# Restriction of content types in properties

Monday, 16 December 2013

By: Linus Ekström

I'm an architect at Ted+Gustaf. I have previously worked 13 years at EPiServer as both a developer and architect.



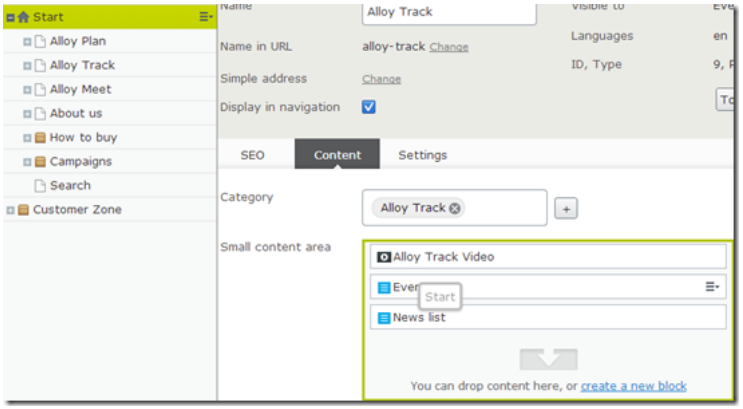
There has been a few blog and forum posts about how to restrict what can be added to a content area. In this blog post I'll show a new attribute in EPiServer 7.5 and how this can be used to accomplish this.

## Restricting a ContentArea

The AllowedTypes attribute (placed in the EPiServer.Shell assembly) can be applied to a property like this:

```
[AllowedTypes(typeof(ProductPage))]
public virtual ContentArea RelatedContentArea { get; set; }
```

When an item that is **not part** of the allowed types, is being dragged over this property, it will be greyed out and the editor will not be able to add the item to the property.




You can specify several allowed types as well as specifying inherited types:

```
[AllowedTypes(new [] {typeof(PageData), typeof(BlockData)})]
public virtual ContentArea RelatedContentArea { get; set; }
```

## Restrict content reference properties

The AllowedTypes attribute can be used for ContentReferences as well:

```
[AllowedTypes( typeof(ProductPage))]
public virtual ContentReference SomeLink { get; set; }
```

This results in the same behavior as for content areas when dragging items to the property: only items of the type "ProductPage" can be added to the property. Since it's also possible to add items by opening the content selector dialog this dialog has also got support to restrict types. Items that are not allowed according to the AllowedTypes attribute are not selectable.

*Note: Though the content selector dialog restricts selection of items that are not allowed, there is room for improvements to better indicate what is selectable or not since there are no differences of selectable/non selectable items at the moment.*

### Known limitations

There are currently a few known bugs and limitations that you might want to be aware of:

- Restriction does not work for overlays when editing on page. (**update: this is now available in the NUGET feed**)
- No server validation. Currently, the attribute only adds restriction in the UI. We hope to be able to add support for server validation soon which woul also give the posibility validate your custom properties.
- No validation when creating local blocks in content areas. If you use the new feature to add local blocks to a content area, there is currently no filtering of the content types when you create your new block.

Hopefully, we can add support for the limitations above quite soon.

EPiServer 7.5

## Comments

(By Alf Nilsson , 16 December 2013 13:21, Permanent link)

Oh great. I was missing that on Ben's blog post the other day!

(By Per Nergård , 16 December 2013 14:15, Permanent link)

Are allowed types possible to set via admin mode like on page types?

(By Linus Ekström , 16 December 2013 14:39, Permanent link)

@Per: No, this support is only possible by defining the attribute on your models and not available through the administrative interface.

(By Dan Matthews , 17 December 2013 13:49, Permanent link)

Dammit. There goes one of my favourite demos when delivering CMS training ;)

(By Dang Viet Hung , 18 December 2013 10:31, Permanent link)

thanks you for great post. I have a question. I hope that you can help me.
In Episerver 7.5, the ContentArea property use _ContentAreaTree.js for rendering items and i have to customize it for request.

This question is: How to reload tree in ContentArea affter changing model of the tree in js?
Thanks,

(By Linus Ekström , 20 December 2013 13:26, Permanent link)

Hi!

The _ContentAreaTree base class is a very internal implementation for the EPiServer user interface and I would not recommend customizing this since you are then very deep into the implementation of this editor, which might change without notice. Without knowing what you are trying to achieve it's hard to give a recommendation of how to solve your requirements.

(By Dzung Le , 13 January 2014 06:36, Permanent link)

Hi,

I've tried this with standard Alloy installation and there is weird thing, I don't know if it's the know limitation or not. Simply edit the PageListBlock, and put AllowedTypes (typeof(ArticlePage)) to the property "Root" which is PageReference.

When I go to Form Editing, try to drag and drop my Article page to the property, the property value is "undefined NaN", I got an error:

Cannot deserialize the current JSON object (e.g. {"name":"value"}) into type 'EPiServer.Core.PageReference' because the type requires a JSON string value to deserialize correctly. To fix this error either change the JSON to a JSON string value or change the deserialized type so that it is a normal .NET type (e.g. not a primitive type like integer, not a collection type like an array or List<T>) that can be deserialized from a JSON object. JsonObjectAttribute can also be added to the type to force it to deserialize from a JSON object. Path 'isPreferredLanguageAvailable', line 1, position 32.

But the picker work pretty well. Is it a bug or known bug?

(By Erik Kärrsgård , 14 January 2014 13:46, Permanent link)

Chao Dung,
There is a workaround for getting what you're trying functioning. The problem is that when you set the AllowedTypes attribute a necessary conversion metadata-info "disappears" for the root property. To add it again define the attribute like:

[AllowedTypes(typeof(ArticlePage), AllowedTypesFormatSuffix = "reference")]

You should of course not have to do that... I will report a bug on this to remove the need for extra config.

(By André Hedberg , 24 January 2014 08:55, Permanent link)

There's one more limitation if I'm not mistaken, it does not work with interfaces. :)

(By Linus Ekström , 24 January 2014 10:03, Permanent link)

@André: Actually, it is possible to use this attribute with interfaces and base classes though you might need some additional setup. Since the allowance check is done on the client, we need to calculate if a given object is allowed to be dropped on an area. However, we do not send the entire class structure for all objects to the client. By default, all content types are registered and sent to the client as well as the base types and interfaces (PageData, ContentFolder, IContentMedia etc.). If you want to add your own base class or interface this can simply be done by adding an UIDescriptor:

```
public interface IFancyPage
{
}

[UIDescriptorRegistration]
public class IFancyPageUIDescriptor : UIDescriptor<IFancyPage>
{
}
```

(By André Hedberg , 24 January 2014 14:58, Permanent link)

Oh, lovely! Thank you Linus!

(By André Hedberg , 24 January 2014 16:06, Permanent link)

It seems that I do not fully understand the implementation. :)

I've got a marker interface "IContentColumnData" for which every BlockData or PageData can be marked with.
I then create a IContentColumnUIDescriptor : UIDescriptor<IContentColumnData>.
Then mark my ContentArea property with the AllowedTypes attribute: typeof(IContentColumnData). Still, I'm not able do drag any of the marked classes to my ContentArea.

What am I missing?

(By Linus Ekström , 30 January 2014 14:14, Permanent link)

André: Just tried the code again and it works fine. Are you sure that you are not experiencing the bug described above with "on page editing"? If so, the latest patch of the EPiServer Framework NUGET package should fix this.

(By Eric Bruno , 03 February 2014 11:26, Permanent link)

I have tried the exact same thing as André described above, and it's not working for me neither. I have all the latest patches installed.

What I have done exactly is the following:

- Created a interface called IAllowRightColumn
- Created a UIDescriptor of the type IAllowRightColumn
[UIDescriptorRegistration]
public class AllowRightColumnUIDescriptor : UIDescriptor<IAllowRightColumn>
{
}

- Marked a blockdata with the interface
public class FreeTextBlockData : BlockData, IAllowRightColumn

- Added AllowedTypes to a contentarea with the IAllowRightColumn as type
[AllowedTypes(typeof(IAllowRightColumn))]

Is this not the correct way to get it to work?

---

(By Linus Ekström , 03 February 2014 15:35, Permanent link)

@Eric: That's the same code that I'm using except for different naming of my interface. I need some more info in order to be able to troubleshoot:
Are you allow to drop anything or nothing on your content areas?
Can you find your interface if you search the response from the following service: [uipath]/shell/Stores/uidescriptor/

---

(By Eric Bruno , 04 February 2014 16:14, Permanent link)

@Linus: I can't drop anything on the content area. Nothing works.

I find my interface in the response and it looks like this:
{"typeIdentifier":"sublime.site.models.iallowrightcolumn","baseTypeIdentifier":null,"iconClass":null,"containerTypeIdentifier":null,"containerTypes":null,"ma
nWidgetType":null,"disabledViews":null,"dndTypes":
["sublime.site.models.iallowrightcolumn"],"languageKey":"iallowrightcolumn","defaultView":null,"createView":null,"publishView":null,"availableViews":
[],"commandIconClass":null,"sortKey":null,"isPrimaryType":false}

---

(By Lars Loennechen Skjelbek , 06 February 2014 15:02, Permanent link)

@Linus: I experience the same behaviour as @Eric and @Andre (it doesn't work).

However, I peeked at the disassembled code (I'm sorry) and made a custom MyProjectAllowedTypes attribute for our project which does the same thing as EPiServers AllowedType in the OnMetadataCreated method, except that we use all classes that are assignable from the types specified in the attribute, not just the types themselves. Is there any reason why EPiServer's implemented isn't like that (maybe perfomance drawbacks)?

---

(By Linus Ekström , 07 February 2014 16:02, Permanent link)

Hi!

Sorry for the delay, installed a new OS this week and after setting up a new site I could finally reproduce the issues you are having. After looking into the code some more my guess is that the reason that we got different behaviour was due to the code depending on in which order types are scanned at start up.

@Lars: It's great that you found a solution. As you mention, there might be performance penalties with doing calculations in OnMetadataCreated since this is called each time an editor visits a content item (page, block , etc.) in the editorial view. Therefore I've done an alternative solution that you can run at start up that makes it possible to use the built in attribute until we can add this support to the core (This sample only shows one interface but I guess it can easily be expanded to handle multiple interfaces):

```
[ModuleDependency(typeof(Web.InitializationModule))]
public class UIDescriptorInitialization : IInitializableModule
{
public void Initialize(InitializationEngine context)
{
var uiDescriptorRegistry = ServiceLocator.Current.GetInstance<UIDescriptorRegistry>();
var descriptors = uiDescriptorRegistry.GetAllRegisteredUIDescriptors();

var fancyType = typeof(IFancyItem);//Where IFancyItem is your interface

foreach (UIDescriptor descriptor in descriptors)
{
if (fancyType.IsAssignableFrom(descriptor.ForType))
{
descriptor.DndTypes.Add(fancyType.FullName.ToLowerInvariant());
}
}
}
```

(By Jonas Boman , 13 February 2014 12:57, Permanent link)

dbl post

(By Jonas Boman , 13 February 2014 12:57, Permanent link)

Any updates on resolving the issue AllowedTypes not working?!

(By Linus Ekström , 13 February 2014 14:06, Permanent link)

The bug with on page edit not working correctly has been released (just install the latest patches). The other issues/limitations still exists.

(By Dzung Le , 01 March 2014 10:40, Permanent link)

Oh I missed the comment from Erik. Thanks alot.

(By Jacob Khan , 20 March 2014 20:39, Permanent link)

Is there a way not to allow the user to create a new block?

(By Linus Ekström , 24 March 2014 10:31, Permanent link)

You can actually override the translated text message that contains the create link. Try adding something like this in your "local" language files:

```
<episerver>
<cms>
<widget>
<overlay>
<blockarea>
<emptyactions>
<template see="/episerver/cms/widget/overlay/blockarea/emptyactions/templatewithoutcreate" />
</emptyactions>
</blockarea>
```

```
</overlay>
</widget>
</cms>
</episerver>
```

(By Edmund Ward , 04 July 2014 16:25, Permanent link)

Hi Linus, the Interface work-around doesn't work for the AllowedTypes decoration of my ContentReference property. Do the Dnd types affect a ContentReference picker, or does it have to be approached in a different way? E.g. I want to do the following:

```
[AllowedTypes(typeof(ICanBePositioned))]
public virtual ContentReference CanBePositionedContent { get; set; }
```

(By Linus Ekström , 04 July 2014 16:39, Permanent link)

The drag and drop work around for interfaces only affects what you can drag and drop to the property. So, you should be able to drag and drop content instances that implement the interface. However, this does not affect the dialog which I suspect will be pretty empty if you only accept an interface.

I can add that we have added support to handle base classes out of the box and that this change is currently under testing. For interfaces however, the change required is a bit trickier, but we have discussed some ideas on how to solve this and I hope that we can start investigating this later this year.

(By Paul Smith , 11 December 2014 13:39, Permanent link)

I was hoping this might work for MediaData (and derived types) but doesn't seem to. Any ideas?

Post by Paul Smith

(By Paul Smith , 12 December 2014 14:05, Permanent link)

I made a validator which seems to work nicely and gives the editor good feeback :

```
public class AllowedTypesValidator : IValidate<PageData>
{
private readonly IContentLoader contentLoader;

public AllowedTypesValidator(IContentLoader contentLoader)
{
this.contentLoader = contentLoader;
}

public IEnumerable<ValidationError> Validate(PageData instance)
{
var validationErrors = new List<ValidationError>();

foreach (var propertyInfo in instance.GetType().GetProperties(BindingFlags.Instance | BindingFlags.Public))
{
var allowedTypesAttributes = propertyInfo.GetCustomAttributes<AllowedTypesAttribute>();

foreach (var allowedTypesAttribute in allowedTypesAttributes)
{
var allowedTypes = allowedTypesAttribute.AllowedTypes;
var propertyValue = propertyInfo.GetValue(instance);
var contentReference = propertyValue as ContentReference;

if (contentReference != null)
{
```

```
if (!this.HasValidContent(contentReference, allowedTypes))
{
validationErrors.Add(CreateValidationError(propertyInfo, allowedTypes));
}

continue;
}

var contentArea = propertyValue as ContentArea;

if (contentArea != null)
{
foreach (var contentItem in contentArea.FilteredItems)
{
if (!this.HasValidContent(contentItem.ContentLink, allowedTypes))
{
validationErrors.Add(CreateValidationError(propertyInfo, allowedTypes));
}
}
}
}
}

return validationErrors;
}

private bool HasValidContent(ContentReference contentReference, Type[] allowedTypes)
{
var content = contentLoader.Get<IContent>(contentReference);
return allowedTypes.Any(type => type.IsInstanceOfType(content));
}

private ValidationError CreateValidationError(PropertyInfo propertyInfo, Type[] allowedTypes)
{
var propertyName = propertyInfo.Name;
var displayAttribute = propertyInfo.GetCustomAttribute<DisplayAttribute>();

if (displayAttribute != null && !string.IsNullOrEmpty(displayAttribute.Name))
{
propertyName = displayAttribute.Name;
}

var message = string.Format(
"The property '{0}' has a value which is not allowed for this type. The allowed types are : ",
propertyName);

foreach (var allowedType in allowedTypes)
{
var typeDisplayName = allowedType.Name;
var contentTypeAttribute = allowedType.GetCustomAttributes<ContentTypeAttribute>().FirstOrDefault();

if (contentTypeAttribute != null && !string.IsNullOrEmpty(contentTypeAttribute.DisplayName))
{
typeDisplayName = contentTypeAttribute.DisplayName;
}

message += typeDisplayName + ";";
}

message = message.Substring(0, message.Length - 1);

return new ValidationError()
{
ErrorMessage = message, PropertyName = propertyName
};
}
}
```

(By Linus Ekström , 13 January 2015 11:16, Permanent link)

Great that you found a solution Paul. There is work ongoing right now to improve the support to restrict types, that will hopefully handle working with interfaces and base types and solve the most common issues.

(By getElephantById , 19 March 2015 04:23, Permanent link)

Howdy,

AllowedTypes attribute seems to be working nicely with base classes. Running version 8.0.0.0 - drag'n'drop gets greyed out in on-page editing and forms editing. And when clicking 'Create new block' - only the allowed block types are available.

This is cool.

Thanks