

PATRICKVANKLEEF.COM

– Passionate Developer –

- [Home](#)
- [EPiServer Find demo site](#)
- [About me](#)
- [GitHub](#)
- [LinkedIn](#)

PATRICKVANKLEEF.COM

– Passionate Developer –

[*EPiServer*](#)

[**EPiServer custom property in DOJO**](#)



For a client I had a requirement to manage an email address by country. The countries are stored in categories. There are a number of ways to get this done. The easiest solution would be to make a container page type that contains two properties a category selector and a text field to store the email address. Personally I don't think it's the best solution, because another page type needs to be created and all the items are displayed in the page tree. It seemed me a better idea to create a property on the page that displayed a label and a text field for each country. Of course DOJO is the framework for creating custom properties in EPiServer. For this blog I created code that also can be found on my [GitHub](#) account.

- I will explain the following topics:
 - Add/edit module.config – *configuring the property*
 - EditorDescriptor – *used for passing configuration data to the property script*
 - Property Type – *used for retrieving and storing the property value*

– JS module script – *JS script for displaying the property*

Add/ edit module.config

The module.config is used for configuring DOJO modules. This configuration file isn't standard created with a EPiServer installation. When creating a DOJO module, such as a custom property, the module.config needs to be created in the root directory of the website. In the module.config the path to the property files are configured.

```
1 <module>
2   <assemblies>
3     <add assembly="EPiServerCustomProperty" />
4   </assemblies>
5
6   <dojoModules>
7     <add name="app" path="Scripts" />
8   </dojoModules>
9 </module>
```

The assemblies section specifies which assemblies belong to the module, also the path to the resource files is configured.

EditorDescriptor

The EditorDescriptor communicates with the custom property how the property should render. By setting the ClientEditingClass it tells the edit mode what JS module script must be used for rendering this property. The same value must be used within the JS script module. When passing data to the module script the EditorConfiguration can be used, which is a name value collection. To create a EditorDescriptor class it must inherit from the EditorDescriptor class and needs to be decorated with the EditorDescriptorRegistration attribute.

```
1 [EditorDescriptorRegistration(TargetType = typeof(IEnumerable<CountryEmailAddress>), UIHint = "CountryEmailAddressesProperty")]
2 public class CountryEmailAddressEditorDescriptor : EditorDescriptor
3 {
4     public CountryEmailAddressEditorDescriptor()
5     {
6         var countryService = new CountryService();
7
8         ClientEditingClass = "app.editors.CountryEmailAddressesProperty";
9
10        EditorConfiguration["Countries"] = countryService.GetAll();
11    }
```

```
12    }
    }
```

In the code above I used the `TargetType` of the `EditorDescriptorRegistration` attribute for defining the target type of the property. The `UIHint` is used for defining the custom property, I will use this later in the code for defining the property on the page type. In the constructor of the class I set the `ClientEditingClass` and insert a collection of countries on the `EditorConfiguration` name value collection.

The method `GetAll` of the `CountryService` class returns a list of country objects. The country class contains two string properties: `Code` and `Name`.

I will explain the `TargetType` in the next section.

Property Type

The property type is used for retrieving and storing the data. A property type can be defined by using the `PropertyDefinitionTypePlugin` attribute and inheriting from the `PropertyData` class or inheriting from one of the existing property types, such as the `PropertyLongString`.

```
1 [PropertyDefinitionTypePlugIn(Description = "A property for list of country email address.", DisplayName = "Country Emailaddress
2 Items")]
3 public class PropertyCountryEmailAddress : PropertyLongString
4 {
5     public override Type PropertyValueType
6     {
7         get { return typeof(IEnumerable<CountryEmailAddress>); }
8     }
9     public override object Value
10    {
11        get
12        {
13            var value = base.Value as string;
14            if (value == null)
15            {
16                return null;
17            }
18            var serializer = new JavaScriptSerializer();
19            return serializer.Deserialize(value, typeof(IEnumerable<CountryEmailAddress>));
20        }
21    }
```

```
22     set
23     {
24         if (value is IEnumerable<CountryEmailAddress>)
25         {
26             var serializer = new JavaScriptSerializer();
27             base.Value = serializer.Serialize(value);
28         }
29         else
30         {
31             base.Value = value;
32         }
33     }
34 }
35
36 public override object SaveData(PropertyDataCollection properties)
37 {
38     return LongString;
39 }
}
```

In the above code I created a property type. A number of properties and methods can be overridden from the PropertyLongString. The Type property indicates which Type is used for working with this property. In my case I would like to work with a IEnumerable of CountryEmailAddress objects, I will show the CountryEmailAddress class later. The Value property is used for serializing and deserializing the property value. The value is stored as a string and deserialized as a IEnumerable of CountryEmailAddress objects.

The CountryEmailAddress contains two properties the country code and the email address.

```
1 public class CountryEmailAddress
2 {
3     public string CountryCode { get; set; }
4     public string EmailAddress { get; set; }
5 }
```

JS module script

```
1 define([
2     "dojo/_base/array",
```

```
3     "dojo/_base/declare",
4
5     "dijit/registry",
6     "dijit/_Widget",
7     "dijit/_TemplatedMixin",
8     "dijit/_Container",
9     "dojox/layout/TableContainer",
10    "dijit/form/ValidationTextBox",
11    "dojo/text!./templates/CountryEmailAddresses.html"
12],
13function (
14    array,
15    declare,
16
17    registry,
18    _Widget,
19    _TemplatedMixin,
20    _Container,
21    TableContainer,
22    ValidationTextBox,
23    template
24) {
25    return declare("app.editors.CountryEmailAddressesProperty", [
26        _Widget,
27        _TemplatedMixin,
28        _Container], {
29        templateString: template,
30        postCreate: function () {
31
32            var layout = new TableContainer({
33                showLabels: true,
34                orientation: "horiz"
35            });
36
37            for (var i = 0; i < this.countries.length; i++) {
38                var textBox = new ValidationTextBox({
39                    name: "id_" + this.countries[i].code,
```

```
40         id: this.countries[i].code,
41         type: "text",
42         class: "countryEmailTextbox",
43         title: this.countries[i].name,
44         invalidMessage: "The entered value is not valid. Must be a valid e-mail.",
45         regexp: "^[a-zA-Z0-9_.-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+$"
46     });
47     layout.addChild(textBox);
48 }
49 this.container.appendChild(layout.domNode);
50 },
51 _getValueAttr: function () {
52     var jsonObject = [];
53     var textboxCollection = dojo.query(".countryEmailTextbox");
54
55     for (var i = 0; i < textboxCollection.length; i++) {
56
57         if (textboxCollection[i] != null) {
58
59             var widget = registry.byNode(textboxCollection[i]);
60
61             var item = {
62                 CountryCode: widget.id,
63                 EmailAddress: widget.get("value")
64             }
65
66             jsonObject.push(item);
67         }
68     }
69     return jsonObject;
70 },
71 _setValueAttr: function (val) {
72     if (val != null) {
73         for (var i = 0; i < val.length; i++) {
74             var widget = registry.byId(val[i].countryCode);
75             if (widget != null) {
76                 widget.set("value", val[i].emailAddress);
```

```

77         }
78     }
79 }
80 },
81 isValid: function () {
82     var isValid = true;
83     var textboxCollection = dojo.query(".countryEmailTextbox");
84
85     for (var i = 0; i < textboxCollection.length; i++) {
86
87         if (textboxCollection[i] != null) {
88
89             var widget = registry.byNode(textboxCollection[i]);
90             isValid = isValid && widget.isValid();
91         }
92     }
93     return isValid;
94 }
95 }
96 );
97 });

```

The script uses the [AMD](#) (Asynchronous Module Definition) pattern for loading Javascript modules. In my script I use a number of modules, I will explain just a couple of the modules:

dojo/_base/declare	Contain functions for define the DOJO class
dijit/registry	Stores a collection of widgets within a page. It contains functions for retrieving a widget by ID or DOM node.
dijit/form/ValidationTextBox	A TextBox control that checks whether the user input is valid
dojox/layout/TableContainer	A layout control that displays the containing widgets in a table layout.
dojo/text!./templates/CountryEmailAddreses.html	A reference to the HTML template of

the property

For more information about DOJO classes and controls visit the [reference guide](#).

My DOJO (property) widget contains five properties, described below:

templateString	Is the reference to the HTML template of the property
postCreate	This function is called when the widget has been rendered. In this function the list of countries (passed from the EditorDescriptor) is looped and for each country a ValidationTextBox control is created.
_getValueAttr	This function is the getter for the value property. The getter will return a JSON that contains a list of countries and email addresses.
_setValueAttr	This function is the setter for the value property. For each item in the JSON value the TextBox control is set with the email address.
isValid	This function checks all TextBox control values if it's a valid email address.

When defining the property on the page type a couple of things needs to be done. First the UIHint attribute must match with the UIHint value set on the EditorDescriptor class. The BackingType attribute is set to the PropertyCountryEmailAddress class for deserialize the property value to a IEnumerable of CountryEmailAddress.

```

1 [UIHint("CountryEmailAddressesProperty")]
2 [Display(
3     Name = "Country email addresses",
4     Description = "Country email addresses",
5     GroupName = SystemTabNames.Content,
6     Order = 10)]
7 [BackingType(typeof(PropertyCountryEmailAddress))]
8 public virtual IEnumerable<CountryEmailAddress> CountryEmailAddresses { get; set; }

```

The result in the CMS:

Country email addresses	Netherlands	netherlands@episerver.c	The entered value is not valid. Must be a valid e-mail.
	Belgium	asdf	
	Sweden	sweden@episerver.com	

Because the property returns a IEnumerable a simple loop can display the items on the page:

```
1 <div>
2   @if (Model.CountryEmailAddresses != null)
3   {
4       <ul>
5           @foreach (var item in Model.CountryEmailAddresses)
6           {
7               <li>@item.CountryCode - @item.EmailAddress</li>
8           }
9       </ul>
10  }
11</div>
```

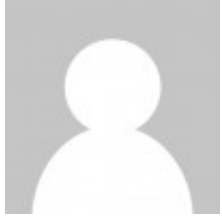
You can find the full source code on my [GitHub](#) account.

[DOJO](#), [EPiServer](#)
[February 3, 2015](#) by [Patrick](#)



Written by Patrick

10 Comments



1. Raju February 6, 2015 at 14:45 [Reply](#)

Nice post! learned something. May i know why overridden ModifyMetadata() from EditorDescriptor is not used? is it fine with coding within constructor?



2. pat_kleef@hotmail.com February 6, 2015 at 15:06 [Reply](#)

Hi Raju,

It's also possible to override the ModifyMetadata when you need the entire metadata object. EPiServer advises that code is placed in the constructor of the EditorDescriptor class when you don't need the entire metadata object.

<http://world.episerver.com/documentation/Class-library/?documentId=episerverframework/7/2e6efd8c-d09f-7911-f656-f4a038f14877>



3. Naz February 11, 2015 at 16:04 [Reply](#)

Excellent !!, very well written and explained everything precisely. Thank you so much.
I have managed to repeat this on my version 7 of episerver and working fine.

My requirement is : I am trying to create a custom property using Dojo which will be like link collection, you click on the button and it opens

up a dialogue box where you fill in few fields and a Page reference field and it stores them in a collection. Means for each page there will be number of entries and each entry will have fields say 2 text boxes and 1 page reference field. I fill the value and when I hit save, it should just save to the page. I need to know how the page value will be saved and how do i do this. Have you done anything similar or if you can guide me through and also I can do this in dojo. Correct me please if I am wrong?

Many thanks once again.



4. Patrick February 12, 2015 at 19:24 [Reply](#)

Hi Naz,

The property value should be saved as JSON, just as how I did in the example. The PropertyCountryEmailAddress class is responsible how the value should be stored and retrieved. The Dojo script will receive the JSON to set the values when in editing mode and it should return a JSON when saving the page.



5. Nazim February 15, 2015 at 14:08 [Reply](#)

Thanks Patrik, I am getting it slowly now, One more question:-

Do you know anything in dojo in the script file actually I can call the Page selection property similar to episerver. A button “...” that looks into the editor pages similar to page reference collection and when I select a page it stores the pagereference? I am struggling to find anywhere.

Many thanks,
Naz



6.

Patrick February 15, 2015 at 16:27 [Reply](#)

At the moment I'm working on a custom property that's using the page selection property of EPiServer. This week I will publish a blog about it and of course share my code.

Patrick



7.

Naz February 16, 2015 at 16:47 [Reply](#)

Thanks Patrick, I will wait for it,
Anyways, if you care to share a quick start please, my email is nazimzia@hotmail.com.

Cheers,
Naz



8.

Patrick February 22, 2015 at 10:14 [Reply](#)

Hi Naz,

Take a look at my latest blog post: <http://www.patrickvankleef.com/2015/02/22/multiple-page-picker-property-in-dojo/>

Patrick



9.

Chris June 17, 2015 at 00:15 [Reply](#)

Hi,

Nice post with high value.

I have one question: how your dojo script notifies EPI that value in any ValidationTextBox has changed?

Chris



o

[Patrick](#) June 17, 2015 at 09:13 [Reply](#)

Hi Chris,

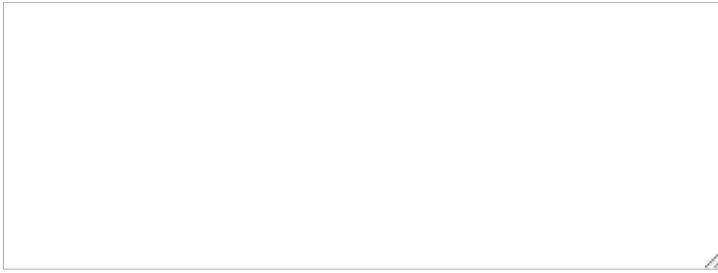
EPiServer will call the `_getValueAttr` method when the value of the textbox changes (onblur event).

Leave a Comment

Name (required)

Mail (required)

Website



Leave a Comment

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

[« Previous post](#)

[Force a page to display in a specific language dynamically.](#)

[Next post »](#)

[EPiServer XForms reCAPTCHA Webforms implementation](#)

Related Posts



[Show content based on the visitor's current weather](#)

[May 25, 2015](#)



[Use culture specific MediaData properties](#)

[June 16, 2014](#)



[Online translation localization provider](#)

[September 20, 2014](#)

Success is the result of perfection, hard work, learning from failure, loyalty, and persistence. - **Colin Powell**

© 2015 copyright patrickvankleef.com// All rights reserved //

