Doc Type Grid Editor

Developers Guide

Contents

Introduction

Doc Type Grid Editor is an advanced grid editor for the new Umbraco grid, offering similar functionality as the macro grid editor but using the full power of the doc type editor and data types.

With the macro grid editor you are limited to only using the macro builder and thus the handful of parameter editors that are available. Of course you can create / config your own parameter editors, however this is cumbersome compared to how we can configure data types.

With the **Doc Type Grid Editor** then, we bridge that gap, allowing you to reuse doc type definitions as blue prints for complex data to be rendered in a grid cell.

Getting Set Up

System Requirements

Before you get started, there are a number of things you will need:

1. .NET 4.5+
2. Umbraco 7.2.x
3. The **Doc Type Grid Editor** package installed

Configuring The Doc Type Grid Editor

The **Doc Type Grid Editor** is configured via the grid.editors.config.js config file located in the */Config* folder. A default configuration should be installed along with the package, but for details on the configuration options, please see below.

Example:

```
1.  [
2.      ...
3.      {
4.          “name”: ”Doc Type”,
5.          “alias”: ”docType”,
6.          “view”: ”/App_Plugins/../doctypegrideditor.html”,
7.          “render”: ”/App_Plugins/../doctypegrideditor.cshtml”,
8.          “icon”: ”icon-item-arrangement”,
9.          “config”: {
10.             “allowedDocTypes: [...],
11.             “enablePreview”: true,
12.             “viewPath”: “/Views/Partials/”
13.     },
14.     ...
15. ]
```

For the main part, the root properties shouldn’t need to be modified, however the only properties that MUST not be changed are the **view** and **render** properties.

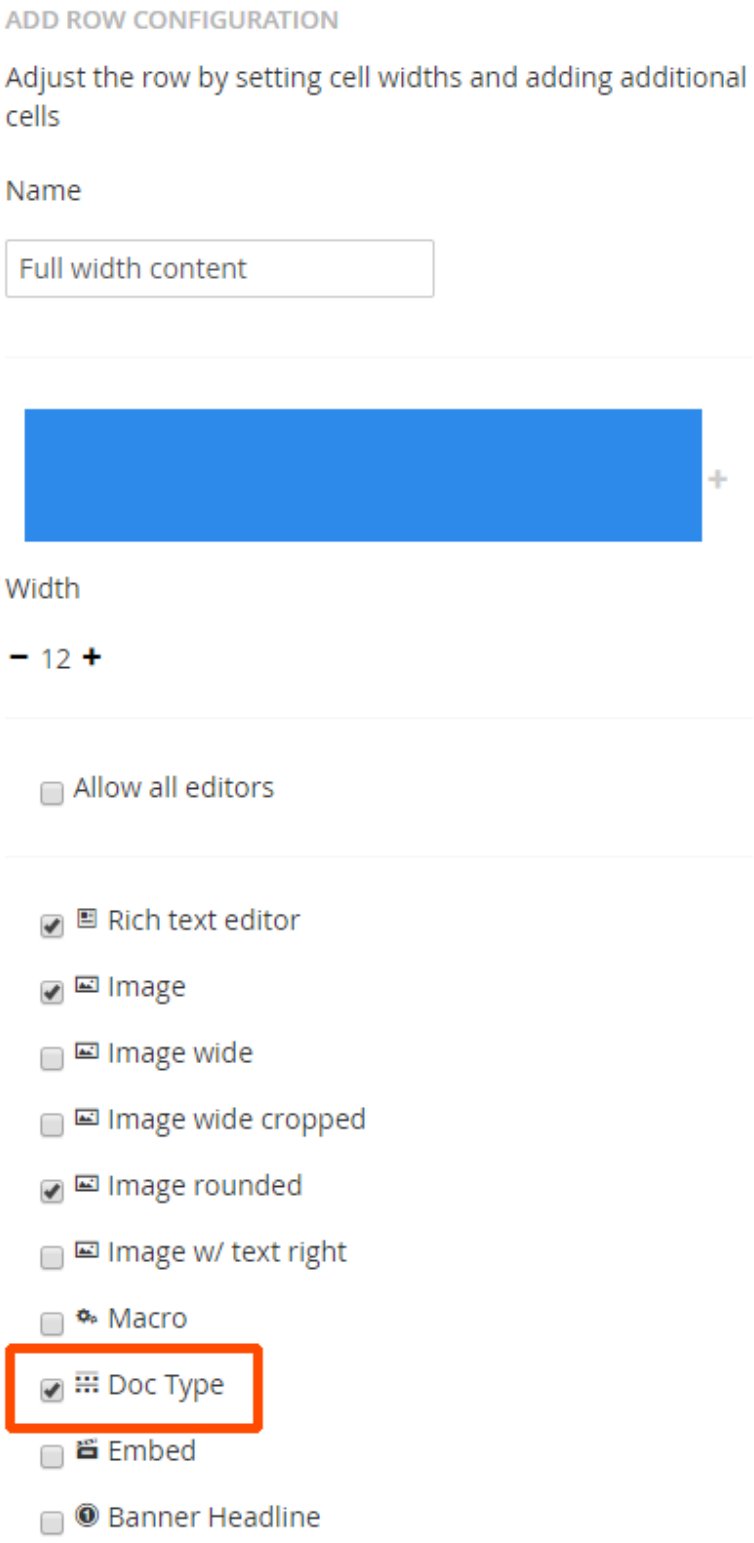
Member	Type	Description
Name	String	The name of the grid editor as it appears in the grid editor prevalue editor / selector screen.
Alias	String	A unique alias for this grid editor.
View	String	The path to the **Doc Type Grid Editor** editor view. **MUST NOT BE CHANGED.**
Render	String	The path to the **Doc Type Grid Editor** render view. **MUST NOT BE CHANGED.**
Icon	String	The icon class name to use for this grid editor (minus the ‘.’)
Config	Object	Config options for this grid editor.

The Doc Type Grid Editor supports 3 config options, all of which are optional.

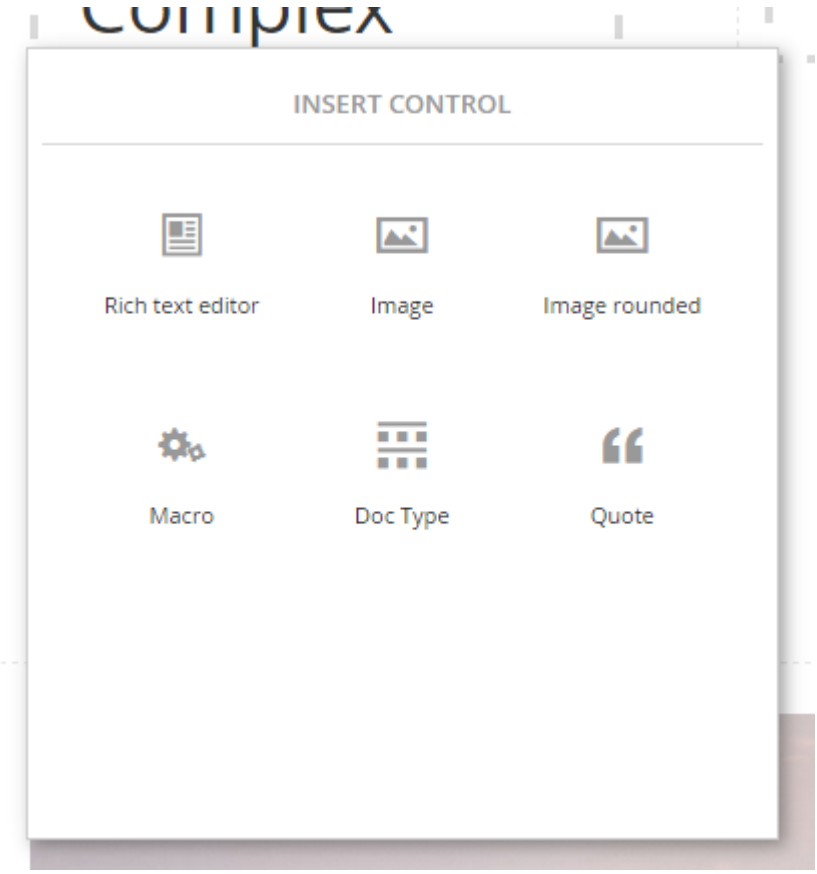
Member	Type	Description
AllowedDocTypes	String[]	An array of doc type aliases of which should be allowed to be selected in the grid editor. Strings can be REGEX patterns to allow matching groups of doc types in a single entry. Ie “Widget$” will match all doc types with an alias ending in “Widget”.
EnablePreview	Boolean	Enables rendering a preview of the grid cell in the grid editor.
ViewPath	String	Set’s an alternative view path for where the **Doc Type Grid Editor** should look for views when rendering. Defaults to */Views/Partials/*

Hooking Up The Doc Type Grid Editor

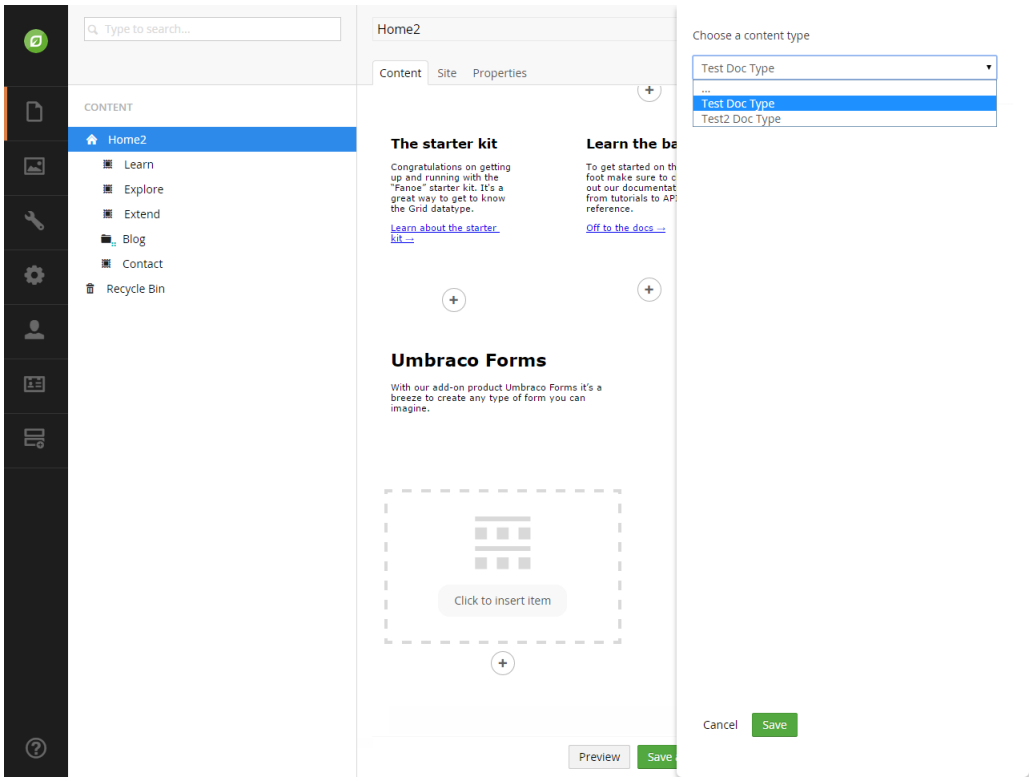
To hook up the **Doc Type Grid Editor**, within your grids prevalue, select the row configs you want to use the **Doc Type Grid Editor** in and for each cell config, check the **Doc Type** checkbox option to true. (**NB**: If you changed the name in the config, then select the item with the name you enter in the config).



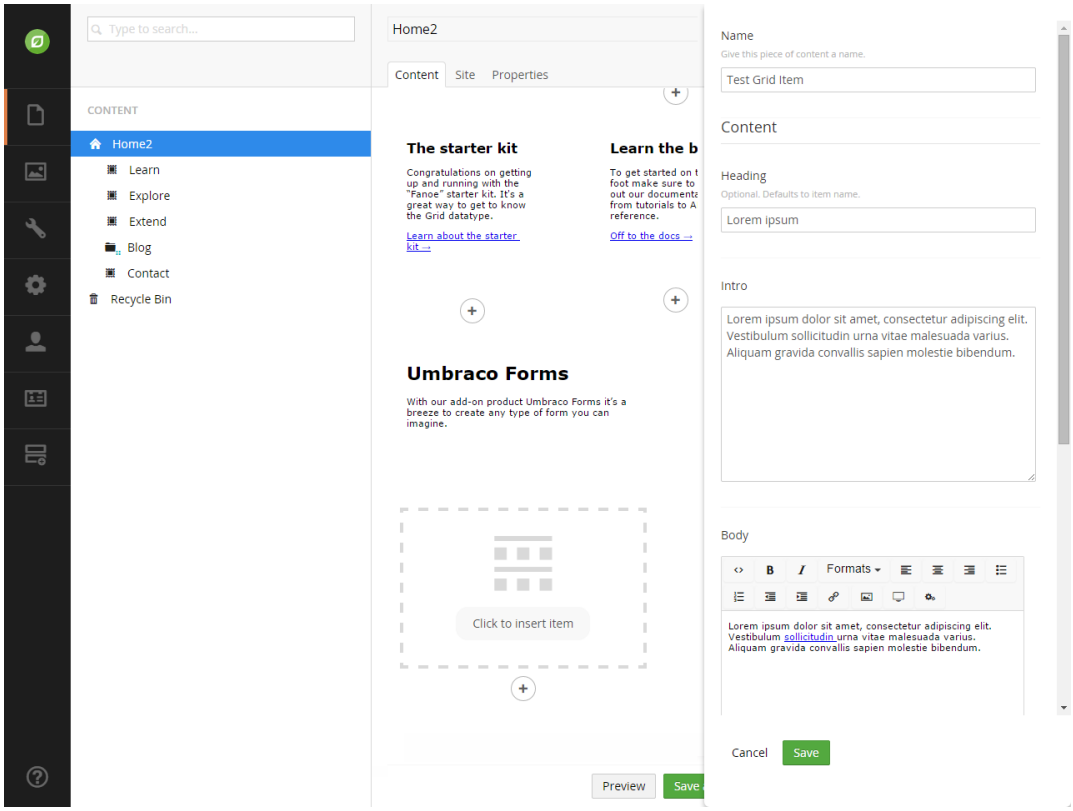
With the Doc Type Grid Editor enabled, from within your grid editor, you should now have a new option in the **Insert Control** dialog.



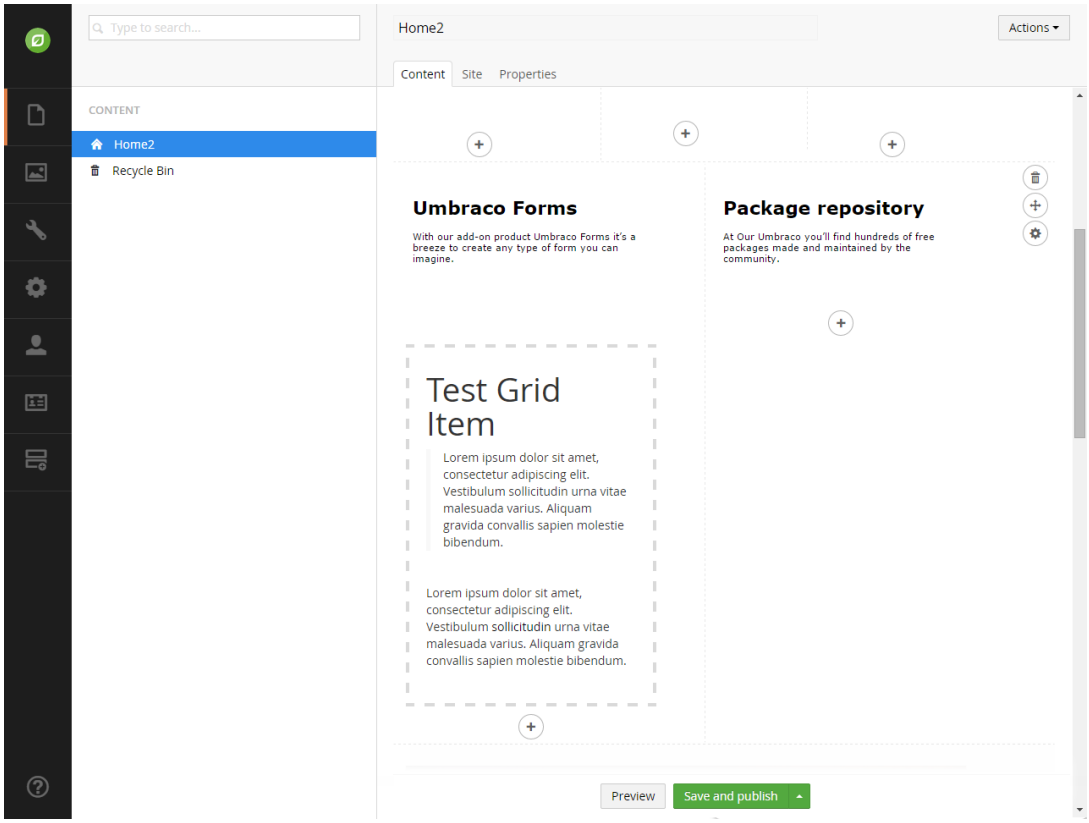
From there, simply click the **Doc Type** icon and chose the doc type you wish to render.



Then you should be presented with a form for all the fields in your doc type.



Fill in the fields and click save. You should then see the grid populated with a preview of your item.



Make sure save / save & publish the current page to persist your changes.

Rendering a Doc Type Grid Editor

The **Doc Type Grid Editor** uses standard ASP.NET MVC partials as the rendering mechanism. By default it will look for partial files in */Views/Partials* with a name that matches the doc type alias. For example, if your doc type alias is *TestDocType*, the **Doc Type Grid Editor** will look for the partial file */Views/Partials/TestDocType.cshtml.*

To access the properties of your completed doc type, simply have your partial view inherit the standard *UmbracoViewPage* class, and you'll be able to access them via the standard *Model* view property as a native *IPublishedContent* instance.

Example:

```
1.  @inherits Umbraco.Web.Mvc.UmbracoViewPage
2.  <h3>@Model.Name</h3>
```

Because we treat your data as a standard IPublishedContent entity, that means you can use all the property value converters you are used to using, aswell as the build in *@Umbraco.Field(…)* helper methods.

Example:

```
1.  @inherits Umbraco.Web.Mvc.UmbracoViewPage
2.  <h3>@Model.Name</h3>
3.  @Umbraco.Field(Model, "bodyText")
4.  <a
    href="@(Model.GetPropertyValue<IPublishedContent>("link").Url)">
    More</a>
```

DocTypeGridEditorSurfaceController

If you are not the type of developer that likes to put business logic in your views, then the ability to have a controller for you partial view is a must. To help with this, the **Doc Type Grid Editor** comes with a base surface controller you can used called *DocTypeGridEditorSurfaceController*.

Simply create your controller inheriting from the above class, giving it a class name of *{DocTypeAlias}SurfaceController* and an action name of *{DocTypeAlias}* and the **Doc Type Grid** Editor will automatically wire it up for you and use it at render time.

Example:

```
1.  public class TestDocTypeSurfaceController
        : DocTypeGridEditorSurfaceController
2.  {
3.      public ActionResult TestDocType()
4.      {
5.          // Do your thing...
6.          Return CurrentPartialView();
7.      }
8.  }
```

By inheriting from the *DocTypeGridEditorSurfaceController* base class, you'll also have instant access to the following helper properties / methods.

Member	Type	Description
Model	IPublishedContent	The IPublishedContent instance for you cells data.
ViewPath	String	A reference to the currently configured ViewPath
CurrentPartialView(object model = null)	Method	Helper method to return you to the default partial view for this cell. If no model is passed in, the standard *Model* will be passed down.
PartialView(string viewName, object model = null)	Method	Helper method to return you to an alternative partial view for this cell. If no model is passed in, the standard *Model* will be passed down.

Useful Links

- **Source**
  https://github.com/mattbrailsford/doc-type-grid-editor

- **Our Umbraco Project Page**
  http://our.umbraco.org/projects/backoffice-extensions/doc-type-grid-editor