**” is either “**one-to-many**” or “**many-to-many**”, the following methods are automatically available to the entity itself.

| Method | Description |
|---------------------------|--|
| get{PropertyName}Iterator | Returns an iterator of related entities. For example: <code>entity.getAddressesIterator()</code> |
| get{PropertyName} | Under the hood, returns <code>get{PropertyName}Iterator</code> . |
| get{PropertyName}Query | Returns a recordset/query of the related entities. For example: <code>entity.getAddressesQuery()</code> |
| has{PropertyName} | Returns a recordcount of related entities. Useful for if/else statements. |
| add{PropertyName} | Pass in the object/entity to save as a related entity, and Mura will automatically save the related entity when executing the call to <code>entity.save()</code> . Useful for adding multiple related entities. For example: <code>entity.addPhoneNumber(phoneNumberObject)</code> |
| remove{PropertyName} | Pass in the object/entity to be deleted, and Mura will automatically delete the related entity when executing the call to <code>entity.save()</code> . For example: <code>entity.removePhoneNumber(phoneNumberObject)</code> |

Related Entity Methods

- If the property is a related entity, and the “**fieldtype**” is either “**one-to-one**” or “**many-to-one**”, the following methods are automatically available to the entity itself.

| Method | Description |
|----------------------|--|
| get(PropertyName) | Returns the related entity. For example: <code>entity.getEmergencyContact()</code> |
| add(PropertyName) | Pass in the object/entity to save as a related entity, and Mura will automatically save the related entity when executing the call to <code>entity.save()</code> . For example: <code>entity.addEmergencyContact(emergencyContactObject)</code> |
| remove(PropertyName) | Pass in the object/entity to be deleted, and Mura will automatically delete the related entity when executing the call to <code>entity.save()</code> . For example: <code>entity.removeEmergencyContact(emergencyContactObject)</code> |

Mura ORM .CFC w/Properties

```
1. component
2.     extends="mura.bean.beanORM"
3.     entityname="something"
4.     table="somethings"
5.     bundleable="true"
6.     displayname="SomethingBean"
7.     public=true
8.     orderby="namefirst,namelast" {
9.
10.    // primary key
11.    property name="somethingid" fieldtype="id";
12.
13.    // attributes
14.    property name="namefirst" datatype="varchar" length="255" required=true;
15.    property name="namelast" datatype="varchar" length="255" required=true;
16.
17.    // methods could go here
18. }
```

Working With Entities

```
1. // Loading entities
2. entity = m.getBean('entityName')
   .loadBy(someAttribute='Something');
3.
4.
5. // Getting attributes
6. entity.get('attributeName');
7.
8. // Setting attributes
9. entity.set('attributeName', 'Some Value')
10.
11. // Deleting entities
12. entity.delete();
13.
14. // Saving entities
15. entity.save();
16.
17. // Get a feed of entites
18. m.getFeed('entityName');
```

Error Handling

```
1. entity.save();
2.
3. // If the bean has errors, then it did not save...
4. if ( entity.hasErrors() ) {
5.
6.     // errors are returned as a struct
7.     errors = entity.getErrors();
8.
9.     WriteOutput('<h3>Please review the following errors:</h3><ul>');
10.
11.    // Loop over the errors
12.    for ( e in errors ) {
13.        WriteOutput('<li>Attribute: #e# <br> Message: #errors[e]#</li>');
14.    }
15.
16.    WriteOutput('</ul>');
17. } else {
18.     WriteOutput('<h2>Success :: No Errors!</h2>');
19. }
```

Custom Validation

MURA ORM ENTITY

Custom Validation

- Mura allows you to include your own custom validation rules.

Custom Validation

- Mura allows you to include your own custom validation rules.
- In order to do so, you simply include a custom “**validate**” method in your entity’s component file (.CFC), and when a “**save**” call is made on your entity, it will trigger your custom method.

Custom Validation

- Mura allows you to include your own custom validation rules.
- In order to do so, you simply include a custom “**validate**” method in your entity’s component file (.CFC), and when a “**save**” call is made on your entity, it will trigger your custom method.
- The key is you don’t “return” anything, you simply add your custom errors to the existing errors struct.

Custom Validation

- You could also use this to do complex validation requirements.

Custom Validation

- You could also use this to do complex validation requirements.
- For example, maybe you have a custom radio button or checkbox, and when it's selected, your form displays additional form fields that you will then want to be “required”.

Custom Validation

- You could also use this to do complex validation requirements.
- For example, maybe you have a custom radio button or checkbox, and when it's selected, your form displays additional form fields that you will then want to be “required”.
- You could simply check for the existence and/or value of the triggering field, and if it exists, run through each of the additional required form fields in your own validation.

```
1. public any function validate() {  
2.  
3.     // This runs the standard validation ...  
4.     // which will use your property definition's "validate" requirements, etc.  
5.     // and it also gives you a way to reference the entity object itself  
6.     var obj = super.validate();  
7.  
8.     // Obtain a reference to any errors  
9.     var errors = obj.getErrors();  
10.  
11.    // Now, you can check anything you want on your properties or entity attributes  
12.    // For example, if you have a property labeled "wantsmoreoptions" and it's true,  
13.    // You could check for the other fields  
14.    if ( obj.get('wantsmoreoptions') ) {  
15.  
16.        // Yes, they want more options!  
17.        // Make sure they completed them ...  
18.        if ( !Len(obj.get('option99')) ) {  
19.            errors.option99 = 'Option 99 is required.';  
20.        }  
21.        // etc. ...  
22.    }  
23.}  
24.  
25.}
```

Mura ORM Feed

Mura ORM Feed

- As we discussed in the Feed Bean section, Feed beans allow developers to programmatically query Mura for other beans/objects.

Mura ORM Feed

- As we discussed in the Feed Bean section, Feed beans allow developers to programmatically query Mura for other beans/objects.
- These same conventions work for custom objects/entities.

Mura ORM Feed

- The example below simply demonstrates how simple it is to create a feed of a custom entity, using the same methods described from the Feed Bean section.

```
1. // Persons Feed, based on the currently logged in user's userid
2. feedPersons = m.getFeed('person')
   .where()
   .prop('userid')
   .isEQ(m.currentUser('userid'))
   .sort('namelast', 'asc')
   .sort('namefirst', 'desc');
3.
4.
5.
6.
7.
8.
9. // Persons Iterator
10. itPersons = feedPersons.getIterator();
```

Mura ORM Events

Mura ORM Events

- As covered in the Mura ORM Configuration section, any .CFC's discovered under a “**../handlers/**” directory within a “**model**” directory, will automatically be registered as custom event handlers.

Mura ORM Events

- As covered in the Mura ORM Configuration section, any .CFC's discovered under a “**../handlers/**” directory within a “**model**” directory, will automatically be registered as custom event handlers.
- For example, “**../model/handlers/myhandler.cfc**”

Mura ORM Events

- In addition to the various events that are announced during the lifecycle of the application and each request, Mura also announces special events for basic “CRUD” activities such as when saving, updating, or deleting an entity.

Mura ORM Events

- In addition to the various events that are announced during the lifecycle of the application and each request, Mura also announces special events for basic “CRUD” activities such as when saving, updating, or deleting an entity.
- To leverage any of these events, you simply add your methods to your custom event handler(s).

Mura ORM Events

| Event | Description |
|----------------------------|--|
| onBefore{EntityName}Save | Announced before an entity is saved. |
| onAfter{EntityName}Save | Announced after an entity is saved. |
| on{EntityName}Save | Announced after an entity is saved. |
| onBefore{EntityName}Update | Announced before an entity is updated. |
| onAfter{EntityName}Update | Announced after an entity is updated. |
| onBefore{EntityName}Create | Announced before an entity is created. |
| onAfter{EntityName}Create | Announced after an entity is created. |
| onBefore{EntityName}Delete | Announced before an entity is deleted. |
| onAfter{EntityName}Delete | Announced after an entity is deleted. |

Mura ORM Events

```
1.  onBefore{Entity}Save(m) {  
2.      // accessing the bean  
3.      var bean = m.event('bean');  
4.  
5.      // do something here  
6. }
```

Supported ORM Entity Events

- In addition to using an event handler, you could use an ORM Entity Event. The following lists supported ORM Entity events. You could simply include these methods in your entity's .CFC to provide custom logic where needed.

- preLoad
- postLoad
- preUpdate
- postUpdate
- preCreate
- preInsert
- postCreate
- postInsert
- preDelete
- postDelete

Mura Iterators

Mura Iterators

- According to Merriam-Webster, **iterate** means "to say or do again or again and again." In programming, an "iterator" is a design pattern.

Mura Iterators

- According to Merriam-Webster, **iterate** means "to say or do again or again and again." In programming, an "iterator" is a design pattern.
- An iterator wraps a collection or group of objects (e.g., queries/recordsets, arrays, lists, structs, beans, etc.) with common helper methods without the need to fully know or understand the underlying data structure.

Mura Iterators

- According to Merriam-Webster, **iterate** means "to say or do again or again and again." In programming, an "iterator" is a design pattern.
- An iterator wraps a collection or group of objects (e.g., queries/recordsets, arrays, lists, structs, beans, etc.) with common helper methods without the need to fully know or understand the underlying data structure.
- This allows programmers to use common methods for looping over, or iterating over, many different types of collections or groups of objects.

Mura Iterators

- Mura Iterators are an array of objects, and have a number of common helper methods, allowing developers to loop/iterate over, collections/groups of objects.

Mura Iterators

- Mura Iterators are an array of objects, and have a number of common helper methods, allowing developers to loop/iterate over, collections/groups of objects.
- Mura Iterators wrap an underlying query, and use the returned recordset as the basis for its array of objects.

Mura Iterators

- Mura Iterators are an array of objects, and have a number of common helper methods, allowing developers to loop/iterate over, collections/groups of objects.
- Mura Iterators wrap an underlying query, and use the returned recordset as the basis for its array of objects.
- This also means the order of objects in the iterator is determined by the sort order of the underlying recordset.

Mura Iterators

- As we've already seen, Mura relies on the iterator design pattern itself.

Mura Iterators

- As we've already seen, Mura relies on the iterator design pattern itself.
- Many of Mura's common beans include helper methods which return some type of iterator.

Mura Iterators

- As we've already seen, Mura relies on the iterator design pattern itself.
- Many of Mura's common beans include helper methods which return some type of iterator.
- For example, in the Content Bean section, we saw methods such as **getRelatedContentIterator**, **getCommentsIterator**, etc.

Mura Iterators

- If you've worked with recordsets/queries in the past, you'll find iterators to be quite similar.

Mura Iterators

- If you've worked with recordsets/queries in the past, you'll find iterators to be quite similar.
- However, working with iterators gives you so much more power than merely looping over a query/recordset.

Mura Iterators

- If you've worked with recordsets/queries in the past, you'll find iterators to be quite similar.
- However, working with iterators gives you so much more power than merely looping over a query/recordset.
- When iterating over a collection of Mura objects, you'll have access to the entire object itself, including its helper methods.

Mura Iterators

- If you've worked with recordsets/queries in the past, you'll find iterators to be quite similar.
- However, working with iterators gives you so much more power than merely looping over a query/recordset.
- When iterating over a collection of Mura objects, you'll have access to the entire object itself, including its helper methods.
- If you were merely looping over a query/recordset using `<cfoutput>` tags, you would only have access to the fields included in the record itself.

Mura Iterators

- In addition, when working with a standard query/recordset, it can be cumbersome to do things like pagination, or even loop backwards without having to rewrite the underlying query itself.

Mura Iterators

- In addition, when working with a standard query/recordset, it can be cumbersome to do things like pagination, or even loop backwards without having to rewrite the underlying query itself.
- Mura Iterators can do all of these things for you, and so much more.

Iterator Basics

Iterator Basics

- As an object itself, Mura Iterators have a number of attributes you can inspect using its helper methods.

Iterator Basics

- As an object itself, Mura Iterators have a number of attributes you can inspect using its helper methods.
- Iterators hold data pertinent to looping over collections or groups of objects.

Iterator Basics

- As an object itself, Mura Iterators have a number of attributes you can inspect using its helper methods.
- Iterators hold data pertinent to looping over collections or groups of objects.
- For example, Mura Iterators know how many objects there are in the collection it's working with.

Iterator Basics

- As an object itself, Mura Iterators have a number of attributes you can inspect using its helper methods.
- Iterators hold data pertinent to looping over collections or groups of objects.
- For example, Mura Iterators know how many objects there are in the collection it's working with.
- They also know if you've started to loop/iterate over the collection, what object you're currently working with, which "page" your on, and so on.

Iterator Basics

- You can use these helper methods to ask the iterator for the next object, or even the previous object, and then do whatever you want with the object itself, and when finished, move on to the next object.

Iterator Basics

- You can use these helper methods to ask the iterator for the next object, or even the previous object, and then do whatever you want with the object itself, and when finished, move on to the next object.
- For example, you could manipulate each object, and then re-save it. If the objects were content items, you could loop over each object, and then output the associated image and a title to create a slideshow.

Iterator Basics

- You can use these helper methods to ask the iterator for the next object, or even the previous object, and then do whatever you want with the object itself, and when finished, move on to the next object.
- For example, you could manipulate each object, and then re-save it. If the objects were content items, you could loop over each object, and then output the associated image and a title to create a slideshow.
- The possibilities are endless.

Iterator Basics

- While Mura Iterators offer several helper methods, there are only two primary methods you need to know to begin working with them; **hasNext()** and **next()**.

hasNext()

- The **hasNext()** method inspects the underlying collection/array of objects, and then returns true if there are still objects in the array of objects you're working with, or false, if there are no more objects in the array to work with.

next()

- The key method, **next()**, queues up the next object in the collection/array of objects, and gives you a way to reference the current item/object so you can work with it.

Tag-based Example

```
1. <cfloop condition="iterator.hasNext()">
2.   <cfset item = iterator.next()>
3.   <!-- Do something with 'item' -->
4.   <cfdump var="#item#">
5. </cfloop>
```

Script Example

```
1. while ( iterator.hasNext() ) {  
2.     item = iterator.next();  
3.     // Do something with 'item'  
4.     WriteDump( item );  
5. }  
6.  
7. // OR  
8.  
9. do {  
10.    item = iterator.next();  
11.    // Do something with 'item'  
12. } while ( iterator.hasNext() );
```

Conditionals

```
1. if ( iterator.hasNext() ) {  
2.     // Iterator has objects  
3. } else {  
4.     // Iterator is empty ... no objects  
5. }
```

Key Iterator Methods

Key Iterator Methods

- hasNext
- next
- peek
- reset
- end
- hasPrevious
- previous
- setQuery
- getQuery
- getRecordCount
- setPage
- getPage
- pageCount
- setItemsPerPage
- getItemsPerPage
- setStartRow
- getStartRow
- getCurrentIndex

Key Iterator Methods

- hasNext
- next
- peek
- reset
- end
- hasPrevious
- previous
- setQuery
- getQuery
- getRecordCount
- setPage
- getPage
- pageCount
- setItemsPerPage
- getItemsPerPage
- setStartRow
- getStartRow
- getCurrentIndex

For details on these methods, visit <http://docs.getmura.com/v7-1/mura-developers/mura-beans-objects/mura-iterators/key-iterator-methods/>

Feed Bean Iterator

```
1. <cfset feedBean = m.getFeed('content').where().prop('credits').containsValue('Steve')>
2. <cfset it = feedBean.getIterator()>
3.
4. <cfif it.hasNext()>
5.   <cfloop condition="it.hasNext()">
6.     <cfset item = it.next()>
7.     <div>
8.       <a href="#item.getUrl()#"\>
9.         #esapiEncode('html', item.getTitle())#
10.      </a>
11.    </div>
12.  </cfloop>
13. <cfelse>
14.   <p class="alert alert-info">
15.     This iterator doesn't have any items.
16.   </p>
17. </cfif>
```

Content Iterator

- You can use a query to populate and use Mura's **contentIterator** bean, as long as the query contains a "**siteid**" and "**contentid**" column.

```
1. rs = QueryExecute("SELECT contentid, siteid  
2. FROM tcontent  
3. WHERE active = 1  
4. ");  
5.
```

Content Iterator

```
1. rs = QueryExecute("SELECT contentid, siteid  
2. FROM tcontent  
3. WHERE active = 1  
4. ");  
5.  
  
1. it = m.getBean('contentIterator').setQuery(rs);  
2.  
3. if ( it.hasNext() ) {  
4.     item = it.next();  
5.     // Do something with the item  
6.     WriteDump( item.getAllValues() );  
7. } else {  
8.     // No items/records exist  
9. }
```

Query Iterator

- This demonstrates how to use a regular query with Mura's **queryIterator** bean.

```
1. rs = QueryExecute("SELECT *  
2.          FROM someTable  
3.          ");  
4.      );
```

Query Iterator

```
1. rs = QueryExecute("  
2.   SELECT *  
3.   FROM someTable  
4. ");
```

```
1. it = m.getBean('queryIterator').setQuery(rs);  
2.  
3. if ( it.hasNext() ) {  
4.   item = it.next();  
5.   // Do something with the item  
6.   WriteDump( item );  
7. } else {  
8.   // No items/records exist  
9. }
```

Lab Exercise

LET'S PLAY WITH MURA'S BEANS & OBJECTS

Lab Exercise

- See `/3-core-developer/4-mura-beans-and-objects/instructions.md`

Summary

- Introduction
- Common Bean Objects
- Mura Scope Objects
- Custom Objects
- Mura Iterators
- Lab Exercise

Mura Rendering

ALTERING THE DISPLAY

Mura Rendering

- The Mura contentRenderer.cfc
- The Mura [m] Tag
- Mura Modules & Display Objects
- Mura Content Types
- JSON API
- Mura.js
- Caching
- Lab Exercise

contentRenderer.cfc

MURA RENDERING

contentRenderer.cfc

- Core
 - {context}/core/mura/content/contentRenderer.cfc

contentRenderer.cfc

- Core
 - `{context}/core/mura/content/contentRenderer.cfc`
- Custom Rendering Methods

contentRenderer.cfc

- Core
 - {context}/core/mura/content/contentRenderer.cfc
- Custom Rendering Methods

```
1. public string function dspHello() {  
2.     return "<p>Hello from dspHello!</p>";  
3. }
```

contentRenderer.cfc

- Core
 - {context}/core/mura/content/contentRenderer.cfc
- Custom Rendering Methods

```
1. public string function dspHello() {  
2.     return "<p>Hello from dspHello!</p>";  
3. }
```

```
1. <cfoutput>  
2. #m.dspHello()#  
3. </cfoutput>
```

contentRenderer.cfc

- Core
 - {context}/core/mura/content/contentRenderer.cfc
- Custom Rendering Methods

```
1. public string function dspHello() {  
2.     return "<p>Hello from dspHello!</p>";  
3. }
```

```
1. <cfoutput>  
2. #m.dspHello()#  
3. </cfoutput>
```

```
1. <p>Hello from dspHello!</p>
```

contentRenderer.cfc

- Lookup Hierarchy
 - Theme
 -/themes/{Theme}/contentRenderer.cfc
 - Site
 - {context}/sites/{SiteID}/contentRenderer.cfc
 - Core
 - {context}/core/mura/content/contentRenderer.cfc

contentRenderer.cfc

- this.{settings}

contentRenderer.cfc

- this.{settings}
- Other Template Helper Methods

contentRenderer.cfc

- this.{settings}
- Other Template Helper Methods
 - mdspInclude('path/to/file.cfm')

contentRenderer.cfc

- this.{settings}
- Other Template Helper Methods
 - mdspInclude('path/to/file.cfm')
 - m dspThemeInclude('path/to/file.cfm')

contentRenderer.cfc

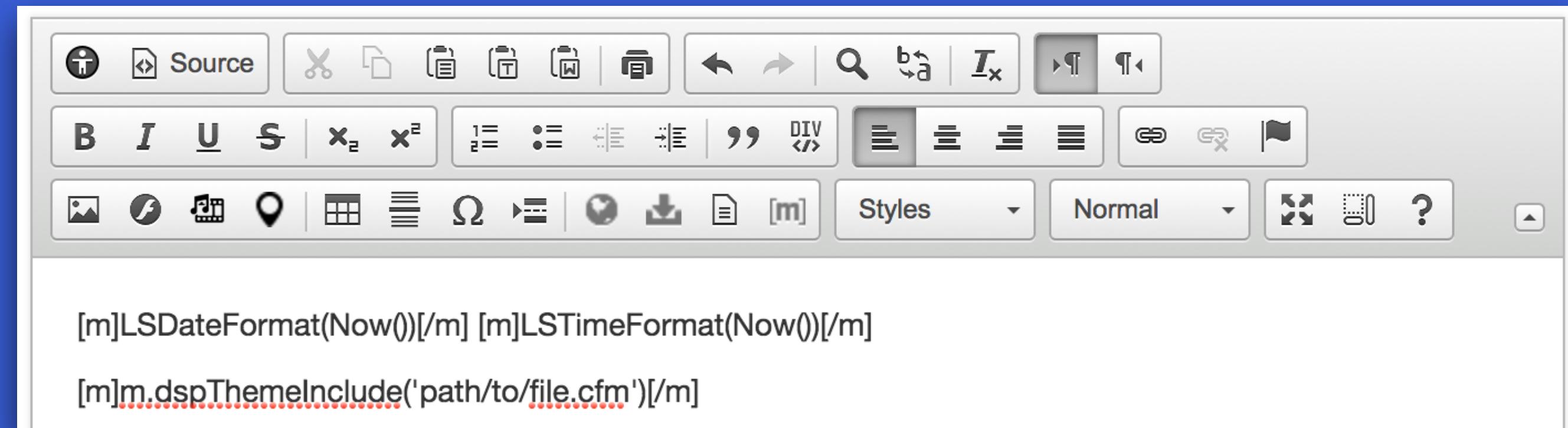
- this.{settings}
- Other Template Helper Methods
 - mdspInclude('path/to/file.cfm')
 - m dspThemeInclude('path/to/file.cfm')
 - & many more!

The Mura [m] Tag

MURA RENDERING

The Mura [m] Tag

- Use “Mura Tags” when you want to output dynamic code within the HTML Editor.



The Mura [m] Tag

- Use “Mura Tags” when you want to output dynamic code within the HTML Editor.
- We use Mura Tags because hash tags (#) are ignored when entered as text in the HTML Editor.

The Mura [m] Tag

- Use “Mura Tags” when you want to output dynamic code within the HTML Editor.
- We use Mura Tags because hash tags (#) are ignored when entered as text in the HTML Editor.
- Hence, you can only render a CFML variable, or function call.

The Mura [m] Tag

- Globally Enable/Disable the Mura Tag
 - **settings.ini.cfm**
 - **enablemuratag={true|false}**

The Mura [m] Tag

- Enable/Disable the Mura Tag on a Site by Site Basis
 - Site or Theme **contentRenderer.cfc**
 - **this.enablemuratag={true | false}**

The Mura [m] Tag

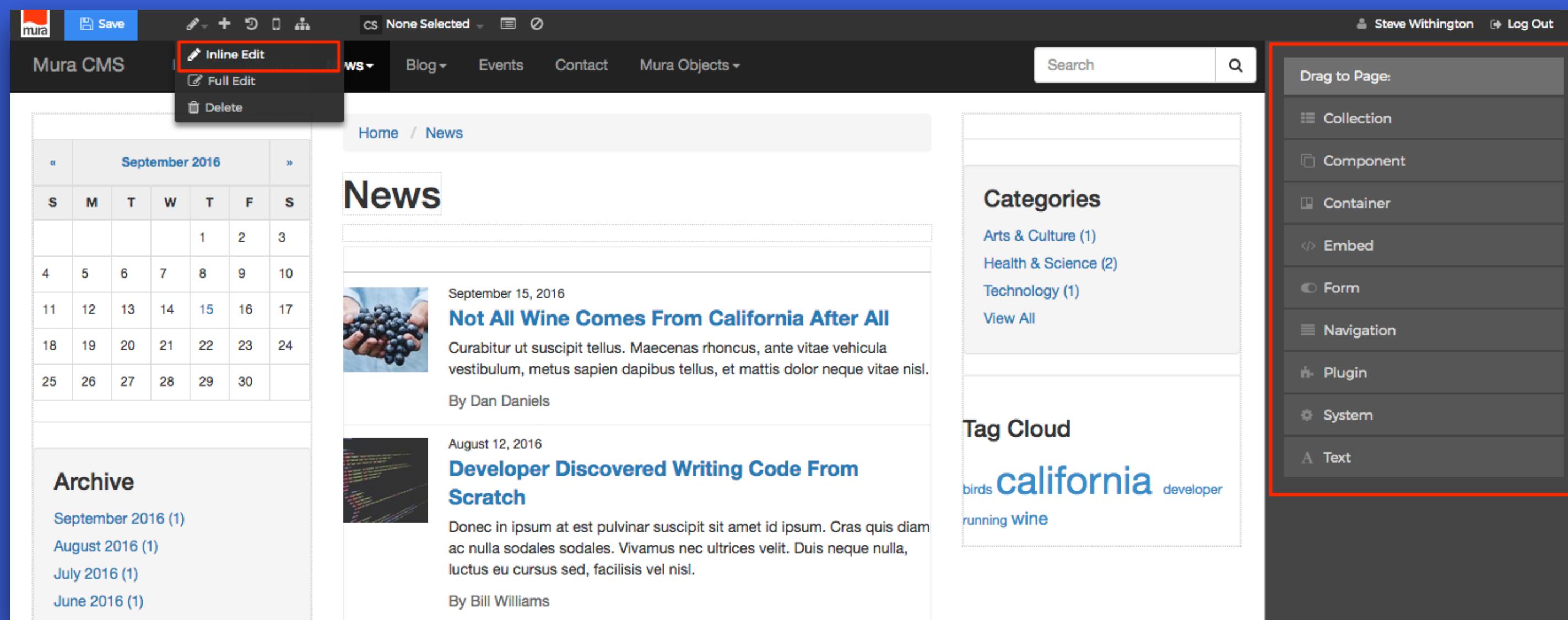
- When you expect Mura Tags in the output
 - **m.setDynamicContent(str)**

Mura Modules & Display Objects

MURA RENDERING

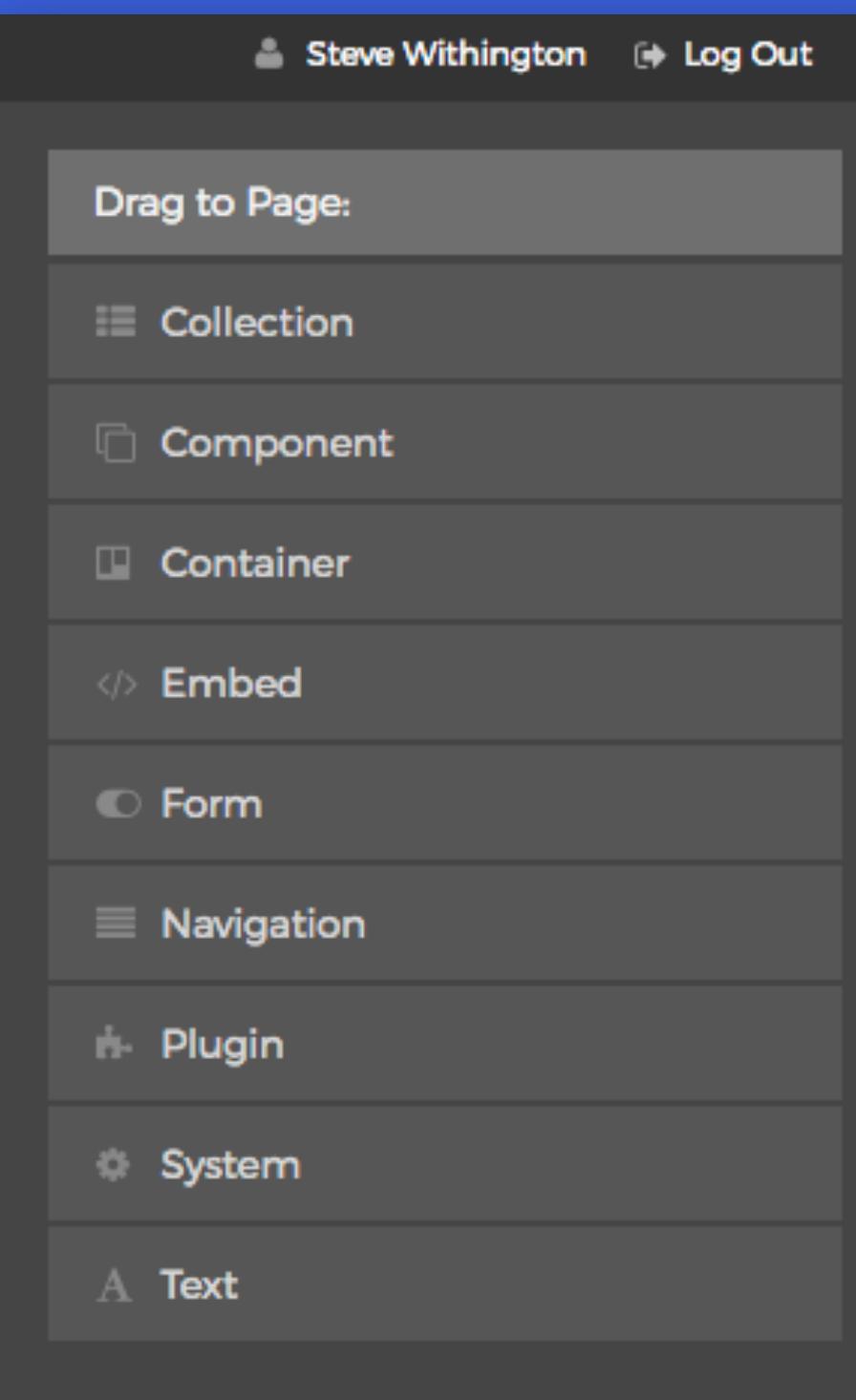
Modules

- Managed via front-end only (*Mura 7+*)



Modules

- Managed via front-end only (*Mura 7+*)



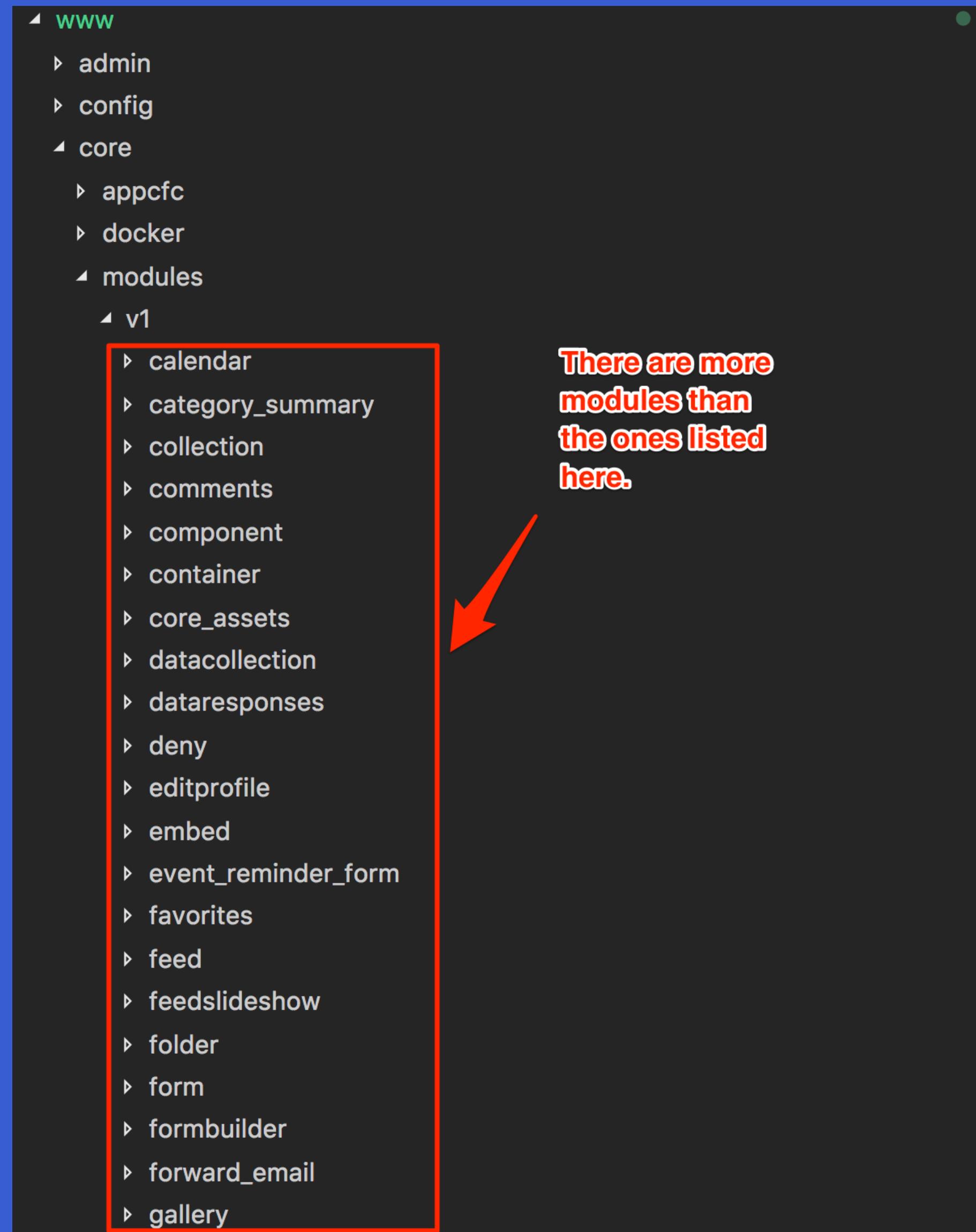
Modifying Mura's Modules

MURA MODULES & DISPLAY OBJECTS

Modifying Mura's Modules

- Mura's base modules (display objects) are located under:
 - {context}/core/modules/v1/

Modifying Mura's Modules



```
▶ www
  ▶ admin
  ▶ config
  ▲ core
    ▶ appfc
    ▶ docker
  ▲ modules
    ▲ v1
      ▶ calendar
      ▶ category_summary
      ▶ collection
      ▶ comments
      ▶ component
      ▶ container
      ▶ core_assets
      ▶ datacollection
      ▶ dataresponses
      ▶ deny
      ▶ editprofile
      ▶ embed
      ▶ event_reminder_form
      ▶ favorites
      ▶ feed
      ▶ feedslideshow
      ▶ folder
      ▶ form
      ▶ formbuilder
      ▶ forward_email
      ▶ gallery
```

There are more modules than the ones listed here.

Modifying Mura's Modules

- Registering a Custom “modules” Directory

```
1. m.siteConfig().registerModuleDir(  
2.     dir='/path/to/your/modules/'  
3. );
```

Modifying Mura's Modules

- Lookup Hierarchy
 - {RegisteredModuleDirectory}
 - . . / {module} / modules /
 - . . / themes / {ThemeName} / modules /
 - {context} / sites / {SiteName} / modules /
 - {context} / modules /
 - {context} / core / modules / v1 / core_assets / modules /

Anatomy of a Module

MURA MODULES & DISPLAY OBJECTS

Anatomy of a Module

- `../modules/mymodule/`
 - `index.cfm`
 - `config.xml.cfm`
 - `configurator.cfm`
 - `model/`
 - See Mura ORM Configuration section
 - `content_types/`
 - See Mura Content Types section
 - `modules/`
 - Yes, you're reading that correctly!

index.cfm

index.cfm

```
1. <cfparam name="objectparams.mytext" default="" />
2.
3. <cfoutput>
4.   <h2>My Object</h2>
5.
6.   <cfif Len(objectparams.mytext)>
7.     <p>
8.       This object has a configurator, and the value of "objectparams.mytext" is:<br>
9.       <strong>#esapiEncode('html', objectparams.mytext)#</strong>
10.    </p>
11.    <cfelse>
12.      <!-- No value entered for "objectparams.mytext" -->
13.    </cfif>
14.  </cfoutput>
```

index.cfm

- Loading CSS or JavaScript

```
1.  <!-- Add the following to your "index.cfm" file -->
2.  <script>
3.    Mura(function(m) {
4.      m.loader()
5.        .loadcss(m.themepath + '/modules/myobject/my.css')
6.        .loadjs(
7.          m.themepath + '/modules/myobject/my.js',
8.          m.themepath + '/modules/myobject/other.js',
9.          function() {
10.            // Do something with the loaded JS, if desired
11.          }
12.        );
13.      });
14.    </script>
```

index.cfm

- Rendering via JavaScript
 - <cfset objectparams.render="client">

```
1. // ../yourmodule/model/handlers/yourhandler.cfc
2.
3. component extends='mura.cfobject' {
4.
5.     function onRenderStart(m) {
6.         // if script should be included in the <head>
7.         arguments.m.addToHTMLHeadQueue('<script src="/path/to/script.js"></script>');
8.
9.         // OR
10.
11.        // if script should be included before closing </body> tag
12.        arguments.m.addToHTMLFootQueue('<script src="/path/to/script.js"></script>');
13.    }
14.
15. }
```

config.xml.cfm

config.xml.cfm

- For details, refer to <https://docs.getmura.com/v7-1/mura-developers/mura-beans-objects/custom-objects/class-extensions/define-with-xml/elements-of-the-config-xml-cfm-file/>

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <mura name="My Module" contenttypes="*" iconclass="mi-rebel">
3.      <!-- May also include other elements here -->
4.  </mura>
```

config.xml.cfm

```
<mura name="My Module"  
      contenttypes="*" />
```

config.xml.cfm

```
<mura name="My Module"  
      contenttypes="*" />
```

** = all content types*

config.xml.cfm

```
<mura name="My Module"  
      contenttypes="*"/>
```

** = all content types*

contenttypes=" {Type|Type/Subtype} ,Folder/Blog,Folder/News"

config.xml.cfm

```
<mura name="My Module"  
      contenttypes="*"/>
```

** = all content types*

contenttypes=" {Type|Type/Subtype} ,Folder/Blog,Folder/News"

contenttypes="" *Blank = never display as a draggable option*

config.xml.cfm

```
<mura name="My Module"  
      contenttypes="*"  
      condition="(m.content('title') eq 'News')"/>
```

config.xml.cfm

```
<mura name="My Module"  
      contenttypes="*"  
      iconclass="mi-somefontawesomesomeiconclass" />
```

config.xml.cfm

```
<mura name="My Module"  
      contenttypes="*"  
  
      <extensions> ... </extensions>  
  
</mura>
```

config.xml.cfm

```
<mura name="My Module"  
      contenttypes="*"  
  
      <extensions> ... </extensions>  
  
      <imagesizes>  
          <imagesize name="yoursize" width="100" height="100"/>  
      </imagesizes>  
  
</mura>
```

configurator.cfm

configurator.cfm

```
<cfparam name="objectparams.mytext" default="">

<cfoutput>

    <div class="mura-control-group">
        <label class="mura-control-label">My Text</label>

        <input type="text"
               name="mytext"
               class="objectParam"
               value="#esapiEncode('html_attr', objectparams.mytext)#">
    </div>

</cfoutput>
```

configurator.cfm

```
<cfparam name="objectparams.mytext" default="">

<cfoutput>
    <cf_objectconfigurator>
        <div class="mura-control-group">
            <label class="mura-control-label">My Text</label>

            <input type="text"
                   name="mytext"
                   class="objectParam"
                   value="#esapiEncode( 'html_attr', objectparams.mytext)#">
        </div>
    </cf_objectconfigurator>
</cfoutput>
```

model/

Anatomy of a Module

- Modules can have their own “**model**” directory
 - `.../mymodule/model/`

Anatomy of a Module

- Modules can have their own “model” directory
 - `.../mymodule/model/`
 - And their own “**beans**” directory
 - `.../mymodule/model/beans/`

Anatomy of a Module

- Modules can have their own “model” directory
 - `.../mymodule/model/`
 - And their own “beans” directory
 - `.../mymodule/model/beans/`
- And “**handlers**” directories too
 - `.../mymodule/model/handlers/`

Anatomy of a Module

- Modules can have their own “model” directory
 - `.../mymodule/model/`
 - And their own “beans” directory
 - `.../mymodule/model/beans/`
- And “**handlers**” directories too
 - `.../mymodule/model/handlers/`
 - `.../mymodule/model/services/handlers/`

Mura Content Types

MURA RENDERING

Content Types

- Control body by Type/Subtype

Content Types

- Control body by Type/Subtype
 - Folder/Contacts
 - Page/Contact

Content Types

- Control body by Type/Subtype
 - {context}/themes/{ThemeName}/content_types/
 - {context}/sites/{SiteID}/themes/{ThemeName}/content_types/
 - {context}/sites/{SiteID}/content_types/
 - ../content_types/{type}/content_types/
 - ../content_Types/{type_subtype}/content_types/

Content Types

- Or, you can register any directory you want

```
1. m.siteConfig().registerContentTypeDir(  
2.     '/path/to/your/content_types/'  
3. );  
4.  
5. // Path is a logical path to a CFML directory  
6. // Usually registered in onApplicationLoad();
```

Content Types

- Control Body By Type/Subtype

Content Types

- Control Body By Type/Subtype

```
1.  ../content_types/{type}/  
2.    
3. // If you wish to target 'Page'  
4.  ../content_types/page/index.cfm
```

Content Types

- Control Body By Type/Subtype

```
1.  ../content_types/{type}/  
2.  
3. // If you wish to target 'Page'  
4. ../content_types/page/index.cfm
```

```
1.  ../content_types/{type}_{subtype}/  
2.  
3. // If you wish to target 'Page/Contact'  
4. ../content_types/page_contact/index.cfm
```

Content Types

- Anatomy of a {type}_{subtype} Directory
 - `../content_types/{type}_{subtype}/index.cfm`

Content Types

- Anatomy of a {type}_{subtype} Directory
 - `../content_types/{type}_{subtype}/index.cfm`
 - `../content_types/{type}_{subtype}/config.xml.cfm`

Content Types

- Anatomy of a {type}_{subtype} Directory
 - `../content_types/{type}_{subtype}/index.cfm`
 - `../content_types/{type}_{subtype}/config.xml.cfm`
 - `index.cfm` = the body/view

Content Types

- Anatomy of a {type}_{subtype} Directory
 - `../content_types/{type}_{subtype}/index.cfm`
 - `../content_types/{type}_{subtype}/config.xml.cfm`
 - `index.cfm` = the body/view
 - `config.xml.cfm` = the configuration

Content Types

- Anatomy of a {type}_{subtype} Directory
 - `../content_types/{type}_{subtype}/index.cfm`
 - `../content_types/{type}_{subtype}/config.xml.cfm`
 - `../content_types/{type}_{subtype}/model/`

Content Types

- Anatomy of a {type}_{subtype} Directory
 - `../content_types/{type}_{subtype}/index.cfm`
 - `../content_types/{type}_{subtype}/config.xml.cfm`
 - `../content_types/{type}_{subtype}/model/`
 - `../content_types/{type}_{subtype}/model/beans/`

Content Types

- Anatomy of a {type}_{subtype} Directory
 - `../content_types/{type}_{subtype}/index.cfm`
 - `../content_types/{type}_{subtype}/config.xml.cfm`
 - `../content_types/{type}_{subtype}/model/`
 - `../content_types/{type}_{subtype}/model/beans/`
 - `../content_types/{type}_{subtype}/model/handlers/`

Content Types

- Anatomy of a {type}_{subtype} Directory
 - `../content_types/{type}_{subtype}/index.cfm`
 - `../content_types/{type}_{subtype}/config.xml.cfm`
 - `../content_types/{type}_{subtype}/model/`
 - `../content_types/{type}_{subtype}/model/beans/`
 - `../content_types/{type}_{subtype}/model/handlers/`
 - `../content_types/{type}_{subtype}/modules/`

Content Types

- Anatomy of a {type}_{subtype} Directory
 - `../content_types/{type}_{subtype}/index.cfm`
 - `../content_types/{type}_{subtype}/config.xml.cfm`
 - `../content_types/{type}_{subtype}/model/`
 - `../content_types/{type}_{subtype}/model/beans/`
 - `../content_types/{type}_{subtype}/model/handlers/`
 - `../content_types/{type}_{subtype}/modules/`
 - `../content_types/{type}_{subtype}/content_types/`

Content Types

- Example “index.cfm” File

```
1. <cfoutput>
2.   <article class="article">
3.     <!-- Page Title -->
4.     <h2 class="article__title">#m.renderEditableAttribute(attribute='title')#</h2>
5.
6.     <!-- Primary Image -->
7.     <cfif m.content().hasImage(usePlaceholder=false)>
8.       <figure class="article__image">
9.         
11.       </figure>
12.     </cfif>
13.
14.     <!-- Content -->
15.     <div class="article__body">
16.       #m.renderEditableAttribute(attribute='body', type='htmlEditor')#
17.     </div>
18.   </article>
19. </cfoutput>
```

Content Types

- How To Output Modules

```
1. <cfoutput>  
2. #m.dspObject(object='{objectName}')#  
3. </cfoutput>
```

Content Types

- Including Configurable Modules
 - There can only be one (1) module that is configurable

```
1. <cfoutput>
2. #m.dspObject(object='{objectName}', objectparams=objectparams)#
3. </cfoutput>
```

Content Types

- Including Configurable Modules
 - There can only be one (1) module that is configurable

```
1. <cfoutput>
2. #m.dspObject(object='{objectName}', objectparams=objectparams)#
3. </cfoutput>
```

```
1. <cfoutput>
2. #m.dspObject(
3.     object='object1',
4.     objectparams={param1=objectparams.param1, param2=objectparams.param2}
5. )#
6.
7. #m.dspObject(object='object2', objectparams=objectparams)#
8. </cfoutput>
```

JSON API

JSON API

- Docs:

<https://docs.getmura.com/v7-1/mura-developers/mura-rendering/json-api/>

JSON API

- **Docs:**

<https://docs.getmura.com/v7-1/mura-developers/mura-rendering/json-api/>

- **Example JSON Tests:**

<http://bit.ly/mura-json-api>

JSON API

- **Docs:**

<https://docs.getmura.com/v7-1/mura-developers/mura-rendering/json-api/>

- **Example JSON Tests:**

<http://bit.ly/mura-json-api>

- **Postman:**

<https://www.getpostman.com>

- **Swagger:**

<https://swagger.io>

Mura.js

MURA RENDERING

Mura.js

- A lightweight utility to decouple dependency on jQuery

Mura.js

- Originally began as a lightweight utility to decouple dependency on jQuery
- Today, it's a JavaScript framework for interacting with the Mura JSON API

Mura.js

- Familiar Syntax

```
<script>
Mura(function(m){
    // Mura ready
});
</script>
```

Mura.js

- Selecting Elements

```
<script>
Mura(function(m) {
  m('.target').each(function() {
    m(this).html('Mura found you!');
  });
});
</script>
```

Mura.js

- AJAX Support

```
<script>
Mura.ajax({
    type: 'post',
    url: 'https://domain.com/path/',
    data: { key: 'value' },
    success: function(response) {
        console.log(response);
    },
    error: function(err) {
        console.log(err);
    }
});
</script>
```

Mura.js

- Promise Support

```
<script>
Mura.get('https://domain.com/path/', { key: 'value' })
  .then(function(response) {
    // handle success
  })
  .catch(function(err) {
    // handle error
  });
</script>
```

Mura.js

- Promise Support

```
<script>
Mura.get('https://domain.com/path/', { key: 'value' })
  .then(function(response) {
    // handle success
    Mura('#target').html(response.data.html);
  })
  .catch(function(err) {
    // handle error
    console.log(err);
  });
</script>
```

Mura.js

- Wrap in the Mura “ready” method for easier syntax

```
<script>
Mura(function(m) {
    m.get('https://domain.com/path/', { key: 'value' })
        .then(function(response) {
            // handle success
            m('#target').html(response.data.html);
        })
        .catch(function(err) {
            // handle error
            console.log(err);
        });
});
</script>
```

Mura.js

- Chaining Promises

Mura.js

- Chaining Promises
 - Avoid the “pyramid of doom”

```
<script>
Mura.post('https://domain.com/path/', { apikey: 'value' })
  .then(function(result) {
    if ( result.hasOwnProperty('id') ) {
      // *** Nested call ... pyramid of doom! ***
      Mura.get('https://domain.com/path/?id=' + id)
        .then(function(result) {
          console.log(result);
        })
        .catch(function(err) {
          throw new Error(err);
        });
    } else {
      throw new Error('result does not contain an id');
    }
  })
  .catch(function(err) {
    throw new Error(err);
  });
</script>
```

Mura.js

- Chaining Promises
- Avoid the “pyramid of doom”

```
if ( result.hasOwnProperty('id') ) {  
    // *** Nested call ... pyramid of doom! ***  
    Mura.get('https://domain.com/path/?id=' + id)  
        .then(function(result) {  
            console.log(result);  
        })  
        .catch(function(err) {  
            throw new Error(err);  
        });  
} else {  
    throw new Error('result does not contain an id');  
}
```

Mura.js

- Chaining Promises
 - Avoid the “pyramid of doom”
 - Wrap the Ajax calls with new Promises instead

```
var exampleFunc1 = function() {
    return new Promise(function (resolve, reject) {
        Mura.post('https://domain.com/path/', { apikey: 'value' })
            .then(function(result) {
                resolve(result);
            })
            .catch(function(err) {
                reject(err);
            });
    });
};
```

```
var exampleFunc1 = function() {
    return new Promise(function (resolve, reject) {
        Mura.post('https://domain.com/path/', { apikey: 'value' })
            .then(function(result) {
                resolve(result);
            })
            .catch(function(err) {
                reject(err);
            });
    });
};
```

```
var exampleFunc1 = function() {
  return new Promise(function (resolve, reject) {
    Mura.post('https://domain.com/path/', {apikey: 'value'})
      .then(function(result) {
        resolve(result);
      })
      .catch(function(err) {
        reject(err);
      });
  });
};
```

```
var exampleFunc1 = function() {
  return new Promise(function (resolve, reject) {
    Mura.post('https://domain.com/path/', { apikey: 'value' })
      .then(function(result) {
        resolve(result);
      })
      .catch(function(err) {
        reject(err);
      });
  });
};
```

Mura.js

- Chaining Promises
 - Avoid the “pyramid of doom”
 - Wrap the Ajax calls with new Promises instead
- Example Gist:

<https://gist.github.com/stevewithington/069926df0df1ec7bacebe555e1e506d3>

Mura.js

- Chaining Promises
 - Avoid the “pyramid of doom”
 - Wrap the Ajax calls with new Promises instead

- Example Gist:

<https://gist.github.com/stevewithington/069926df0df1ec7bacebe555e1e506d3>

- Learn more about chaining Promises:

<https://javascript.info/promise-chaining>

Mura.js

- DOM Event Handlers

```
<script>
Mura('#mybutton').on('click', function(e) {
  e.preventDefault();
  console.log(e);
});
</script>
```

Mura.js

- **Mura.DOMSelection** class
 - Wraps selected target via the Mura() method

Mura.js

- **Mura.DOMSelection** class
 - Wraps selected target via the Mura() method
 - Allows you to handle selection as a single object or a collection

Mura.js

- **Mura.DOMSelection** class
 - Wraps selected target via the Mura() method
 - Allows you to handle selection as a single object or a collection
 - **Single object**

```
Mura('.target').html('Hello world!');
```

Mura.js

- **Mura.DOMSelection** class
 - Wraps selected target via the Mura() method
 - Allows you to handle selection as a single object or a collection
 - **Single object**

```
Mura('.target').html('Hello world!');
```

- **Collection**

```
Mura('.target').each(function() {  
  Mura(this).html('Hello world');  
});
```

Mura.js

- Supported DOMSelection methods:
 - <https://github.com/blueriver/MuraJS/blob/master/src/core/domselection.js>

Mura ORM With Mura.js

MURA.JS

Mura ORM With Mura.js

- Loading/Reading Mura ORM objects/entities

```
<script>
var personid = 'some-uuid';

Mura.getEntity('person')
    .loadBy('personid', personid)
    .then(function(person) {
        console.log(person);
    })
    .catch(function(err) {
        console.log(err.get('errors'));
    });
</script>
```

Mura ORM With Mura.js

- **Creating/Updating** Mura ORM objects/entities

Mura.js

- **Creating/Updating** Mura ORM objects/entities

```
var getPersonByID = function(personid) {
    return new Promise(function(resolve, reject) {
        Mura.getFeed('person').loadBy('personid', personid)
            .then(function(person) {
                resolve(person);
            })
            .catch(function(err) {
                reject(err);
                console.log(err.get('errors'));
            });
    });
};
```

Mura.js

- **Creating/Updating** Mura ORM objects/entities

```
var savePerson = function(person) {
    return new Promise(function(resolve, reject) {
        person.save()
            .then(function(result) {
                resolve(result);
            })
            .catch(function(err) {
                reject(err);
            });
    });
}
```

Mura.js

- **Creating/Updating** Mura ORM objects/entities

```
getPersonByID('some-uuid')
  .then(function(person) {
    person.set('namelast', 'Withington');
    return savePerson(person);
})
  .then(function(result) {
    console.log(result);
})
  .catch(function(err) {
    console.log(err);
});
```

Mura ORM With Mura.js

- **Deleting** Mura ORM objects/entities

Mura ORM With Mura.js

- Deleting Mura ORM objects/entities

```
var getPersonByID = function(personid) {
    return new Promise(function(resolve, reject) {
        Mura.getFeed('person').loadBy('personid', personid)
            .then(function(person) {
                resolve(person);
            })
            .catch(function(err) {
                reject(err);
                console.log(err.get('errors'));
            });
    });
};
```

Mura ORM With Mura.js

- Deleting Mura ORM objects/entities

```
var deletePerson = function(person) {
    return new Promise(function(resolve, reject) {
        person.delete()
            .then(function(result) {
                resolve(result);
            })
            .catch(function(err) {
                reject(err);
            });
    });
}
```

Mura ORM With Mura.js

- Deleting Mura ORM objects/entities

```
getPersonByID('some-uuid')
  .then(function(person) {
    return deletePerson(person);
  })
  .then(function(result) {
    console.log(result);
  })
  .catch(function(err) {
    console.log(err);
  });
});
```

Mura.js ORM Feed API

MURA ORM WITH MURA.JS

Mura.js ORM Feed API

```
1. var person;
2.
3. Mura
4.   .getFeed('person')
5.   .where() //optional
6.   .prop('namelast')
7.   .isEQ('Levine')
8.   .orProp('namelast')
9.   .beginsWith('Withing')
10.  .getQuery()
11.  .then(function(people) {
12.    // handle success
13.    people.each(function(person, idx) {
14.      result = person.get('namefirst') + ' ' + person.get('namelast');
15.      console.log(result);
16.    });
17.  })
18.  .catch(function(err) {
19.    // handle error
20.    console.log(err);
21.  });

```

Mura.js ORM Feed API

```
1. var widget;
2.
3. Mura
4.   .getFeed('widget')
5.   .aggregate('count', '*')
6.   .aggregate('sum', 'price')
7.   .aggregate('min', 'price')
8.   .aggregate('max', 'price')
9.   .aggregate('avg', 'price')
10.  .getQuery()
11.  .then(function(widgets) {
12.    // handle success
13.    console.log(widgets.getAll());
14.  })
15.  .catch(function(err) {
16.    // handle error
17.    console.log(err);
18.  });
19.
```

Mura.js ORM Feed API

Mura

```
.getFeed(entityName)
.where()
.prop(property)
.orProp(property)
.isEQ(criteria)
.isNEQ(criteria)
.isLT(criteria)
.isLTE(criteria)
.isGT(criteria)
.isGTE(criteria)
.containsValue(criteria)
.null()

.beginsWith(criteria)
.endsWith(criteria)
.openGrouping()
.andOpenGrouping()
.closeGrouping()
.sort(property, asc|desc)
.itemsPerPage(itemsPerPage)
.maxItems(maxItems)
.isIn(list criteria)
.isNotIn(list criteria)
.innerJoin(relatedEntity)
.leftjoin(relatedEntity)
.aggregate()
.groupBy(prop)
.getQuery()
```

Loading JavaScript & CSS Files

MURA.JS

Loading JavaScript & CSS Files with Mura.js

- `Mura.loader()`

Loading JavaScript & CSS Files with Mura.js

- `Mura.loader()`
 - This method is for JavaScript and CSS parallel loading with dependencies management.

Loading JavaScript & CSS Files with Mura.js

- `Mura.loader()`
 - This method is for JavaScript and CSS parallel loading with dependencies management.
 - Using this method also prevents duplicate JavaScript and/or CSS files from being loaded.

Loading JavaScript & CSS Files with Mura.js

- `Mura.loader()`
 - This method is for JavaScript and CSS parallel loading with dependencies management.
 - Using this method also prevents duplicate JavaScript and/or CSS files from being loaded.
 - There are two (2) primary methods associated with `Mura.loader()`

Loading JavaScript & CSS Files with Mura.js

- `Mura.loader().loadjs(url, cb)`

| Parameter | Description |
|-----------|---|
| url | If the first parameter is an <code>array</code> , files will be loaded <i>asynchronously</i> . If it's a <code>string</code> , the files will be loaded <i>synchronously</i> . If the file is located under your theme, you may use <code>m.themepath</code> to dynamically generate the path to the theme location. For example: <code>loadjs(m.themepath + '/script.js')</code> |
| cb | A callback function to execute when all scripts have been loaded. |

Loading JavaScript & CSS Files with Mura.js

- `Mura.loader().loadcss(url, attrs, cb)`

| Parameter | Description |
|-----------|---|
| url | The URL for the CSS file. If the file is located under your theme, you may use <code>m.themepath</code> to dynamically generate the path to the theme location. For example: <code>loadcss(m.themepath + '/file.css')</code> |
| attrs | You may optionally pass in an object to specify the <code>media</code> type attribute for the CSS file. For example: <code>loadcss(m.themepath + '/file/css', {media: "print"})</code> Valid options include: <ul style="list-style-type: none">• all• aural• braille• embossed• handheld• print• projection• screen• tty• tv |
| cb | A callback function which executes immediately. |

Examples

LOADING JAVASCRIPT & CSS FILES WITH MURA.JS

Loading JavaScript & CSS Files with Mura.js

```
1. <script>
2.   Mura(function(m) {
3.     m.loader()
4.       .loadjs(m.themepath + '/js/mylibrary.js', function() /* callback */);
5.   });
6. </script>
```

Loading JavaScript & CSS Files with Mura.js

```
1.  <script>
2.  Mura(function(m) {
3.    m.loader()
4.      .loadjs(m.themepath + '/myLib.js')
5.      .loadjs(
6.        m.themepath + '/myRequiredLib.js',
7.        m.themepath + '/myDependentLib.js',
8.        function() {
9.          // callback
10.        });
11.  });
12. </script>
```

Loading JavaScript & CSS Files with Mura.js

```
1. <script>
2. Mura(function(m) {
3.   m.loader()
4.     .loadcss(m.themepath + '/path/to/all.css', { media: 'all' })
5.     .loadcss(m.themepath + '/path/to/print.css', { media: 'print' })
6.     .loadjs(
7.       m.themepath + '/path/to/script1.js',
8.       m.themepath + '/path/to/script2.js',
9.       function() {
10.         // Now do something with the loaded JS
11.       });
12.   });
13. </script>
```

Modules With Mura.js

MURA.JS

Modules With Mura.js

- Rendering modules purely with JavaScript

Modules With Mura.js

- Rendering modules purely with JavaScript
- The module’s “index.cfm” file should only contain the following line of code, which informs Mura it has nothing to render via server-side processing, and all rendering will occur via the “client” or browser.

```
1. <cfset objectparams.render="client">
```

Modules With Mura.js

- To load your script(s), you'll need to use a custom event handler, and register the “onRenderStart” event to dynamically load your script file(s).

```
1. // ../yourmodule/model/handlers/yourhandler.cfc
2.
3. component extends='mura.cfobject' {
4.
5.     function onRenderStart(m) {
6.         // if script should be included in the <head>
7.         arguments.m.addToHTMLHeadQueue('<script src="/path/to/script.js"></script>');
8.
9.         // OR
10.
11.        // if script should be included before closing </body> tag
12.        arguments.m.addToHTMLFootQueue('<script src="/path/to/script.js"></script>');
13.    }
14.
15. }
```

Modules With Mura.js

- Running Scripts on Every Request

```
1. // your-script.js  
2. console.log('Hello from your module!');
```

Modules With Mura.js

- Running Scripts Only If Your Module Has Been Applied

```
1. // your-script.js
2. Mura.Module.YourModuleDirectoryName = Mura.UI.extend({
3.   render: function() {
4.     // Your code goes here ...
5.     return this;
6.   }
7.});
```

Modules With Mura.js

- `this.context`

Modules With Mura.js

- `this.context`
 - How to target the container element of where the module has been applied to the layout.

Modules With Mura.js

- `this.context`
 - How to target the container element of where the module has been applied to the layout.
 - How to access the “`objectparams`”, or the data collected via a configurator.

Modules With Mura.js

- `this.context.targetEl`
- Target the container element of where the module has been applied to the layout. Each instance will have its own, unique, container

Modules With Mura.js

- `this.context.targetEl`
 - Target the container element of where the module has been applied to the layout. Each instance will have its own, unique, container
- `this.context.{yourObjectParam}`
 - Use this to access any “objectparams”, or data collected via a configurator.

Modules With Mura.js

```
1. // your-script.js
2. Mura.Module.YourModuleDirectoryName = Mura.UI.extend({
3.
4.     render: function() {
5.         // reference to the container element
6.         this.container = Mura(this.context.targetEl);
7.
8.         // assuming you have an objectparam named 'mytext'
9.         var txt = this.context.mytext || 'mytext is empty';
10.
11.        // Having fun by replacing the content of the container
12.        // Remember, using Mura.js, we can do jQuery-esque DOM manipulation
13.        this.container.html('<h3>Hello from yourDisplayObject</h3>');
14.        this.container.append('<h4>' + txt + '</h4>');
15.
16.        // The rest of your code goes here ...
17.        return this;
18.    }
19.
20.});
```

Modules With Mura.js

- <https://github.com/blueriver/MuraJS/blob/master/src/core/ui.js>
- <https://github.com/stevewithington/muracontacts/tree/js>

Forms & Mura.js

MURA.JS

Forms & Mura.js

- See <https://docs.getmura.com/v7-1/mura-developers/mura-rendering/murajs/forms-mura-js/>

Caching

MURA RENDERING

Caching

- Mura has a built-in caching mechanism that, when enabled, will store data in memory for reuse.

Caching

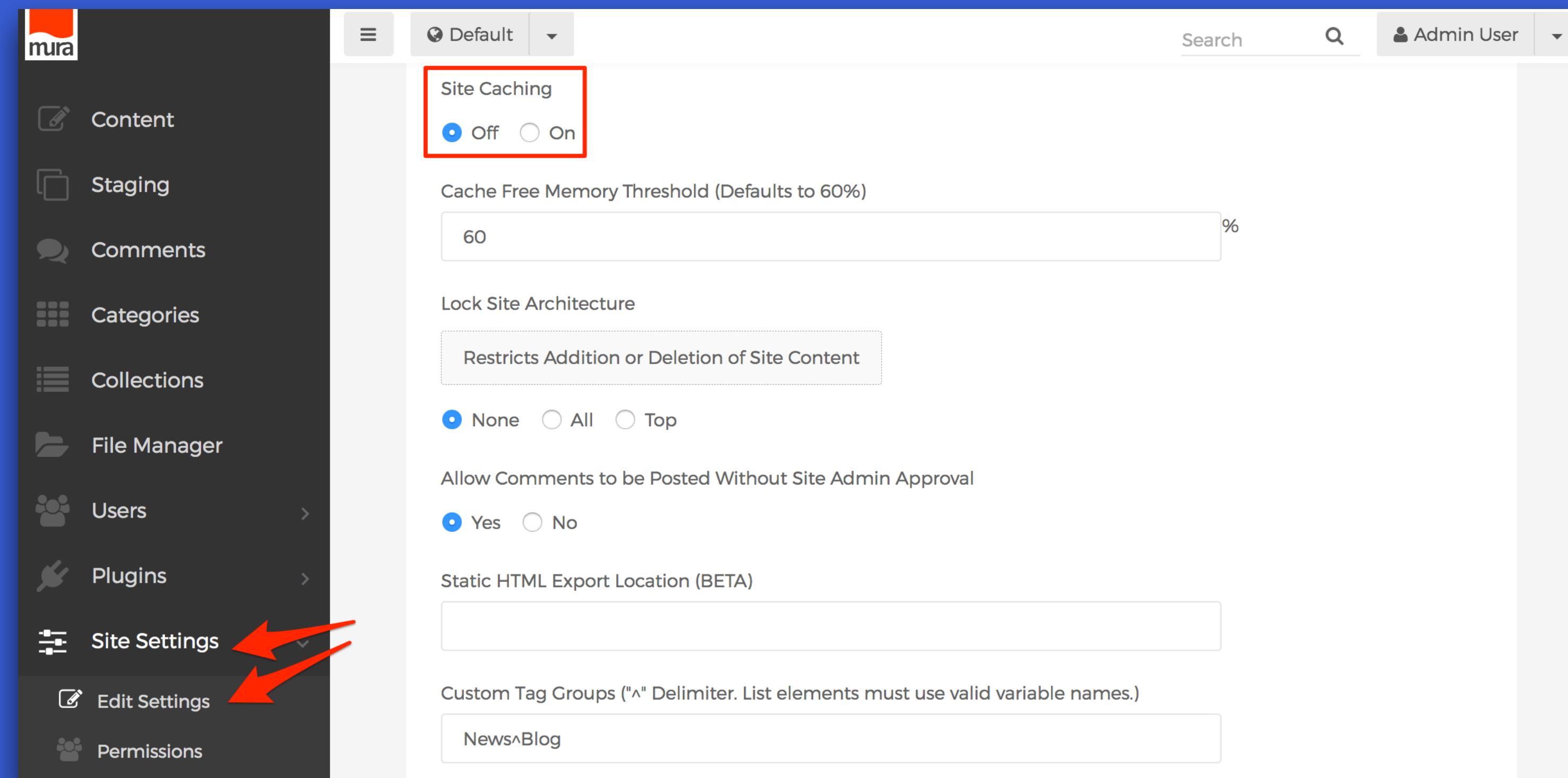
- Mura has a built-in caching mechanism that, when enabled, will store data in memory for reuse.
- The purpose is to improve overall site performance, by decreasing the amount of reads to the database.

Caching

- Mura has a built-in caching mechanism that, when enabled, will store data in memory for reuse.
- The purpose is to improve overall site performance, by decreasing the amount of reads to the database.
- Uses a “lazy load” approach, which means data won’t be cached until it’s called upon the first time.

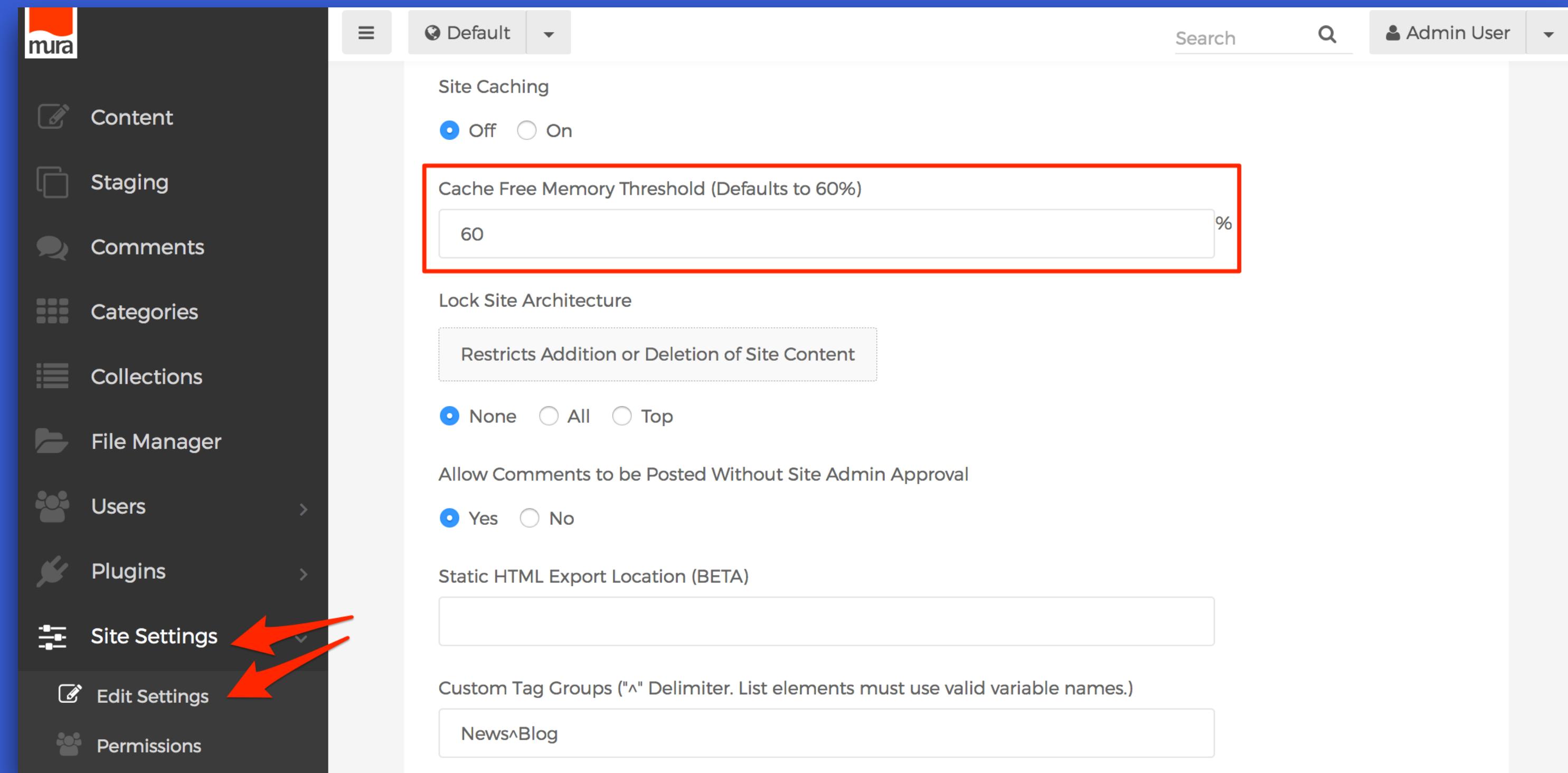
Enable/Disable Site Caching

- Enable/Disable Site Caching
 - Site Settings > Edit Settings > Site Caching



Memory Threshold

- Enable/Disable Site Caching
- Site Settings > Edit Settings > Site Caching



cf_CacheOMatic Tags

CACHING

cf_CacheOMatic Tags

- Allows you to add data to Mura's cache.

cf_CacheOMatic Tags

- Allows you to add data to Mura's cache.
- Only works when Site Caching is enabled.

cf_CacheOMatic Tags

- Allows you to add data to Mura's cache.
- Only works when Site Caching is enabled.
- You may specify the **key**, **timespan**, **scope**, and more.

cf_CacheOMatic Tags

- Allows you to add data to Mura's cache.
- Only works when Site Caching is enabled.
- You may specify the **key**, **timespan**, **scope**, and more.
- You may also use it specify NOT to cache the data!

cf_CacheOMatic Tags

```
<cf_CacheOMatic  
    key="{a unique key}"  
    nocache="{1=don't cache, 0=cache}"  
    timespan="{timespan object}">  
  
    <!-- Your data here. -->  
  
</cf_CacheOMatic>
```

cf_Cache0Matic Tags

```
<cf_Cache0Matic  
    key="#CGI.script_name##CGI.query_string#"  
    nocache="0"  
    timespan="#CreateTimeSpan(0,0,30,0)#">  
  
    <!-- Your data here. -->  
  
</cf_Cache0Matic>
```

cf_CacheOMatic Tags

- ./?purgeCache=true

cf_CacheOMatic Tags

- ./?purgeCache=true
- {context}/core/mura/customtags/CacheOMatic.cfm

cf_CacheOMatic Tags

- ./?purgeCache=true
- {context}/core/mura/customtags/CacheOMatic.cfm
- Or, you could also use CFML's more recent inline caching techniques

cf_CacheOMatic Tags

- ./?purgeCache=true
- {context}/core/mura/customtags/CacheOMatic.cfm
- Or, you could also use CFML's more recent inline caching techniques
 - ... **but** it won't be tied to Mura's Site Caching feature

Programmatic Caching

CACHING

Programmatic Caching

- See <https://docs.getmura.com/v7-1/mura-developers/mura-rendering/caching/programmatic-caching/>

Lab Exercise

LET'S RENDER!

Lab Exercise

- See `/3-core-developer/5-mura-rendering/instructions.md`

Summary

- The Mura contentRenderer.cfc
- The Mura Tag
- Mura Display Objects (Modules)
- Mura Content Types
- JSON API
- Mura.js
- Caching
- Lab Exercise

Mura Plugins

CREATING DISTRIBUTABLE APPLICATIONS

Mura Plugins

- Available Plugins
- Plugin Anatomy
- Installing Plugins
- Setting Permissions for Plugins
- Deleting Plugins
- Advanced Options
- Plugins vs. Modules

Mura Plugins

- Update-safe way to extend Mura CMS & add functionality

Mura Plugins

- Update-safe way to extend Mura CMS & add functionality
- Distribute code across sites & installations

Mura Plugins

- Update-safe way to extend Mura CMS & add functionality
- Distribute code across sites & installations
- Integrate existing applications

Mura Plugins

- Update-safe way to extend Mura CMS & add functionality
- Distribute code across sites & installations
- Integrate existing applications
- GPL 2.0 License

Mura Plugins

- Update-safe way to extend Mura CMS & add functionality
- Distribute code across sites & installations
- Integrate existing applications
- GPL 2.0 License
- **Custom Administrator***

Available Plugins

Available Plugins

- No “official” repository (yet)

Available Plugins

- No “official” repository (yet)
- Where to host your plugins?

Available Plugins

- No “official” repository (yet)
- Where to host your plugins?
- Popular plugins
 - See <http://www.getmura.com/downloads/mura-cms-plugins/>

Plugin Anatomy

Plugin Anatomy

- Mura plugins are distributed as “.zip” files

Plugin Anatomy

- Mura plugins are distributed as “.zip” files
- The .zip archive itself contains the underlying files which comprise the plugin

Plugin Anatomy

- Mura plugins are distributed as “.zip” files
- The .zip archive itself contains the underlying files which comprise the plugin
- All of your plugin’s directories and files must be packaged together under a common directory, and will ultimately be deployed under Mura as:
{context}/plugins/{YourPluginDirectoryName}/

Plugin Anatomy

- {context}/plugins/{YourPluginDirectoryName}
 - /plugin/
 - config.xml.cfm
 - plugin.cfc
 - config.cfm
 - index.cfm
 - license.txt
 - /content_types/
 - /modules/

Plugin Anatomy

- **Note:** By default, Mura does not automatically scan for a “model” directory within plugins.

Plugin Anatomy

- {context}/plugins/**MyPlugin/**
 - **plugin/**
 - **config.xml.cfm** (*the only required file*)
 - plugin.cfc
 - config.cfm
 - index.cfm
 - licence.txt
 - content_types/
 - modules/

Plugin Anatomy

- **Note:** By default, Mura does not automatically scan for a “model” directory within plugins. This is intentional to avoid any collisions, as your plugin may have its own which scans for this directory as well.

/plugin/config.xml.cfm

- **Setup Plugin Variables**
- **Optionally Register**
 - Event Handler(s)
 - Class Extensions
 - Custom Image Sizes
 - Custom Tag Paths
 - Custom Mappings
 - See <http://docs.getmura.com/v7-1/mura-developers/mura-plugins/plugin-anatomy/the-plugins-config-xml-cfm-file/> for more information

/plugin/config.xml.cfm

```
<cfoutput>
    <plugin>
        <name>Plugin1</name>
        <package>Plugin1</package>
        <directoryFormat>packageOnly</directoryFormat>
        <loadPriority>5</loadPriority>
        <version>1.0.0</version>
        <provider>Blue River Interactive</provider>
        <providerURL>http://blueriver.com</providerURL>
        <category>Application</category>
        <settings></settings>
        <eventHandlers></eventHandlers>
        <extensions></extensions>
    </plugin>
</cfoutput>
```

/plugin/config.xml.cfm

- Example <setting> node

```
<setting>
  <name>yourSettingName</name>
  <label>Your Label</label>
  <hint>Your Hint</hint>
  <type>text | radioGroup | textArea | select | multiSelectBox</type>
  <required>false | true</required>
  <validation>none | email | date | numeric | regex</validation>
  <regex>Your JavaScript regex (if validation=regex)</regex>
  <message>Your message if validation fails</message>
  <defaultValue>1</defaultValue>
  <optionlist>1^2^3</optionlist>
  <optionalabellist>One^Two^Three</optionalabellist>
</setting>
```

Lab Exercise

PLUGIN 1

/plugin/config.xml.cfm

- See <https://docs.getmura.com/v7-1/mura-developers/mura-plugins/plugin-anatomy/the-plugins-config-xml-cfm-file/> for more details.

/plugin/plugin.cfc

```
component accessors=true extends='mura.plugin.plugincfc' output=false {  
    // pluginConfig is automatically available as variables.pluginConfig  
  
    public void function install() {  
        // Do custom installation stuff  
    }  
  
    public void function update() {  
        // Do custom update stuff  
    }  
  
    public void function delete() {  
        // Do custom delete stuff  
    }  
}
```

Mura Scope

- **Bean Factory**
 - `m.getBean('user').loadBy(username='Steve')`
- **Mura Events**
 - `m.event()`, `m.announceEvent('yourEvent')`, etc.
- **Subscopes**
 - `m.content()`, `m.currentUser()`, `m.globalConfig()`, `m.siteConfig()`
- **Helpers**
 - `m.getPlugin()`, `m.getImageURL()`, etc.

/plugin/config.cfm

- Hooking Into Mura

```
<cfscript>

    if ( !IsDefined('m') ) {
        siteid = StructKeyExists(session, 'siteid') ? session.siteid : 'default';
        m = application.serviceFactory.getBean('m').init(siteid);
    }

    if ( !IsDefined('pluginConfig') ) {
        pluginConfig = m.getBean('pluginManager').getConfig('Plugin2');
    }

</cfscript>
```

```
pluggingConfig =  
m.getBean('pluginManager').getConfig('packageName')
```

```
.getDirectory()
```

```
.getPackage()
```

```
.getName()
```

```
.getApplication()
```

```
.getSession()
```

```
.getAssignedSites()
```

```
.getSettings()
```

```
.getSetting()
```

/index.cfm

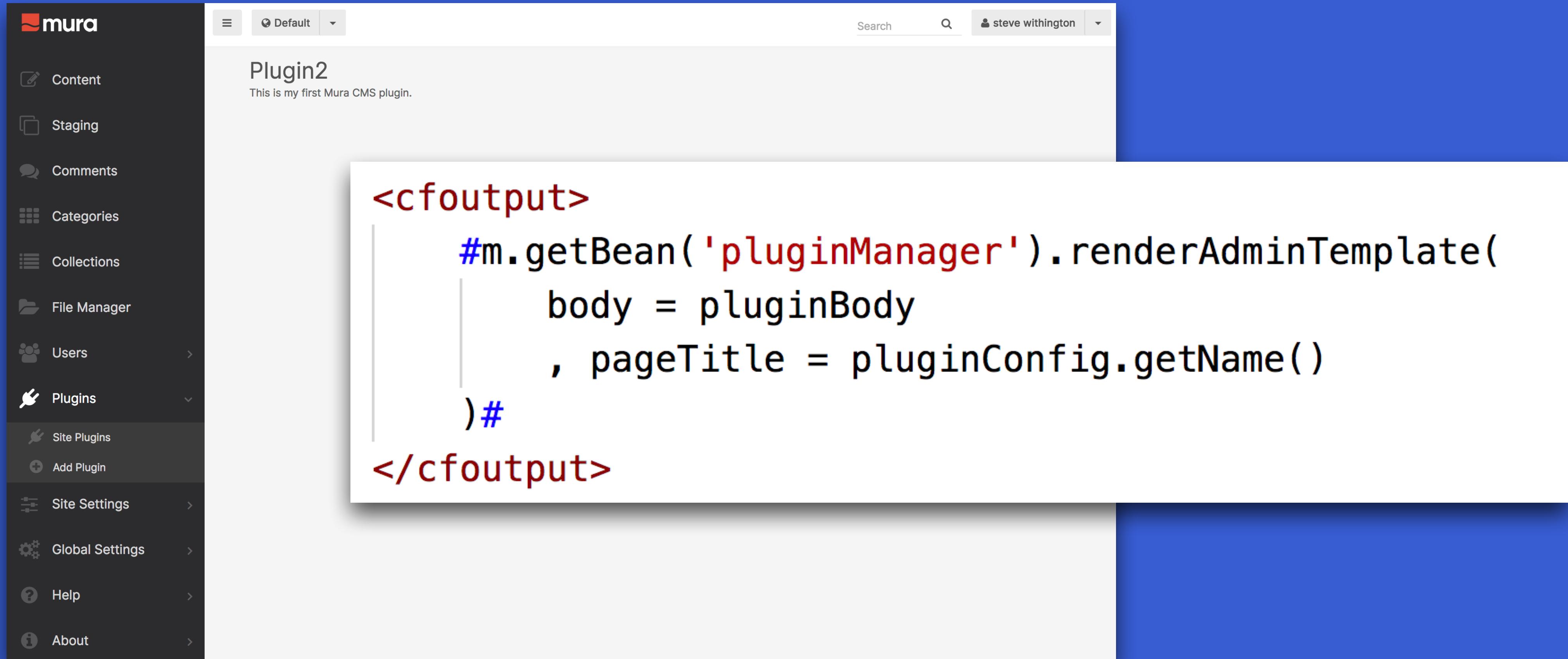
```
<cfinclude template="plugin/config.cfm" />

<cfsavecontent variable="pluginBody">
    <cfoutput>
        <h1>#esapiEncode('html', pluginConfig.getName())#</h1>
        <p>This is my first Mura CMS plugin.</p>
    </cfoutput>
</cfsavecontent>

<cfoutput>
    #m.getBean('pluginManager').renderAdminTemplate(
        body = pluginBody
        , pageTitle = pluginConfig.getName()
    )#
</cfoutput>
```

Admin Template

- Optionally use the Mura CMS look and feel



Lab Exercise

PLUGIN 2

/index.cfm

- Publicly Accessible

/index.cfm

- Publicly Accessible
- May Restrict Access (*if you want*)
 - Admin template does NOT control access!

/index.cfm

- Publicly Accessible
- May Restrict Access (*if you want*)
 - Admin template does NOT control access!
- Use Application.cfc (*if you want*)
 - <https://github.com/stevewithington/MuraPlugin>

Lab Exercise

[LOGOUT & NAVIGATE TO PLUGIN 2](#)

Mura Events

- Livecycle Events & Contextual Events

Mura Events

- Livecycle Events & Contextual Events
- Events can be intercepted to:
 - Provide *additional* logic
 - Or, even *replace* business logic

Event Handlers

- CFML Application Events
- Standard Events
- App-Related Events
- Content-Related Events
- User-Related Events
- Custom Events

Event Handlers

- Mura will NOT automatically look for event handlers like it does in the Site or Themes

Event Handlers

- Mura will NOT automatically look for event handlers like it does in the Site or Themes (unless they're within a module or content type's **model/handlers** directory)

Event Handlers

- Mura will NOT automatically look for event handlers like it does in the Site or Themes (unless they're within a module or content type's `model/handlers` directory)
- You must register in the `/plugin/config.xml.cfm` file.

Event Handlers

- Mura will NOT automatically look for event handlers like it does in the Site or Themes (unless they're within a module or content type's `model/handlers` directory)
- You must register in the `/plugin/config.xml.cfm` file.

```
<eventHandlers>

    <!-- This is the only eventHandler you need to register -->
    <eventHandler    event="onApplicationLoad"
                    component="model.handlers.eventHandler" />

</eventHandlers>
```

eventHandler.cfc

```
component accessors=true extends='mura.plugin.pluginGenericEventHandler' output=false {  
    property name='m' hint='mura scope';  
  
    public any function onApplicationLoad(required struct m) {  
        // Register all event handlers/listeners of this .cfc with Mura CMS  
        variables.pluginConfig.addEventHandler(this);  
        setm(arguments.m);  
    }  
  
    public any function onPageDefaultBodyRender(required struct m) {  
        // Dynamically alter the 'Title' of the page  
        m.content('title', 'Hijacked!');  
        // Override the output of the main body area of Page / Default  
        return '<h3>'#m.content('title')#</h3>' & m.content('body');  
    }  
  
    // You could place any other event handlers/listeners here  
}
```

Lab Exercise

PLUGIN 3

license.txt

Apache License
Version 2.0, January 2004
<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation

license.txt



Apache
Version 2.0,
<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Section 2.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the license.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation

Plugin Settings

Name: Plugin4 Category: Application Version: 4.0.0 Provider: Blue River Interactive Plugin ID: 5 Package: Plugin4

End User License Agreement

Apache License
Version 2.0, January 2004
<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

I Do Not Accept

UPDATE

Lab Exercise

PLUGIN 4

Modules

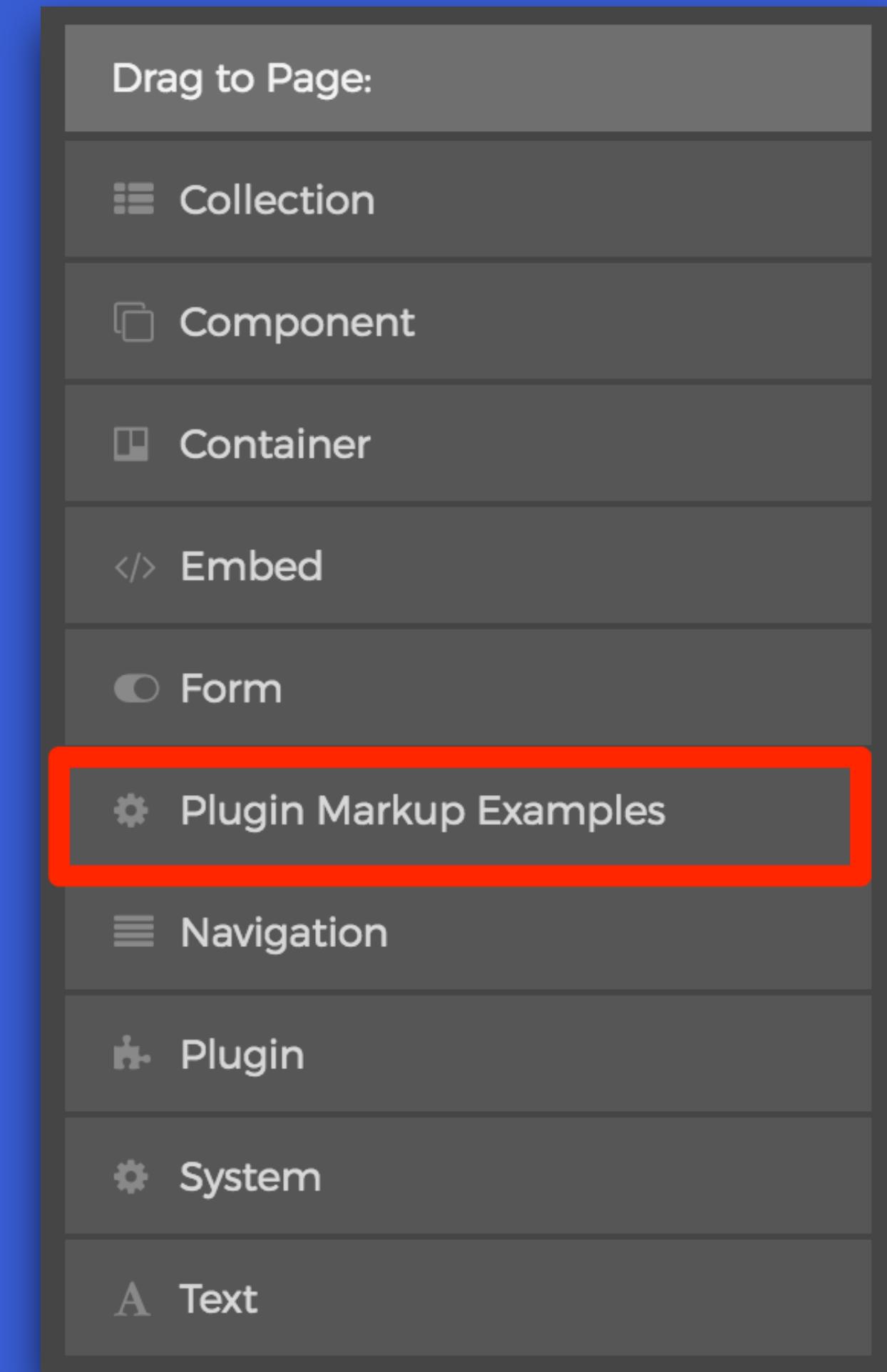
- /MyPlugin/
 - modules/

Modules

- /MyPlugin/
- modules/
- your_module_folder/
 - config.xml.cfm
 - index.cfm
 - configurator.cfm (*optional*)

Modules

- /MyPlugin/
- modules/
 - your_module_folder/
 - config.xml.cfm
 - index.cfm
 - configurator.cfm (*optional*)



Custom Content Types

- /MyPlugin/
 - content_types/

Custom Content Types

- /MyPlugin/
- content_types/
 - page/
 - index.cfm

Custom Content Types

- /MyPlugin/
- content_types/
 - page/
 - index.cfm
 - {type}_{subtype}/
 - index.cfm

Lab Exercise

PLUGIN 5

Installing Plugins

Installing Plugins

- Distributed as “**.zip**” files.

Installing Plugins

- Distributed as “**.zip**” files.
- Deployed via the administrator:
 - Via upload
 - Via URL

Installing Plugins

- Distributed as “`.zip`” files.
- Deployed via the administrator:
 - Via upload
 - Via URL
- Deployed via auto-discovery
 - `{context}/plugins/`

Installing Plugins

- Distributed as “`.zip`” files.
- Deployed via the administrator:
 - Via upload
 - Via URL
- Deployed via auto-discovery
 - `{context}/plugins/`
- Managed &/or deleted via administrator

Setting Permissions for Plugins

Setting Permissions for Plugins

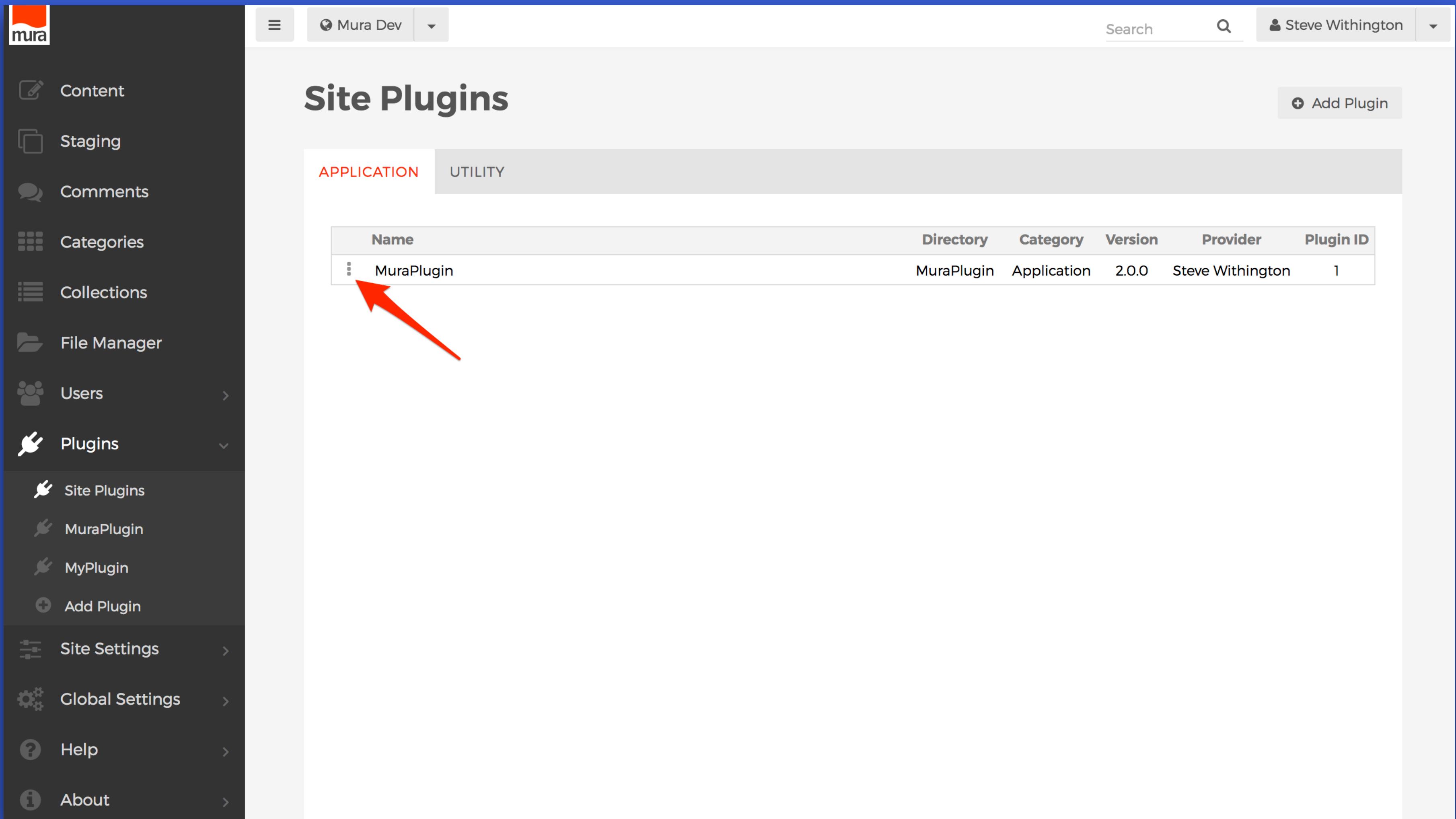
The screenshot shows the Mura CMS interface. The left sidebar contains a navigation menu with various options: Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Plugins, MuraPlugin, MyPlugin, Add Plugin, Site Settings, Global Settings, Help, and About. Two red arrows point from the text 'Setting Permissions for Plugins' to the 'Site Plugins' link in the sidebar.

Site Plugins

APPLICATION **UTILITY**

| Name | Directory | Category | Version | Provider | Plugin ID |
|------------|------------|-------------|---------|------------------|-----------|
| MuraPlugin | MuraPlugin | Application | 2.0.0 | Steve Withington | 1 |

Setting Permissions for Plugins



The screenshot shows the Mura CMS interface with the title "Site Plugins". On the left, there's a sidebar with various menu items like Content, Staging, Comments, etc. Under the "Plugins" section, "Site Plugins" is selected, showing entries for "MuraPlugin" and "MyPlugin". A red arrow points to the "MuraPlugin" row in the table.

| Name | Directory | Category | Version | Provider | Plugin ID |
|------------|------------|-------------|---------|------------------|-----------|
| MuraPlugin | MuraPlugin | Application | 2.0.0 | Steve Withington | 1 |

Setting Permissions for Plugins

The screenshot shows the Mura CMS interface with the title "Site Plugins". The left sidebar contains navigation links such as Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins (with Site Plugins, MuraPlugin, MyPlugin, and Add Plugin), Site Settings, Global Settings, Help, and About. The main content area displays a table titled "Site Plugins" with columns: Name, Directory, Category, Version, Provider, and Plugin ID. A single row is shown for "MuraPlugin" with values: MuraPlugin, Application, 2.0.0, Steve Withington, and 1. Below the table, there are tabs for APPLICATION and UTILITY, and a button for "Add Plugin". A red arrow points to the "Permissions" link in the "Edit" dropdown menu for the "MuraPlugin" row.

| Name | Directory | Category | Version | Provider | Plugin ID |
|------------|------------|-------------|---------|------------------|-----------|
| MuraPlugin | MuraPlugin | Application | 2.0.0 | Steve Withington | 1 |

Setting Permissions for Plugins

The screenshot shows the Mura CMS interface with the title "Permissions" displayed. On the left, a dark sidebar lists various site management options like Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings, and Web Services. The "Plugins" section is expanded, showing Site Plugins, MuraPlugin (which is selected), MyPlugin, and Add Plugin. The "Site Settings" section is also expanded, showing Edit Settings, Permissions, Approval Chains (which is checked), Class Extensions, Create Site Bundle, Deploy Site Bundle, and Web Services.

The main content area is titled "Permissions" and contains instructions: "To set permissions for MuraPlugin, check the box for each user group that should have access." It is divided into two sections: "System Groups" and "Member Groups".

System Groups: A section titled "Allow Group" with checkboxes for "Human Resources" and "Marketing".

Member Groups: A section titled "Allow Group" with a checkbox for "Employees". Below this, a note states: "These permissions can allow site members to edit content on the front-end of the site. Site members will *not* have access to the back-end administration area."

A blue "UPDATE" button is located at the bottom right of the main content area.

Setting Permissions for Plugins

The screenshot shows the Mura CMS interface for setting permissions. The left sidebar contains navigation links such as Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins (with Site Plugins, MuraPlugin, MyPlugin, and Add Plugin), and Site Settings (with Edit Settings, Permissions, Approval Chains, Class Extensions, Create Site Bundle, Deploy Site Bundle, and Web Services). The main content area is titled "Permissions" and displays instructions: "To set permissions for MuraPlugin, check the box for each user group that should have access." It shows two sections: "System Groups" and "Member Groups". In the "System Groups" section, there is a table with "Allow Group" rows for "Human Resources" (unchecked) and "Marketing" (checked, indicated by a red arrow pointing to it). In the "Member Groups" section, there is a table with "Allow Group" rows for "Employees" (unchecked). A blue "UPDATE" button is located at the bottom right.

To set permissions for MuraPlugin, check the box for each user group that should have access.

System Groups

| Allow Group |
|---|
| <input type="checkbox"/> Human Resources |
| <input checked="" type="checkbox"/> Marketing |

Member Groups

These permissions can allow site members to edit content on the front-end of the site. Site members will *not* have access to the back-end administration area.

| Allow Group |
|------------------------------------|
| <input type="checkbox"/> Employees |

UPDATE

Setting Permissions for Plugins

The screenshot shows the Mura CMS interface for setting permissions for a plugin. The left sidebar contains navigation links for Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins (with Site Plugins, MuraPlugin, MyPlugin, and Add Plugin), and Site Settings (with Edit Settings, Permissions, Approval Chains, Class Extensions, Create Site Bundle, Deploy Site Bundle, and Web Services). The main content area is titled "Permissions" and includes a note: "To set permissions for MuraPlugin, check the box for each user group that should have access." It shows two sections: "System Groups" and "Member Groups". In "System Groups", under "Allow Group", "Marketing" is checked. In "Member Groups", under "Allow Group", "Employees" is unchecked. A red arrow points to the "UPDATE" button at the bottom.

To set permissions for MuraPlugin, check the box for each user group that should have access.

System Groups

Allow Group

Human Resources
 Marketing

Member Groups

These permissions can allow site members to edit content on the front-end of the site. Site members will *not* have access to the back-end administration area.

Allow Group

Employees

UPDATE

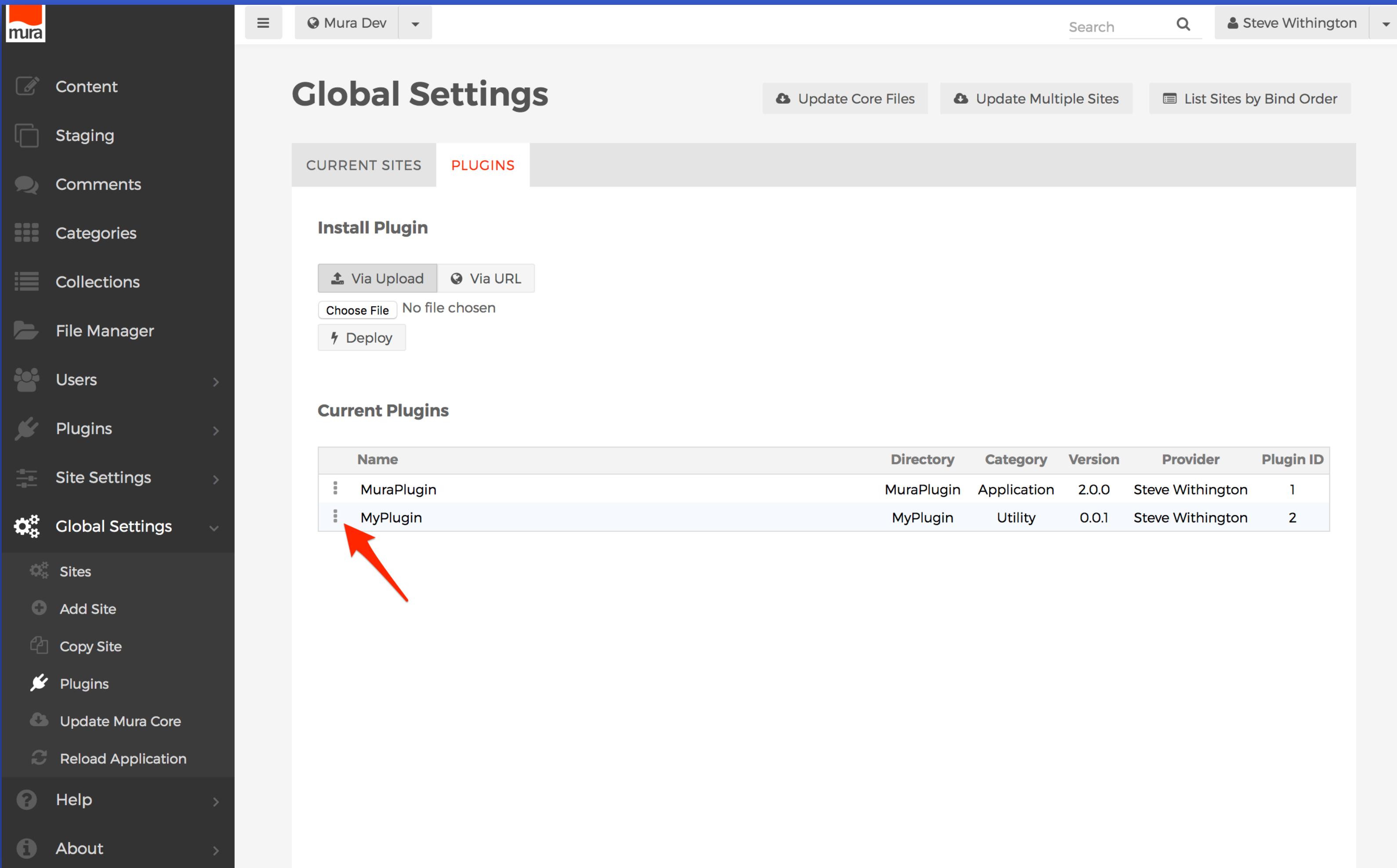
Deleting Plugins

Deleting Plugins

The screenshot shows the Mura Global Settings page. On the left, a sidebar menu includes options like Content, Staging, Comments, Categories, Collections, File Manager, Users, Plugins, Site Settings, Global Settings, Sites, Add Site, Copy Site, Plugins, Update Mura Core, Reload Application, Help, and About. Two red arrows point from the 'Plugins' option in the sidebar to the 'Plugins' tab in the main content area. The main content area has tabs for CURRENT SITES and PLUGINS, with PLUGINS selected. It features an 'Install Plugin' section with 'Via Upload' and 'Via URL' buttons, and a 'Choose File' input field showing 'No file chosen'. Below this is a 'Deploy' button. The 'Current Plugins' section contains a table:

| Name | Directory | Category | Version | Provider | Plugin ID |
|------------|------------|-------------|---------|------------------|-----------|
| MuraPlugin | MuraPlugin | Application | 2.0.0 | Steve Withington | 1 |
| MyPlugin | MyPlugin | Utility | 0.0.1 | Steve Withington | 2 |

Deleting Plugins



The screenshot shows the Mura Global Settings interface. On the left is a dark sidebar with various site management options. The main area is titled "Global Settings" and has tabs for "CURRENT SITES" and "PLUGINS". The "PLUGINS" tab is selected, showing the "Install Plugin" section with upload options and a current list of installed plugins.

Current Plugins

| Name | Directory | Category | Version | Provider | Plugin ID |
|------------|------------|-------------|---------|------------------|-----------|
| MuraPlugin | MuraPlugin | Application | 2.0.0 | Steve Withington | 1 |
| MyPlugin | MyPlugin | Utility | 0.0.1 | Steve Withington | 2 |

Deleting Plugins

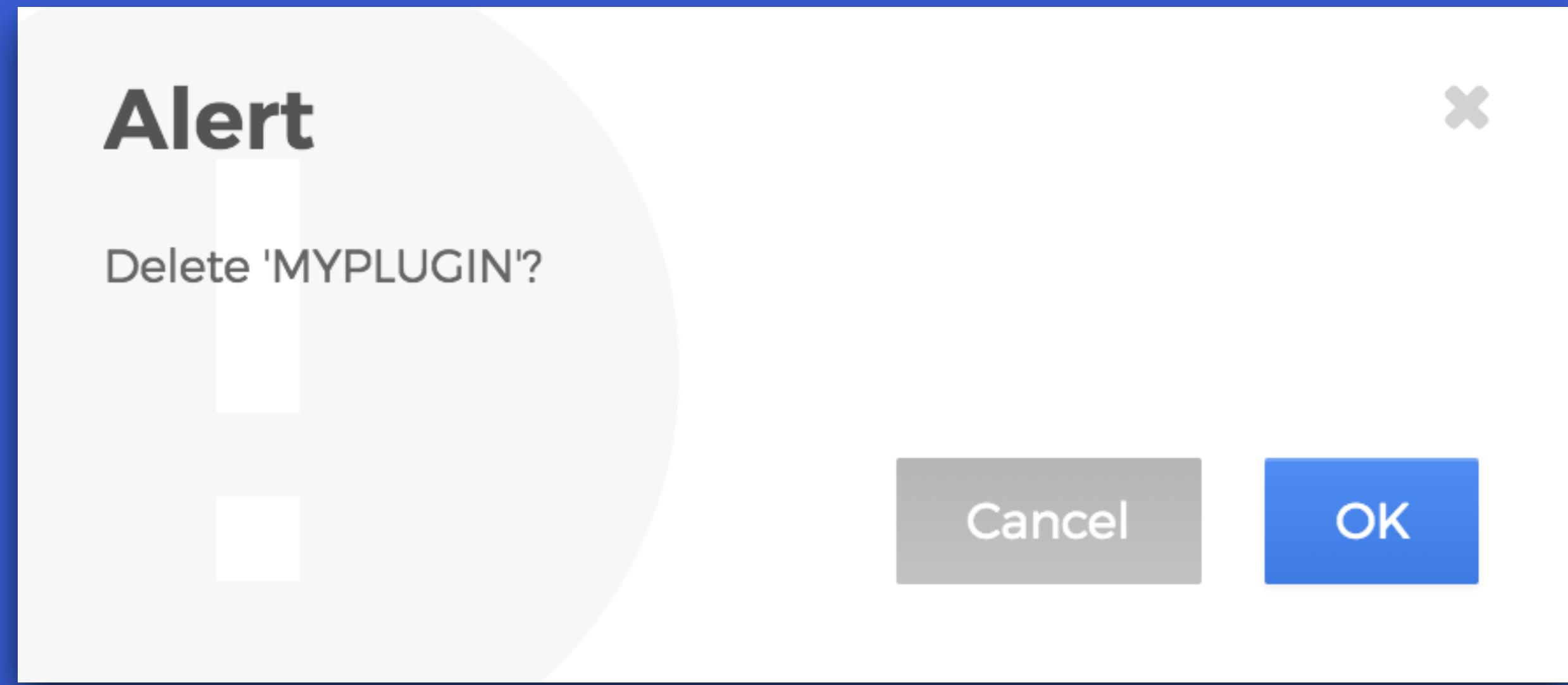
The screenshot shows the Mura Global Settings interface. On the left is a dark sidebar with various site management options. The main area is titled "Global Settings" and has tabs for "CURRENT SITES" and "PLUGINS". Under "PLUGINS", there's an "Install Plugin" section with upload and URL import options, and a "Current Plugins" table.

The "Current Plugins" table lists two entries:

| Name | Directory | Category | Version | Provider | Plugin ID |
|------------|------------|-------------|---------|------------------|-----------|
| MuraPlugin | MuraPlugin | Application | 2.0.0 | Steve Withington | 1 |
| MyPlugin | MyPlugin | Utility | 0.0.1 | Steve Withington | 2 |

A context menu is open over the first row ("MuraPlugin"). The menu items are "Edit" and "Delete". A red arrow points to the "Delete" option.

Deleting Plugins



Advanced Options

Advanced Options

- Endless possibilities
- Mura tools to accomplish pretty much anything you want:
 - Mura Scope
 - Mura Events
 - Mura Beans & Objects
 - Mura Rendering
 - CFML Frameworks
 - <https://github.com/stevewithington/MuraFW1>

Plugins vs. Modules

Plugins vs. Modules

- Need a custom administrator or need to restrict access to the administrator?

Plugins vs. Modules

- Need a custom administrator or need to restrict access to the administrator?
- **Yes**
 - Then you'll want to create a plugin. Each plugin can be restricted to specific sites, and also includes the ability to restrict which group(s) have access to the plugin's administrator area

Plugins vs. Modules

- Need a custom administrator or need to restrict access to the administrator?
 - **Yes**
 - Then you'll want to create a plugin. Each plugin can be restricted to specific sites, and also includes the ability to restrict which group(s) have access to the plugin's administrator area
 - **No**
 - Then creating a module is perfectly fine

Lab Exercise

- See `/3-core-developer/6-mura-plugins/instructions.md`

Summary

- Available Plugins
- Plugin Anatomy
- Installing Plugins
- Setting Permissions for Plugins
- Deleting Plugins
- Advanced Options
- Plugins vs. Modules

Where to Go From Here

HINT: BE A JEDI.

Getting Support

- The Mura Community (Google Group & LinkedIn)
- Issues
- MuraCon
- Training (*obviously*)
- Support

What We Covered

- Introduction
- Where Mura is Installed
- Mura Directory Structure
- Mura Scope
- Mura Events
- Mura Beans & Objects
- Mura Rendering
- Mura Plugins
- Where to Go From Here
- Getting Support

Thanks,

YOU ROCK. NOW, GO BUILD SOMETHING AWESOME.