



# Ascend

# LAB: Extending Episerver in Five Easy Ways

# VisitorGroups Criteria

Personalize based on  
weather

Criteria

Match **All** ▼

Drop new criterion here

Weather Type

Fallback Location

Use visitors location

Weather

Hawaii

Clear

✕

Weather Type

Fallback Location

Use visitors location

Weather

☒

Rain

✕

Other Information

Name

Promote Hawaii Sunny Travel to Rain plagued visitors

Notes

Security role

Statistics

☐ Make this visitor group available when setting access rights for pages and files

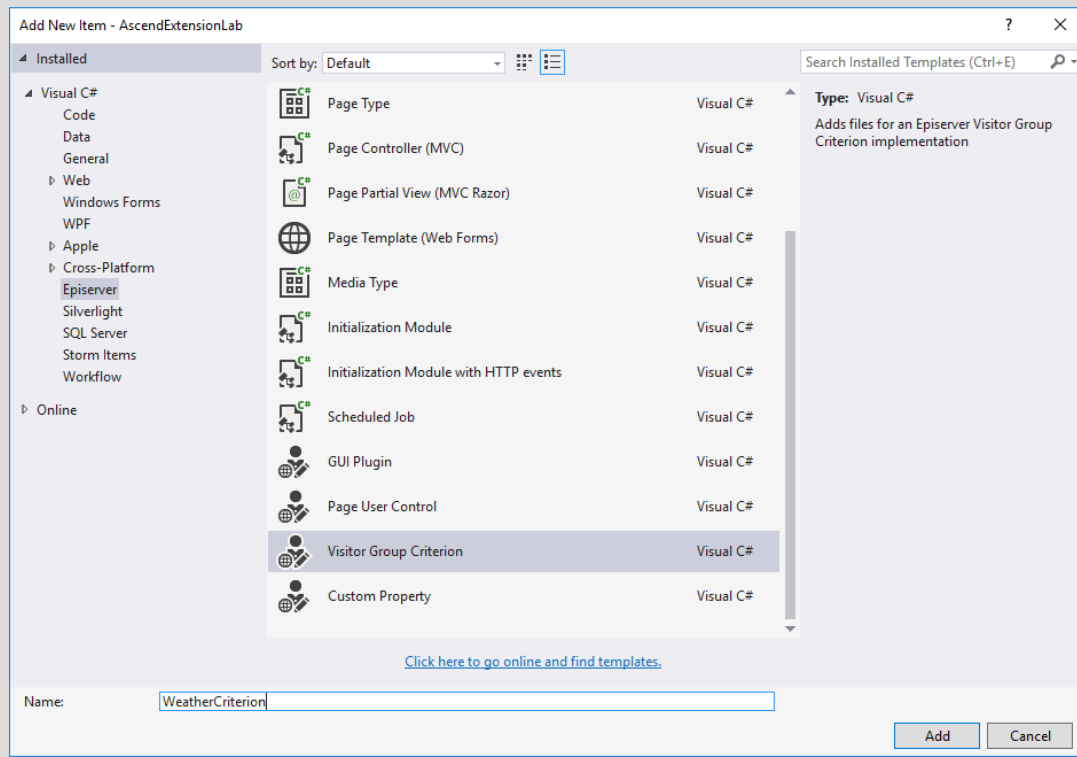
☒ Enable statistics for this visitor group

## Target message based on visitors location

- Criteria is a great way to customize personalization
- Typically consists of a Criterion and a Model
- Can subscribe and unsubscribe to system events
- Key method to override:

```
public override bool IsMatch(IPrincipal principal, HttpContextBase httpContext)
```

# Start from Episerver Visual Studio Templates



## Preexisting code

WeatherTypes (Enum)

Types of weather

WeatherBroker (class)

Calls the Weather API

# Model

```

1 reference
public class WeatherModel : CriterionModelBase
{
    [DojoWidget(LabelTranslationKey = "/weathermodel/defaultlocation")]
    1 reference
    public string DefaultLocation { get; set; }

    [DojoWidget(LabelTranslationKey = "/weathermodel/usevisitorslocation")]
    1 reference
    public bool UseVisitorsLocation { get; set; }

    [DojoWidget(
        SelectionFactoryType = typeof(EnumSelectionFactory),
        AdditionalOptions = "{ selectOnClick:true}")]
    2 references
    public WeatherTypes Weather { get; set; }

    0 references
    public override ICriterionModel Copy()
    {
        return ShallowCopy();
    }
}

```

# Criterion

```
[VisitorGroupCriterion(  
    DisplayName = "Weather Type",  
    Category = "Weather")]  
0 references  
public class WeatherCriterion : CriterionBase<WeatherModel>  
{  
    0 references  
    public override bool IsMatch(IPrincipal principal, HttpContextBase httpContext)  
    {  
        try  
        {  
            var pos = GeoPosition.GetUsersPositionOrNull();  
            if (Model.UseVisitorsLocation && (pos != null))  
            {  
                return (WeatherBroker.GetWeatherType(pos) == Model.Weather);  
            }  
            return (WeatherBroker.GetWeatherType(Model.DefaultLocation) == Model.Weather);  
        }  
        catch (Exception)  
        {  
            return false;  
        }  
    }  
}
```



## Included files

Helpers:

- /Resources/LanguageFiles/VisitorGroups.xml
- /Helpers/GeoPosition.cs
- /Business/Criteria/WeatherBroker.cs
- /Business/Criteria/WeatherTypes.cs

Solution

- /Solutions/Business/Criteria/WeatherCriterion.cs
- /Solutions/Business/Criteria/WeatherModel.cs

## Read more

<https://world.episerver.com/documentation/developer-guides/CMS/personalization/developing-custom-visitor-group-criteria/>

# Custom Property Lists

Make a property that can contain a list of addresses.

Our Addresses ×

+

Line1	Line2	Zipcode	City
Njalsgade 23		2300	Copenhagen
542 Amherst Street		3063	Nashua
20 Pacifica	Suite 430	92618	Irvine
Regeringsgatan 67	5th floor	11156	Stockholm

## Simple Lists

Since CMS UI 11.1.0

```
[Required]
[Display(Order = 305)]
[UIHint(Global.SiteUIHints.StringsCollection)]
[CultureSpecific]
0 references
public virtual IList<string> UniqueSellingPoints { get; set; }
```

Unique selling points

1	::	<input type="text" value="Project planning"/>
2	::	<input type="text" value="Reporting and statistics"/>
3	::	<input type="text" value="Email handling of tasks"/>
4	::	<input type="text" value="Risk calculations"/>
5	::	<input type="text" value="Direct communication to"/>
+		

## Complex Lists

- List of multi-property objects
- Access through `IList<>` and POCO
- Useful when `ContentArea/Content` is overkill
- Useful for behind-the-scenes properties
- Requires base type (POCO), `PropertyDefinitionPlugIn` and property definition

## Live examples of complex lists

Send email after form submission



**From**

**To**

**Subject**

**Message**

*There are no items available.*

Trigger webhook after form submission



**Webhook URL**

**POST data in JSON format**

*There are no items available.*

## Define base type (POCO)

```

3 references
public class Address
{
    0 references
    public string Line1 { get; set; }
    0 references
    public string Line2 { get; set; }
    0 references
    public int Zipcode { get; set; }
    0 references
    public string City { get; set; }
}

```

# PropertyDefinitionPlugIn

```
[PropertyDefinitionTypePlugIn]
```

0 references

```
public class AddressesListProperty : PropertyList<Address>  
{  
}
```



## Property Definition

NOTE: 'OfficesBlock' is already existing

```
public class OfficesBlock : BlockData
{
    [EditorDescriptor(EditorDescriptorType = typeof(CollectionEditorDescriptor<Address>))]
    [Display(Name = "Our Addresses")]
    0 references
    public virtual IList<Address> OurAddresses { get; set; }
}
```

# Property Rendering

```
<div @Html.EditAttributes(m => m.OurAddresses)>
  @if (Model.OurAddresses != null)
  {
    foreach (var o in Model.OurAddresses)
    {
      <p>
        @o.Line1<br/>
        @o.Line2<br/>
        @o.City<br/>
        @o.Zipcode
      </p>
    }
  }
</div>
```

## Included files

### Helpers

- /Models/Blocks/OfficesBlock.cs
- /Views/Shared/Blocks/OfficesBlock.cs

### Solution

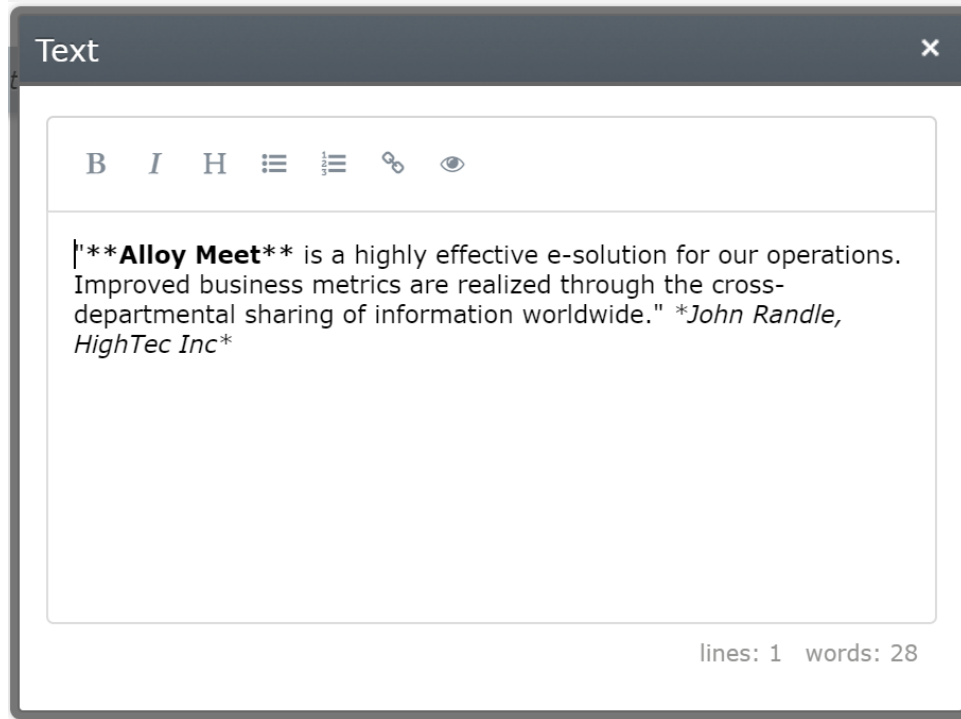
- /Solutions/Models/Address.cs
- /Solutions/Models/AddressPropertyDefinition.cs
- /Solutions/Views/Shared/Blocks/OfficesBlock.cs
- /Solutions/Models/Blocks/OfficesBlock.cs

## Learn more about Property Value Lists

- <https://world.episerver.com/blogs/bartosz-sekula/dates/2017/10/property-value-list/>
- <https://world.episerver.com/blogs/Per-Magne-Skuseth/Dates/2015/11/trying-out-propertylistt/>

## Custom string editor

Use Markdown instead of XHTML.



## Markdown for editing content

- Used by Wikipedia, Jira, etc.
- Avoids broken HTML syntax
- Separates markup from styling
- We'll use SimpleMDE as the text editor
  - And Markdig for rendering in Razor

# EditorDescriptor – MarkdownEditorDescriptor.cs

```
[EditorDescriptorRegistration(
    TargetType = typeof(string),
    UIHint = UIHint,
    EditorDescriptorBehavior = EditorDescriptorBehavior.PlaceLast)]
public class MarkdownEditorDescriptor : EditorDescriptor
{
    public const string UIHint = "Markdown";

    public override void ModifyMetadata(ExtendedMetadata metadata, IEnumerable<Attribute> attributes)
    {
        base.ModifyMetadata(metadata, attributes);

        metadata.ClientEditingClass = "alloy/editors/markdowneditor/Editor";
    }
}
```

## UIHint – TeaserBlock.cs (for example)

```
// ...  
public class TeaserBlock : SiteBlockData  
{  
    // ...  
    [UIHint(MarkdownEditorDescriptor.UIHint)]  
    public virtual string Text { get; set; }  
  
    // ...  
}
```



# Demo editing

## DisplayTemplate – Markdown.cshtml

```
@model string
```

```
@Html.Raw(Markdig.Markdown.ToHtml(Model ?? string.Empty))
```

# Demo rendering

# Gadget life-cycle – Editor.js

```
define([/* ... */], function (/* ... */) {  
    return declare([_Widget, /* ... */], {  
  
        buildRendering: function () {  
            // When the DOM has finished loading.  
        },  
  
        destroy: function () {  
            // Make sure to not leak memory.  
        },  
  
    });  
});
```

# Gadget integration – Editor.js

```
define([/* ... */], function (/* ... */) {  
    return declare([_Widget, /* ... */ _ValueRequiredMixin], {  
  
        onChange: function (value) {  
            // Triggers auto-save.  
        },  
  
        resize: function () {  
            // Called when switching tabs.  
        },  
  
        isValid: function () {  
            // Support _ValueRequiredMixin.  
        },  
  
        _setValueAttr: function (value) {  
            // Startup and undo.  
        },  
  
        _setReadOnlyAttr: function (value) {  
            // Support compare view.  
        },  
  
    });  
});
```

## Included files

### Helpers

- /ClientResources/Scripts/Editors/MarkdownEditor/simplemde/\*
- /ClientResources/Scripts/Editors/MarkdownEditor/Editor.js
- /ClientResources/Scripts/Editors/MarkdownEditor/Template.[html|css]

### Solution

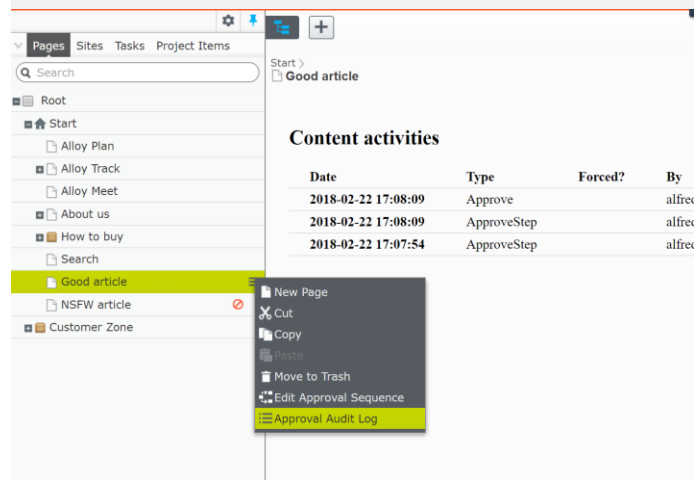
- /Solution/Business/EditorDescriptors/MarkdownEditorDescriptor.cs
- /Solution/Models/Blocks/TeaserBlock.cs
- /Solution/Views/Shared/Blocks/TeaserBlock(Wide).cshtml
- /Solution/Views/Shared/DisplayTemplates/Markdown.cshtml

## Learn more about custom editors

- <https://world.episerver.com/documentation/developer-guides/CMS/editing/Registering-a-custom-editor/>

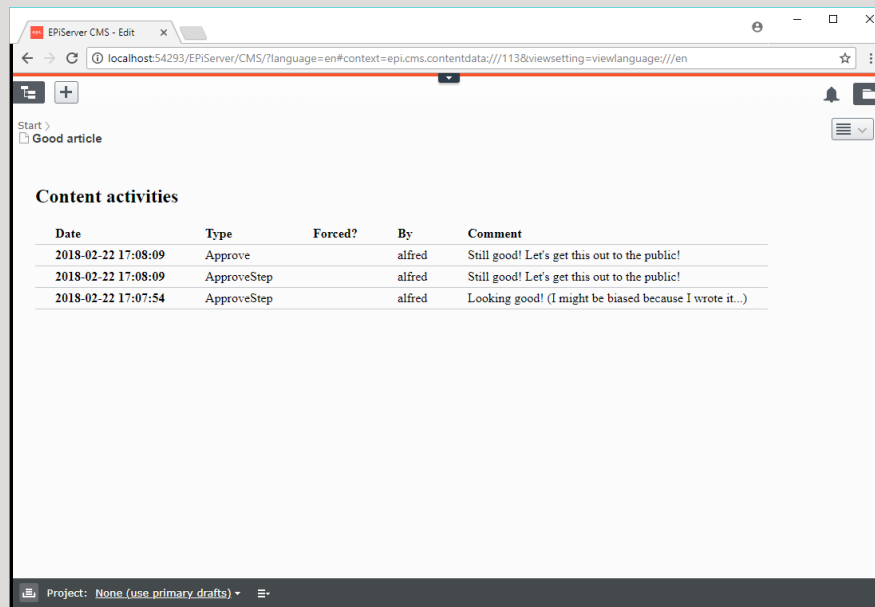
## Custom content view

Show an audit log of Content Approval changes made on content (page, block, or media).





# Helpers: ApprovalLog controller & view



The screenshot shows the Episerver CMS 'Edit' view for a content item titled 'Good article'. The browser address bar indicates the URL: `localhost:54293/EPiServer/CMS/?language=en#context=epi.cms.contentdata:///113&viewsetting=viewlanguage:///en`. The page displays a table of 'Content activities' (approval log) for this item.

Date	Type	Forced?	By	Comment
2018-02-22 17:08:09	Approve		alfred	Still good! Let's get this out to the public!
2018-02-22 17:08:09	ApproveStep		alfred	Still good! Let's get this out to the public!
2018-02-22 17:07:54	ApproveStep		alfred	Looking good! (I might be biased because I wrote it...)

At the bottom of the interface, the 'Project' dropdown is set to 'None (use primary drafts)'.

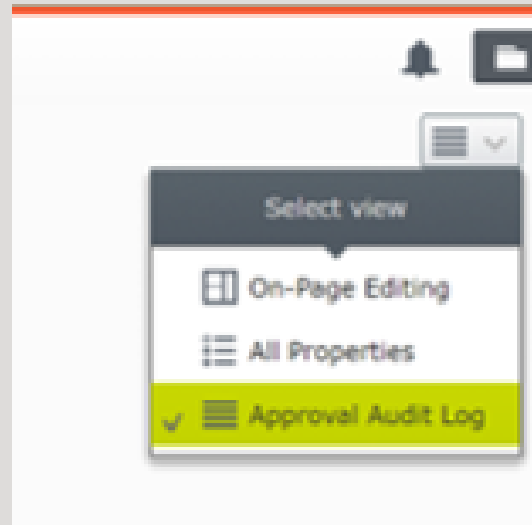
# Route – Global.asax.cs

```
public class EPiServerApplication : EPiServer.Global
{
    // ...

    protected override void RegisterRoutes(RouteCollection routes)
    {
        base.RegisterRoutes(routes);

        routes.MapRoute(
            "ApprovalLog",
            "ApprovalLog",
            new { controller = "ApprovalLog", action = "index" }
        );
    }
}
```

## View selection

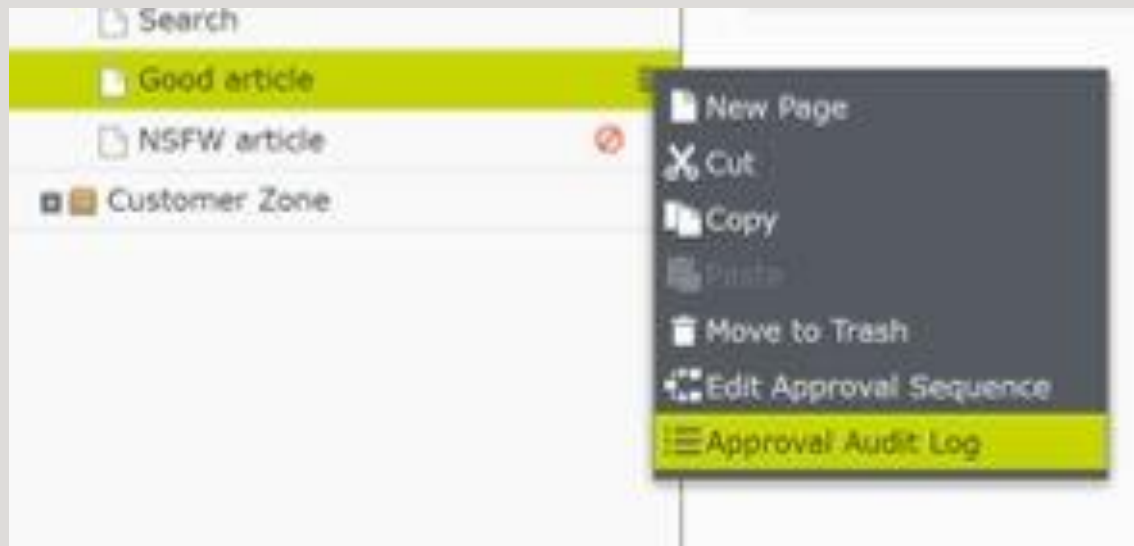


# View configuration – ApprovalLogView.cs

```
[ServiceConfiguration(typeof(ViewConfiguration))]  
public class ApprovalLogView : ViewConfiguration<IContentData>  
{  
    public const string ViewKey = "approvalLog";  
  
    public ApprovalLogView()  
    {  
        Key = ViewKey;  
        Name = "Approval Audit Log";  
        Description = "Approval Audit Log";  
        IconClass = "epi-iconList";  
  
        ControllerType = "epi-cms/widget/IFrameController";  
        ViewType = "/ApprovalLog/";  
    }  
}
```

# Demo view selection

## Navigation plug-in area



## Module setup – module.config

```
<?xml version="1.0" encoding="utf-8"?>
<module>
  <!-- ... -->
  <clientModule initializer="alloy/NavigationTreePlugins">
    <moduleDependencies>
      <add dependency="CMS" type="RunAfter" />
    </moduleDependencies>
  </clientModule>
</module>
```

## Navigation plug-in area – NavigationTreePlugins.js

```
define([/* ... */], function (/* ... */) {  
    return declare([_Module], {  
  
        initialize: function () {  
            this.inherited(arguments);  
  
            navigationTreePluginArea.add(ApprovalLogCommand);  
        }  
  
    });  
});
```



## Command – ApprovalLogCommand.js

```
define([/* ... */], function (/* ... */) {  
    return declare([_Command], {  
  
        _execute: function () {  
            topic.publish('/epi/shell/action/changeview', 'approvalLog', null, null, true);  
  
            topic.publish('/epi/shell/context/request', { uri: 'epi.cms.contentdata:/// ' + this.model.contentLink });  
        }  
  
    });  
});
```

# Demo navigation tree

## Included files

### Helpers

- /Controllers/ApprovalLogController.cs
- /Models/ViewModels/ContentActivityViewModel.cs
- /Views/ApprovalLog/Index.cshtml
- /ClientResources/Scripts/ApprovalViewLog/ApprovalLogCommand.js
- /ClientResources/Scripts/NavigationTreePlugins.js

### Solution

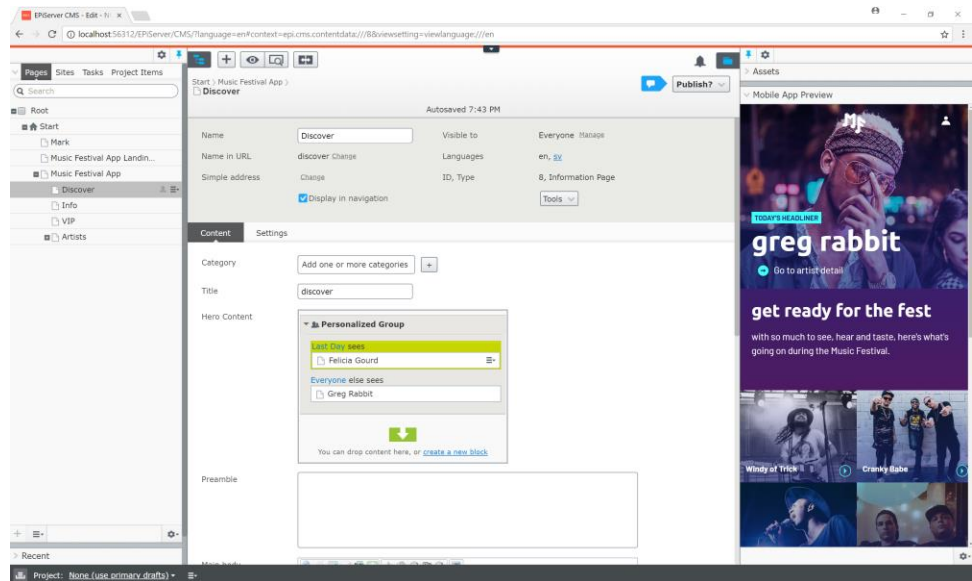
- /Solution/ApprovalLogView/ApprovalLogView.cs
- /Solution/Global.asax.cs
- /Solution/module.config

## Learn more about custom views

- <https://world.episerver.com/documentation/developer-guides/CMS/user-interface/plugin-areas/>

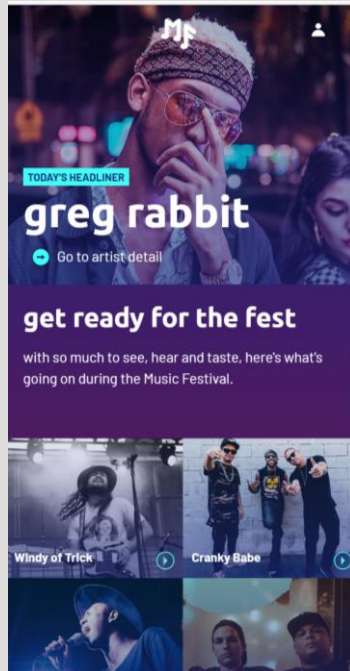
# UI Component

Preview a mobile app while editing content.



## Headless API

- Lab tomorrow!
  - Episerver as Headless  
10:30 am – 11:45 am  
Starvine 4
- We'll borrow the sample site
- And the mobile site!



## Content site

- Open \MusicFestival\EPiServer.ContentApi.sln
- Run site
- Log in
  - Username: epiadmin
  - Password: Epi123!

## Mobile app

- Run `\MusicFestival\setup.cmd`
- Log in
  - Username: **epiadmin**
  - Password: **Epi123!**



# Demo mobile app

## IFrameComponent – AppPreviewComponent.cs

```
[IFrameComponent(Url = "http://localhost:8080",  
    ReloadOnContextChange = false,  
    PlugInAreas = "/episerver/cms/assets",  
    Title = "Mobile App Preview",  
    Categories="cms",  
    MinHeight = 100,  
    MaxHeight = 500)]  
public class AppPreviewComponent : ContentWebFormsBase  
{  
}
```

# Demo preview gadget

## "beta/contentSaved" event

- Session tomorrow!
  - Exploring OPE with Angular and React  
1:05 pm – 1:55 pm  
Ironwood 7
- We'll listen to the event, to know when to refresh the app view

## window message "beta/contentSaved" – main.js

```
// ...  
  
window.addEventListener('message', function (event) {  
  let eventArgs = event.data  
  if (eventArgs && eventArgs.id === 'beta/contentSaved') {  
    EventBus.$emit('contentSaved', eventArgs.data)  
  }  
}, false)  
  
// ...
```

## Vue EventBus "contentSaved" - Discover.vue

```
// ...
created () {
  // ...
  EventBus.$on('contentSaved', this.updateData)
},
methods: {
  updateData (data) {
    let id = '8'
    if (data && data.contentLink.startsWith(id)) {
      id = data.contentLink
    }
    // ...
  }
},
// ...
```

# Demo preview gadget updates

# Thank you!