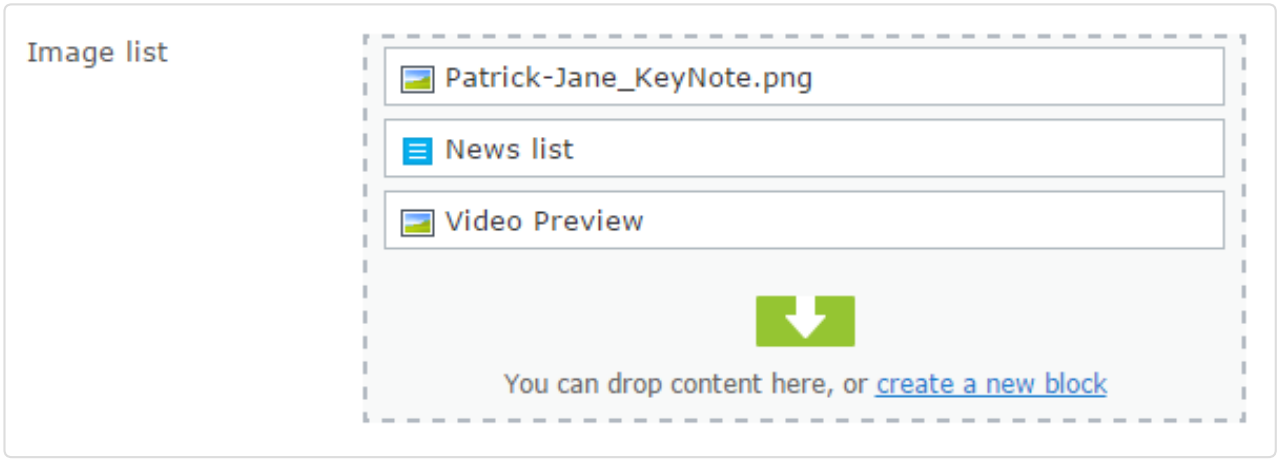
Content Area as Image list

15 April 2015 EPiServer (https://gregwiechec.com/category/episerver/), EPiServer properties (https://gregwiechec.com/category/episerver-properties/)

We have a lot of list properties in my current project. And most of them are edited entirely in the forms view which means that the editors are not able to preview the chosen images before publishing.

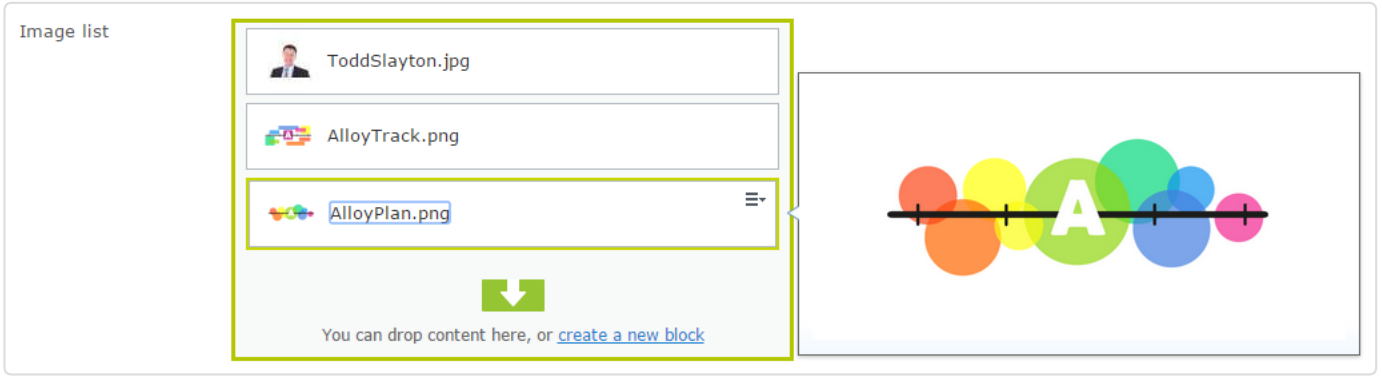


(https://gregwiechec.com/wp-content/uploads/2015/04/ContentArea.png)

I decided to give it a try and find a way to somehow inject the thumbnails to the built-in Content Area editor.

The component extends built-in Content Area editor so all features of native widget like D&D, sorting and editing are available. Property backing type is still of type ContentArea, so there will be no problems with link validation and content migration to new EPiServer versions.

For now the property supports images from media assets, but it's not difficult to implement thumbnails preview for custom blocks.



(http://gregwiechec.com/wp-content/uploads/2015/04/ContentAreaWithPreview-editor.png)

Implementing EditorDescriptor

The new descriptor simply derives from ContentAreaEditorDescriptor class and its only role is to change the client widget class to my custom implementation – “alloy.editors.contentAreaWithPreview”.

```
[EditorDescriptorRegistration (TargetType = typeof(ContentArea), UIHint = UiHint)]
public class ContentAreaEditorWithPreviewDescriptor : ContentAreaEditorDescriptor
{
    public const string UiHint = "content-area-with-preview";

    public ContentAreaEditorWithPreviewDescriptor()
    {
        this.ClientEditingClass = "alloy.editors.contentAreaWithPreview";
    }
}
```

Extending dojo widget

The extension point in Dojo widget is **_craeteNode** method of **_ContentAreaTree** widget. After calling the base _craeteNode method using inherited function, I’m modifying the content item to enable new functionality. Using this construction original _createTreeNode method of _ContentAreaTree class is executed before running custom code.

```
var _createTreeNode = this.tree._createTreeNode;
this.tree._createTreeNode = lang.hitch(this.tree, function(model) {
    var node = _createTreeNode.call(this, model);
    self._modifyNode.call(self, node, model);
    return node;
});
```

Thumbnail and preview URL fields are not available while adding a new item to the Content Area, but still we do have the access to the content ID.

To get the content together with properties we could use “epi.storeregistry” registry.

```
var registry = dependency.resolve("epi.storeregistry");
var store = registry.get("epi.cms.content.light");
```

TooltipDialog preview

I didn’t want to use large thumbnails inside ContentArea because a component with many images could be too high and it could be difficult to sort its items. That’s why I decided to show image preview as tooltip. It’s implemented using dijit TooltipDialog (http://dojotoolkit.org/reference-guide/1.10/dijit/TooltipDialog.html) widget.

```
node.imageTooltip = new TooltipDialog({
    connectId: [node.id],
    content: createHtml(),
    onMouseLeave: function() {
        popup.close(node.imageTooltip);
    }
});
```

TooltipDialog widget doesn't have open or close methods by itself. To show tooltip we have to use a popup helper from dijit controls.

```
on(imgNode, 'click', function() {
    popup.open({
        popup: node.imageTooltip,
        around: dom.byId(node.id),
        orient: ["after-centered"]
    });
});
```

Using editor

You need to copy two code snippets to install the component:

- EditorDescriptor
- Dojo widget.

To use the editor add the UIHint to ContentArea property.

```
[UIHint(ContentAreaEditorWithPreviewDescriptor.UiHint)]
public virtual ContentArea ContentAreaWithThumbnails { get; set; }
```

Property was developed and tested in EPiServer 8.

Below you can find full widget code.

```
define([
        "dojo/_base/array",
        "dojo/_base/connect",
        "dojo/_base/declare",
        "dojo/_base/lang",
        "dojo/query",
        "dojo/dom-class",
        "dojo/on",
        "dojo/dom",
        "dijit/popup",
        "dijit/TooltipDialog",
        "epi/epi",
        "epi/dependency",
        "epi-cms/contentediting/editors/ContentAreaEditor"
    ],
    function(
        array,
        connect,
        declare,
        lang,
        query,
        domClass,
        on,
        dom,
        popup,
        TooltipDialog,
        epi,
        dependency,
        _ContentAreaEditor
    ) {
        return declare("alloy.editors.contentAreaWithPreview", [_ContentAreaEditor],
            buildRendering: function() {
                this.inherited(arguments);

```

```
35                  var self = this;
36
37                  // override _createTreeNode method
38                  var _createTreeNode = this.tree._createTreeNode;
39                  this.tree._createTreeNode = lang.hitch(this.tree, function(model) {
40                      var node = _createTreeNode.call(this, model);
41                      self._modifyNode.call(self, node, model);
42                      return node;
43                  });
44              },
45
46              _modifyNode: function(node, model) {
47                  var imgNode = query("img.dijitIcon", node.domNode);
48                  if (imgNode == null || imgNode.length == 0) {
49                      return;
50                  }
51
52                  var spanLabel = query("span.dijitTreeLabel", node.domNode);
53                  if (spanLabel == null || spanLabel.length == 0) {
54                      return;
55                  }
56
57                  this._resolveContentData(model.item.contentLink, lang.hitch(this, fu
58                      if (!content.thumbnailUrl) {
59                          return;
60                      }
61
62                      // setup image
63                      imgNode = imgNode[0];
64                      domClass.add(imgNode, "epi-thumbnail");
65                      domClass.remove(imgNode, "epi-iconObjectImage");
66                      domClass.remove(imgNode, "dijitTreeIcon");
67                      imgNode.src = content.thumbnailUrl;
68
69                      // setup span
70                      spanLabel = spanLabel[0];
71                      domClass.remove(spanLabel, "dijitTreeLabel");
72                      domClass.add(spanLabel, "dojoxEllipsis");
73
74                      // preview on dblclick
75                      on(imgNode, 'dblclick', function() {
76                          window.open(content.previewUrl, '_blank');
77                      });
78
79                      this._createTooltip(node, imgNode, content.previewUrl);
80                  }));
81              },
82
83              _resolveContentData: function(contentlink, callback) {
84                  var registry = dependency.resolve("epi.storeregistry");
85                  var store = registry.get("epi.cms.content.light");
86                  if (!contentlink) {
87                      return null;
88                  }
89
90                  var contentData;
91                  dojo.when(store.get(contentlink), function(returnValue) {
92                      contentData = returnValue;
93                      callback(contentData);
94                  });
95                  return contentData;
96              },
97
98              _createTooltip: function (node, imgNode, previewUrl) {
99                  var createHtml = function() {
```

```
100                     return "<img src='" + previewUrl + "' style='max-height: 250px;m
101                 };
102
103                 node.imageTooltip = new TooltipDialog({
104                     connectId: [node.id],
105                     content: createHtml(),
106                     onMouseLeave: function() {
107                         popup.close(node.imageTooltip);
108                     }
109                 });
110
111                 on(imgNode, 'mouseleave', function() {
112                     popup.close(node.imageTooltip);
113                 });
114
115                 on(imgNode, 'click', function() {
116                     popup.open({
117                         popup: node.imageTooltip,
118                         around: dom.byId(node.id),
119                         orient: ["after-centered"]
120                     });
121                 });
122             }
123         });
124     });
```