Working with Blocks

- get visible
- get trusted
- get engaged
- get in touch
- get converted

the user journey

#epi2012
episerver.com/epi2012
[CultureSpecific]
[Display(
    GroupName = SystemTabNames.Content,
    Name = "Heading",
    Description = "The heading of the block.",
    Order = 1)]
public virtual string Heading { get; set; }
Edit "Teaser"

General

Name: TeaserBlock
Display name: Teaser
Description: Simple block that displays a main heading
Sort index: 100
Available in Edit mode

Display Template

Web Form template: TeaserBlockControl
MVC template: 

Advanced

Guid: 9e4c0d85-1139-48ea-b905-67c08bee3c69

Delete Custom Data

Save Cancel
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using EPiServer.DataAbstraction;
using EPiServer.DataAnnotations;
using System.ComponentModel.DataAnnotations;
using EPiServer.Core;

{
    [ContentType(GUID = "9E4C0D85-1139-48EA-B905-67C08BEE3C69"),
    DisplayName = "Teaser",
    Description = "Simple block that displays a main heading and a XHTML text.",
    GroupName = "AlloyTech",
    Order = 100)]
    public class TeaserBlock : BlockData
    {
        [CultureSpecific]
        [Display(
            GroupName = SystemTabNames.Content,
            Name = "Heading",
            Description = "The heading of the block.",
            Order = 1)]
        public virtual string Heading { get; set; }

        [CultureSpecific]
        [Display(
            GroupName = SystemTabNames.Content,
            Name = "Main body",
            Description = "Content of this block",
            Order = 2)]
        public virtual XhtmlString MainBody { get; set; }
    }
}
Organization and reuse
Assembling Pages Using Blocks

Basic Steps in CMS 7:
1. Open a Panel
2. Search/Find your Block
3. Drag/Drop on to a Page

Creating a block:
Visual Studio
> Add New Item…
> Select the EPiServer node
> Select the BlockType template
Demonstration #1
Concepts
Display Channels

Display Channels enables us to render content based on the request and which browser the user is using.

When you are logged in – you can preview what these might look like by setting the appropriate resolution.

There are two basic approaches to using Display Channels:
1. **Multiple Templates** - (WebForm, UserControl, View or partial view) that should be used to render depending on the request.

2. **Single Template** - In that template check which channels are active and then control the output (for example which style sheet to use) depending on which channel is active.
Display Channels

If we have two different templates registered for same type and one template has a tag that matches the ChannelName then that template will be used when the channel is active.

```csharp
public class MyBlock : BlockData {}

[RenderDescriptor(Default=true)]
public partial class MyBlockControl : BlockControlBase<MyBlock>
{
}

[RenderDescriptor(Tags=new string[]{TagConstants.Mobile})]
public partial class MyBlockMobileControl : BlockControlBase<MyBlock>
{
}
```
SDK
Best Practices

The web application’s root folder contains a directory called `Templates`.

Each Website in the application falls under a sub-folder that is the name of the application or website.

/Blocks
/BlockTypes

(and also)

/Pages
/PageTypes

(etc)
Coding Scenario

**Scenario:** Let’s recreate the TeaserBlock while linking the ‘Heading’ to a page

The existing sample, TeaserBlock, has two properties:

```csharp
[CultureSpecific]
[Display(
    GroupName = SystemTabNames.Content,
    Name = "Heading",
    Description = "The heading of the block.",
    Order = 1))
public virtual string Heading { get; set; }

[CultureSpecific]
[Display(
    GroupName = SystemTabNames.Content,
    Name = "Main body",
    Description = "Content of this block",
    Order = 2)]
public virtual XhtmlString MainBody { get; set; }
```

**Task:** Create a new property called, “PageURL”
CurrentBlock.[PropertyName]

With 3 properties, “Heading”, “PageURL”, and “MainBody” we will want to format the display to look something like this:

```html
<a href={PageURL}>
  <b>{Heading}</b>
</a>

{MainBody}
```

Approach:
1. Let’s grab Heading and MainBody using the `<EPiServer:Property>` tag
2. And we’ll also demonstrate how we can assign the value of a block by referencing it directly:

   We now have `CurrentBlock` to access data from the block:

   For example:
   ```jsp
   %= CurrentBlock.PageURL %>
   will display the data for us!
   ```
Coding in Visual Studio
More Powerful…

We can now store modularized data into a new structure called, “BlockData” instead of hiding it behind PageData.

This should also ease the task of migrating websites onto EPiServer because modular data is now more easily defined (and controlled) through blocks and strongly typed code definitions.

The pre-defined properties for BlockTypes are now accessible to the user control using “CurrentBlock”

… And data stored for a block is not specific to a page but the block itself!

We can also search and organize these blocks using a user-definable folder structure that makes organizing blocks simple.
- get visible
- get trusted
- get engaged
- get in touch
- get converted

the user journey