Css

```
.myClass{width:10px;height:20px;}

#myVideo {
    width: 100%; /* width needs to be set to 100% */
    height: 100%; /* height needs to be set to 100% */
    position: absolute;
    left: 0;
    top: 0;
    z-index: -1;

}

.videocontent {
    position: fixed;
    bottom: 0;
    background: rgba(0, 0, 0, 0.5);
    color: #fff1f1;
    width: 100%;
    padding: 20px;
}
```

CONTENTFUL, ADDON DEVELOPMENT, TECH TALK, CMS

Getting started with Contentful UI Extensions - part 2

Allan Thraen | 5 Months Ago | PDF | 0

In this post, I'll show how to make a field editor that will let you have any kind of syntax highlighted code in a long text field, as well as taking a look at command line interface (CLI) and Github distribution.

This is the second post in the series based on a presentation I gave at a recent Contentful meetup on how to create UI Extensions. Read the first one here. In the first post, we had a look at the different kinds of UI extensions - and how you can create a field extension that changes the apperance of a field to a color editor. Now, we'll examine another example of a field extension - that takes it one step further...

Example: Code Editor

When I first started playing around with Contentful, I - as many others - quickly started to think: This is a nice way to work with content, but how do I manage the content delivery to my web channel? Obviously there are many approaches to this (and I'll probably go through some + a few new ideas in a future post) - but one of my thoughts was that it would be kind of cool if I could manage even the code/templating aspect of web delivery, directly inside Contentful - as Content. This includes both Css, Javascript and something that can help generate the Html - and my choice of weapon is typically Razor. Having code as content is something I have experimented with in the past - and even though it's not always as nice to work in a Browser as in a full-blown IDE, it does seem pretty practical for rapid prototyping.

In order to make the experience even smoother I decided to incorporate a UI extension that would give me a syntax highlighted code editor in my browser, which is the example I'm about to show.

After achieving this I was obviously incredibly proud - a feeling that lasted a few days until I accidentally noticed that there was already a pretty identical example in the official UI Extensions Samples repository that I hadn't bothered to look through first :-)

But anyway, here is the code I used - it's far from perfect, but it does give an impression of what I'm trying to achieve:

```
<!DOCTYPE html>
<html>
  <head>
<script src="https://unpkg.com/contentful-ui-extensions-sdk@3"></script>

<!-- Load the ACE code editor -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/ace/1.4.3/ace.js"></script>

                                        #editor {
                                            position: absolute;
                                            top: 0;
                                            right: 0;
                                            bottom: 0;
                                            left: 0;
    <style type="text/css" media="screen">    }                    </style>
</head>
<body>
<div id="editor"></div>

        window.contentfulExtension.init(function(api) {
          //Get the current field value
          var val=api.field.getValue();

          //Initialize the editor
          var editor = ace.edit("editor");

          //Set editor theme
          editor.setTheme("ace/theme/monokai");

          //Set which mode the editor should be in (razor, csharp, css, html, javascript, .
          //This fetched from an instance parameter
          editor.session.setMode("ace/mode/"+api.parameters.instance.mode);

          //If the value is defined, set it.
          if(val!=undefined)  editor.setValue(val,-1);

          //When the editor leaves the field, update the value
          editor.on("blur", function(){   api.field.setValue(editor.getValue());    });
```

Related

- ACE Code Editor
- First post in the series Getting started with Contentful UI
- Extensions - Part 3

Blog posts

2019
November
October
September
August
July
June
May
April
March
February
January
2018
December
November
October
September
August
June

Available Topics

.NET DEVELOPMENT
ADDON DEVELOPMENT
AI
API BUILDING
AZURE
BEHAVIOR ANALYTICS
BIG DATA
C#
CMS
CONTENTFUL
DIGIZUITE
ELASTICSEARCH
EPISERVER
FREELANCING
INFORMATION RETRIEVAL
INTEGRATIONS
LIFE AS A CODER
MICRO SERVICES
OFFSHORE DEVELOPMENT
PRODUCT MANAGEMENT
TECH TALK
TIPS AND TRICKS
VISION DEMOS & PROTOTYPES

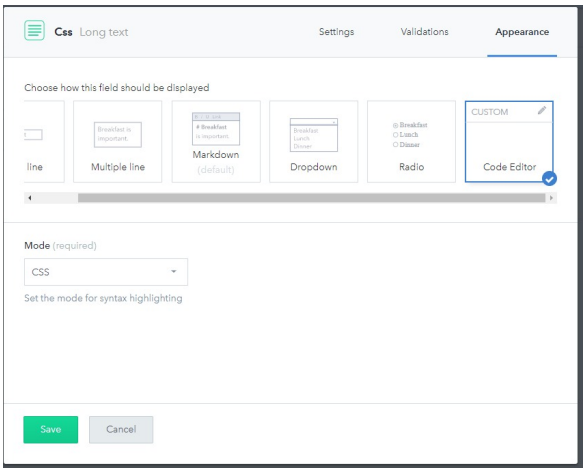
About Allan

I'm a freelance software architect / developer and Episerver expert (EMVP) based in Copenhagen Denmark. I've worked for Episerver for more than 10 years before going freelance and understand both your business needs as well as the technical possibilities.

MVP MOST VALUABLE PROFESSIONAL

```
        //Ensure proper height in our window
        api.window.updateHeight(500);
<script>});
</body>
</html>
```

As you can see in the code and the comments, one key new thing is that we are retrieving the 'Mode' for the code editor from an Instance Parameter. Instance parameters are parameters set on the specific instance of the UI Extension, when used on a content type. In this case I have defined a Mode and it shows as a Dropdown list when the Code Editor is selected as the Apperance for a long text.



To achieve this, we have to not only create the HTML file for the extension, but also a Json manifest / definition for the UI Extension. You can see it here:

```
{
  "id": "4esx4JW2CYsu14ePjk21IP",
  "name": "Code Editor",
  "srcdoc": "codeeditor.html",
  "fieldTypes": ["Text"],
  "parameters": {
    "instance": [
{       "id": "mode",
        "name": "Mode",
        "description": "Set the mode for syntax highlighting",
        "type": "Enum",
        "options": [{"javascript": "Javascript"}, {"csharp": "csharp"}, {"html": "HTML"},
        "labels": {"empty": "Choose the display mode"},
        "required": true}
    ]
  }
}
```

Notice how I define the instance parameter and provide options for the drop down in the json definition.

Command line interface

This does however spark a new issue - you can't edit the definition json through the web interface. So instead you have to use the CLI.

For those accustomed to NodeJS and NPM, this is business as usual. For others, like me, that is more Visual Studio/.NET / Nuget oriented it's still fairly easy.

First, to get setup with the CLI in the first place, follow these steps here:
https://github.com/contentful/contentful-cli

Then, after everything is installed, you'll want to log in, using the command "contentful login" and select a space ("contentful space" command) after which you for instance can have a look at which extensions are installed, calling "contentful extension list" as I've done here.

```
C:\GitHub\ContentfulExtensions>contentful extension list
```

Extension Name	Extension ID	Version
Code Editor	4esx4JW2CYsu14ePjk21IP	29
CP	2Z8kSqFwIJqOThVLWIpVtJ	14
CreationTest	2X6UVgjrdnQxY2IgZzYt6n	9
Named Entity Extraction	jpqCbiqlkpC4uKiiVSiOl	16
Stock Photos	1iEUaY1h6Z4KRXx6cNwsnz	75
TreeStructure	6RJyjLtgNZeSUUY4RacZxl	2

You can see the details of the 'extension' command here:
https://github.com/contentful/contentful-cli/tree/master/docs/extension.

So, once your CLI is ready, you can basically send a local extension to your space by calling "contentful extension create --descriptor .\codeeditor.json", or - if it already exists in your space update it using "contentful extension update ...".

You can also use the "--src" option to provide a url for a selfhosted extension.

```
C:\GitHub\CodeArt.ContentfulExamples\UIExtensions\Code Editor>contentful extension create --descriptor .\codeeditor.json
 Successfully created extension:
Space: mx3buqkca631
Environment: master
Your extension: https://app.contentful.com/spaces/mx3buqkca631/settings/extensions/codeEditor
```

Property	Value
ID	codeEditor
Name	ACE Code Editor
Field types	Text
Src	[user_srcdoc]

I love solving real business problems with innovative and creative coding!

Learn more about my skills and specialities, reach out to me at info@codeart.dk or use this form to contact me.

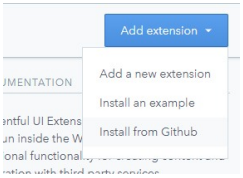
Recommended for you

- CodeArt
- Ascend 2019: Code Mania
- Error: No parameterless constructor defined for this object

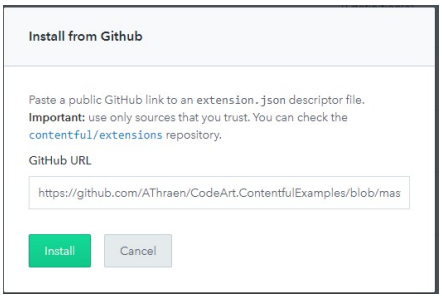
After running this command our extension is available in our space - ready for use, including the custom parameters!

Using Github to distribute extensions

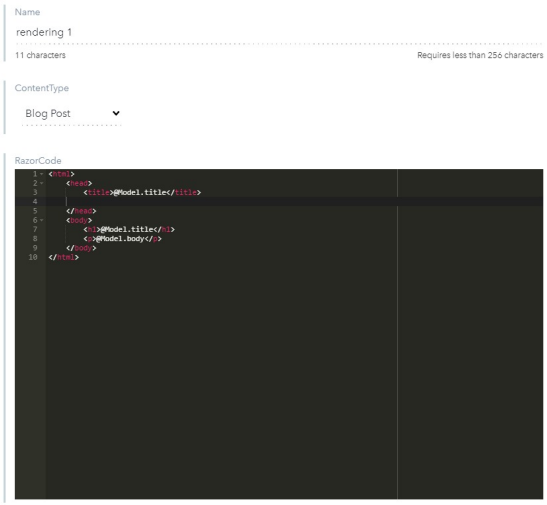
If you want to quickly make your extension available to the masses, you can put your extension files in a public GitHub repository. In this case it's important that the definition json file is called "extension.json" as there is a validation check for that. Then, all that's needed to install it is for someone to select "Install from Github" when they add a new extension.



Followed by providing the Url to the specific definition file. In this case I've checked the Code Editor into this repository:
https://github.com/AThraen/CodeArt.ContentfulExamples/blob/master/UIExtensions/Code%20Editor/extension.json



Once the extension has been successfully installed we can go to our Content Modelling, add a new (long) text field and select it under "Appereances". Remember to select the mode as well!



That's it for this time. In the next post in this series we'll take a look at sidebar & dialog extensions and just how much power is available in the javascript SDK.

CONTENTFUL ADDON DEVELOPMENT TECH TALK CMS

Previous Post
Getting started with Contentful UI Extensions - Part 1

Next Post
Getting started with Contentful UI Extensions - Part 3

Post Comments (0)

Phone: +45 26 13 66 96
CVR: 39680688