Optimize Your View Layer

Matt Honeycutt
http://trycatchfail.com
mbhoneycutt@gmail.com

pluralsight
hardcore developer training

Our Goals

Eliminate Repetitive Markup

Build a Consistent UI

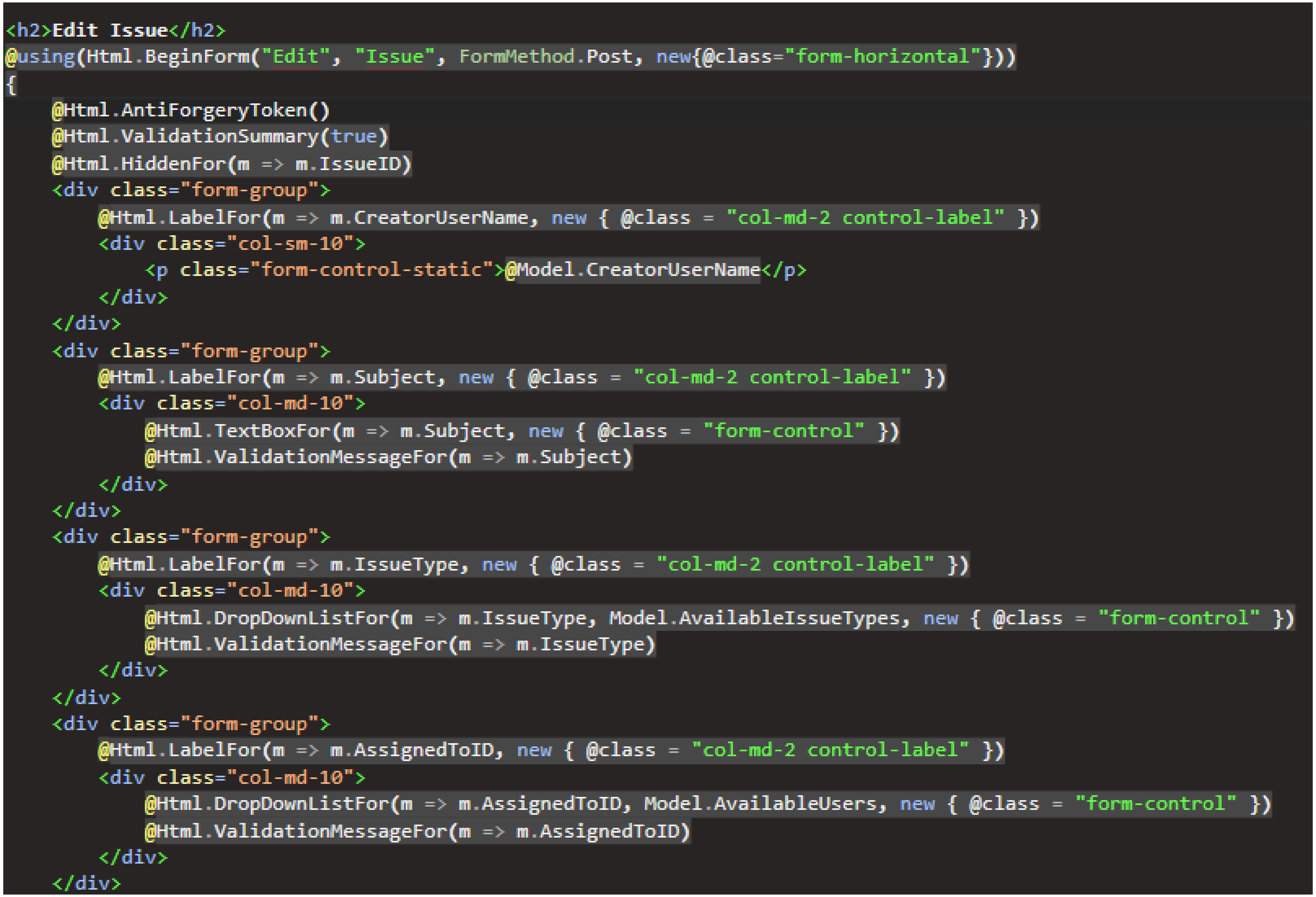
Improve Maintainability

Maintain Flexibility

Outline

- Building Editor Templates
- Encapsulation Through HtmlHelpers
- An Extensible Model Metadata Provider
- Metadata-Driven Conventions
- Putting It All Together – Conventional UI

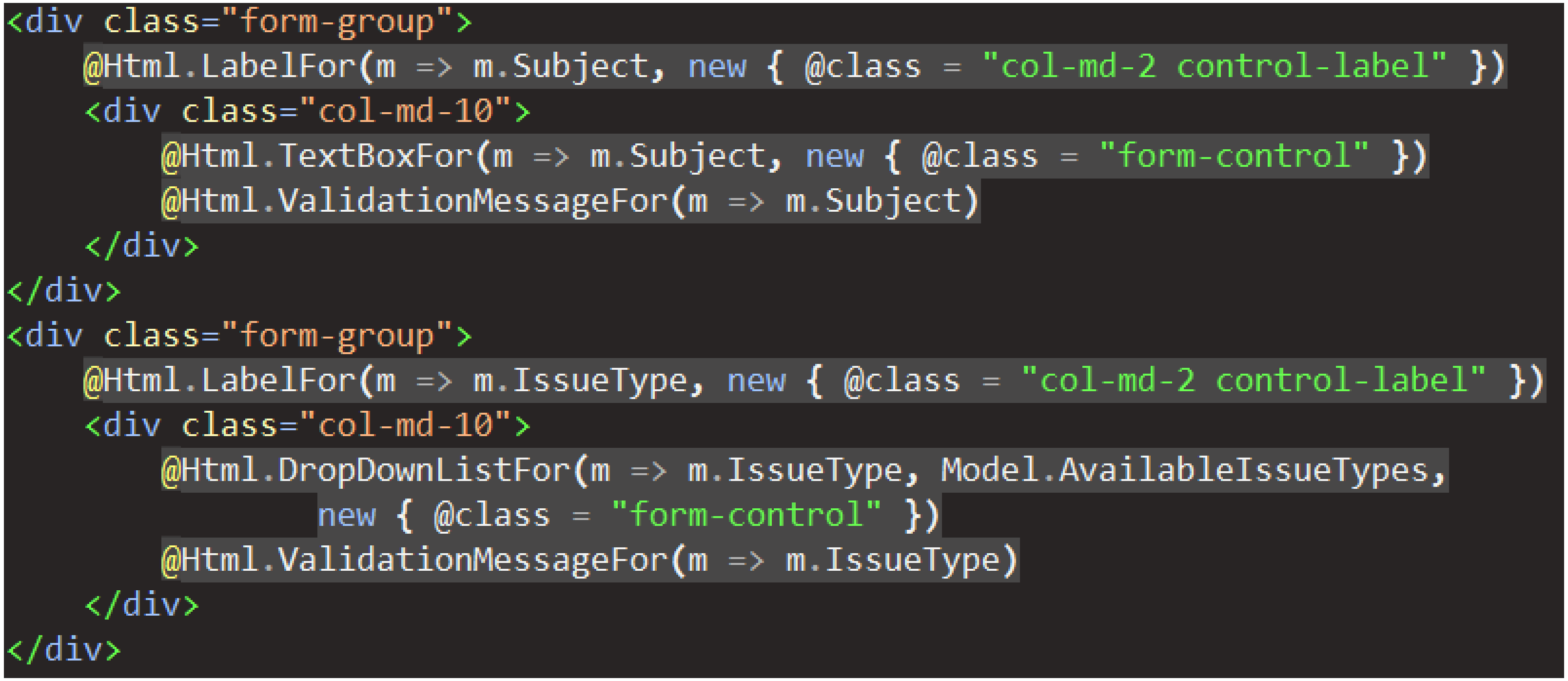
Typical Razor Markup



```
<h2>Edit Issue</h2>
@using(Html.BeginForm("Edit", "Issue", FormMethod.Post, new{@class="form-horizontal"}))
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)
    @Html.HiddenFor(m => m.IssueID)
    <div class="form-group">
        @Html.LabelFor(m => m.CreatorUserName, new { @class = "col-md-2 control-label" })
        <div class="col-sm-10">
            <p class="form-control-static">@Model.CreatorUserName</p>
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.Subject, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Subject, new { @class = "form-control" })
            @Html.ValidationMessageFor(m => m.Subject)
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.IssueType, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.IssueType, Model.AvailableIssueTypes, new { @class = "form-control" })
            @Html.ValidationMessageFor(m => m.IssueType)
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.AssignedToID, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.AssignedToID, Model.AvailableUsers, new { @class = "form-control" })
            @Html.ValidationMessageFor(m => m.AssignedToID)
        </div>
    </div>
```

Problems

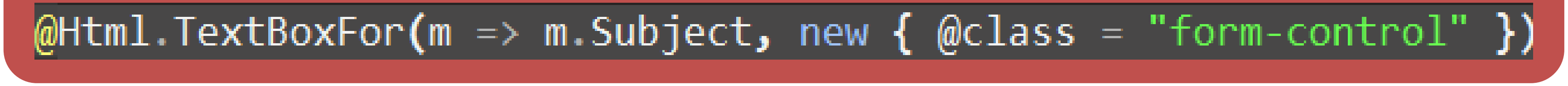
- Repetition
- Bootstrap Classes
- Duplication
- High Cognitive Cost



```
<div class="form-group">
    @Html.LabelFor(m => m.Subject, new { @class = "col-md-2 control-label" })
    <div class="col-md-10">
        @Html.TextBoxFor(m => m.Subject, new { @class = "form-control" })
        @Html.ValidationMessageFor(m => m.Subject)
    </div>
</div>
<div class="form-group">
    @Html.LabelFor(m => m.IssueType, new { @class = "col-md-2 control-label" })
    <div class="col-md-10">
        @Html.DropDownListFor(m => m.IssueType, Model.AvailableIssueTypes,
                new { @class = "form-control" })
        @Html.ValidationMessageFor(m => m.IssueType)
    </div>
</div>
```

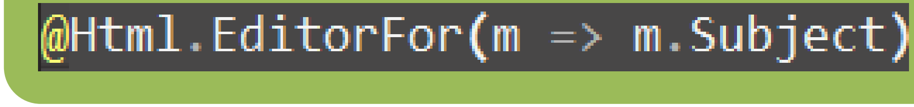
Editor Templates

Typical Markup



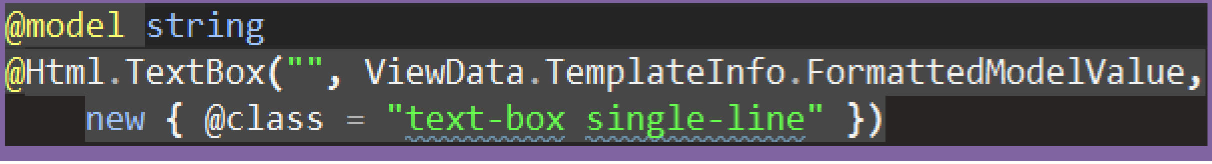
```
@Html.TextBoxFor(m => m.Subject, new { @class = "form-control" })
```

EditorFor Markup



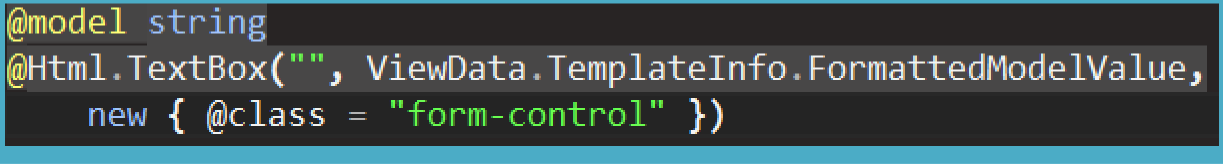
```
@Html.EditorFor(m => m.Subject)
```

Standard String Template



```
@model string
@Html.TextBox("", ViewData.TemplateInfo.FormattedModelValue,
    new { @class = "text-box single-line" })
```

Custom, Bootstrap-Friendly Template



```
@model string
@Html.TextBox("", ViewData.TemplateInfo.FormattedModelValue,
    new { @class = "form-control" })
```

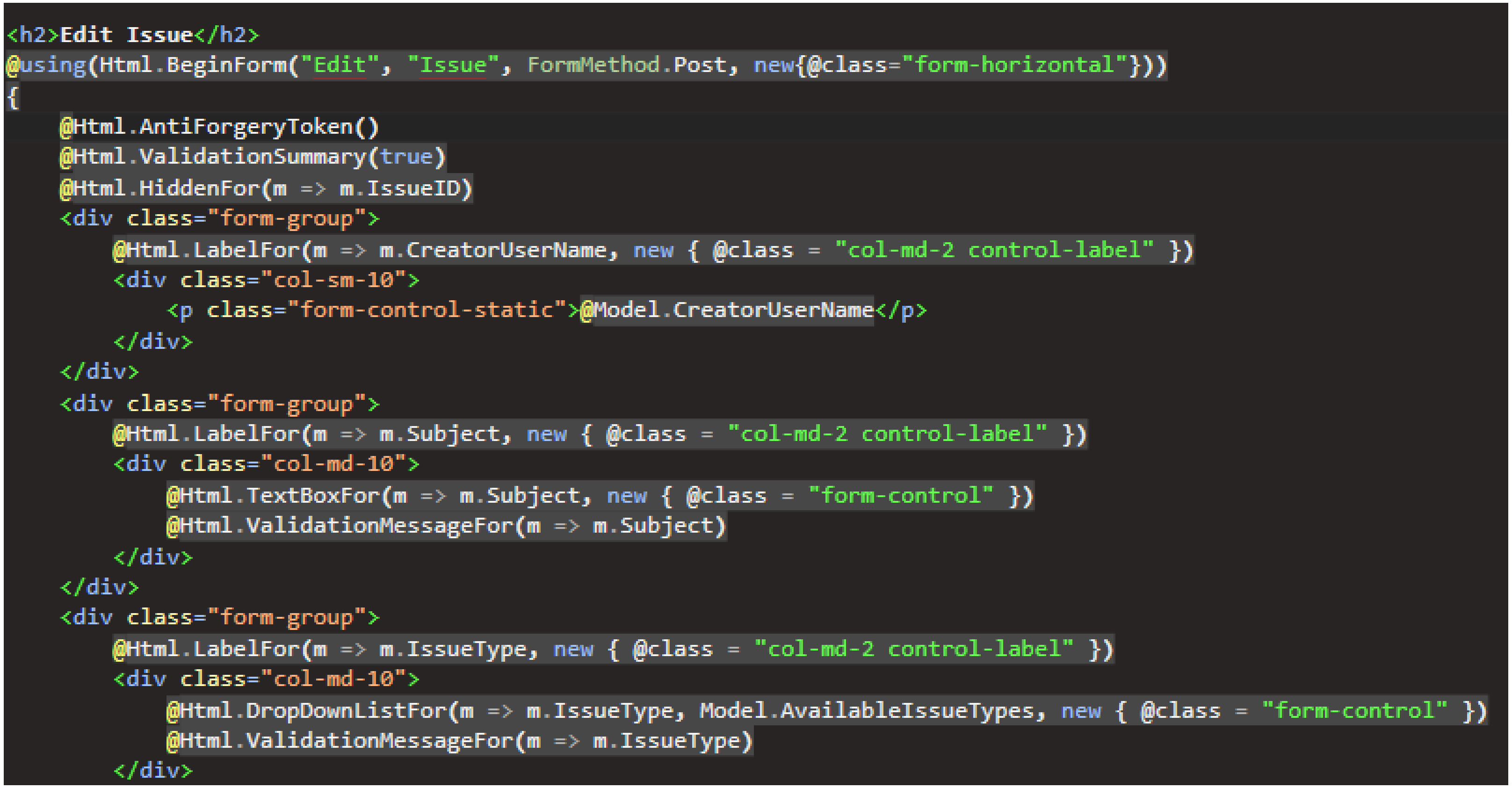
Editor Template Advantages

Streamlined Development Workflow

Consistency

Encapsulation

Simplified Markup



```
<h2>Edit Issue</h2>
@using(Html.BeginForm("Edit", "Issue", FormMethod.Post, new{@class="form-horizontal"}))
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)
    @Html.HiddenFor(m => m.IssueID)
    <div class="form-group">
        @Html.LabelFor(m => m.CreatorUserName, new { @class = "col-md-2 control-label" })
        <div class="col-sm-10">
            <p class="form-control-static">@Model.CreatorUserName</p>
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.Subject, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Subject, new { @class = "form-control" })
            @Html.ValidationMessageFor(m => m.Subject)
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.IssueType, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.IssueType, Model.AvailableIssueTypes, new { @class = "form-control" })
            @Html.ValidationMessageFor(m => m.IssueType)
        </div>
```

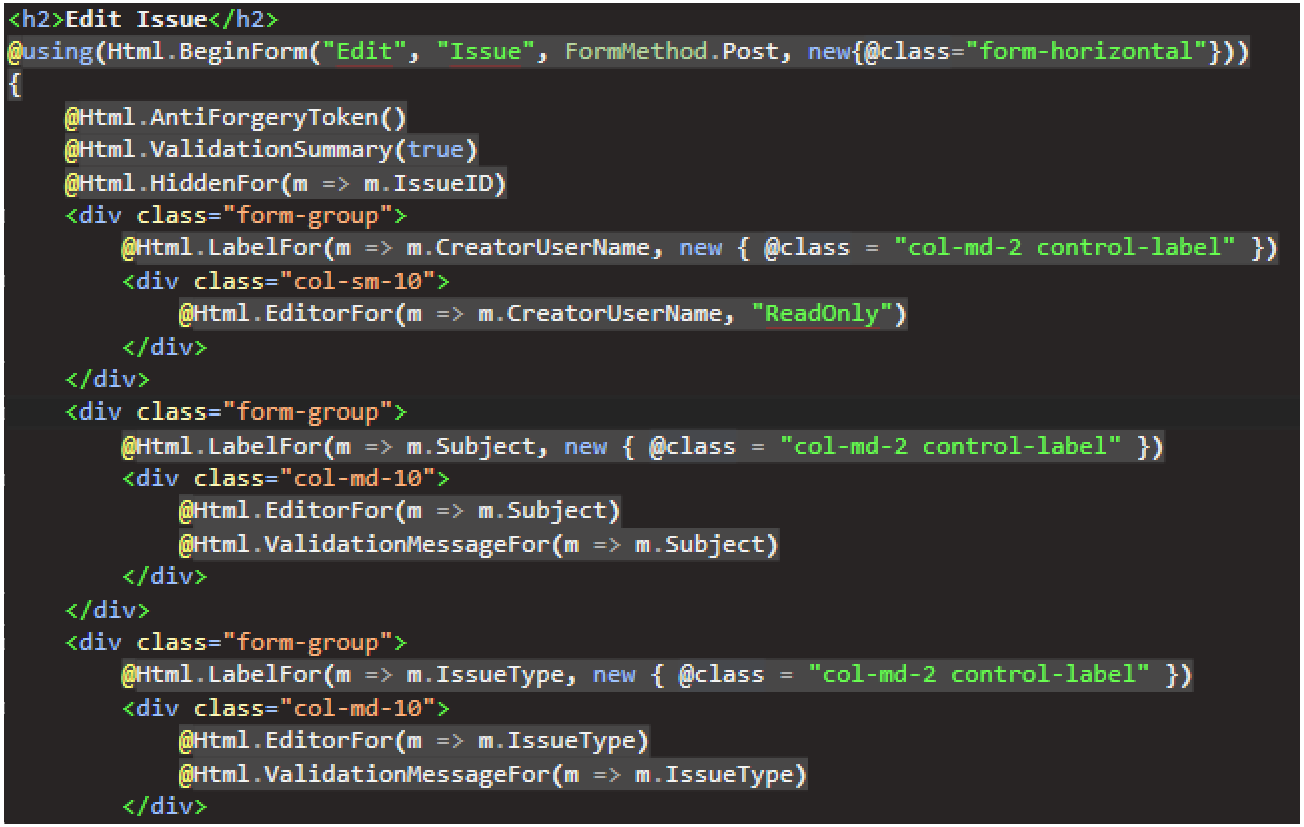
Editor Template Advantages

Streamlined Development Workflow

Consistency

Encapsulation

Simplified Markup



```
<h2>Edit Issue</h2>
@using(Html.BeginForm("Edit", "Issue", FormMethod.Post, new{@class="form-horizontal"}))
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)
    @Html.HiddenFor(m => m.IssueID)
    <div class="form-group">
        @Html.LabelFor(m => m.CreatorUserName, new { @class = "col-md-2 control-label" })
        <div class="col-sm-10">
            @Html.EditorFor(m => m.CreatorUserName, "ReadOnly")
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.Subject, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.EditorFor(m => m.Subject)
            @Html.ValidationMessageFor(m => m.Subject)
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.IssueType, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.EditorFor(m => m.IssueType)
            @Html.ValidationMessageFor(m => m.IssueType)
        </div>
```

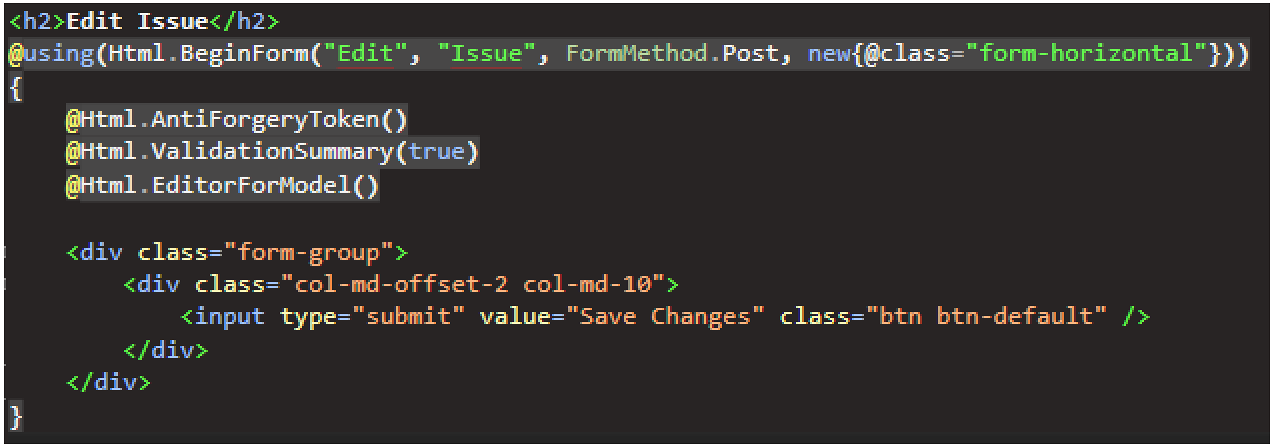
Editor Template Advantages

Streamlined Development Workflow

Consistency

Encapsulation

Simplified Markup

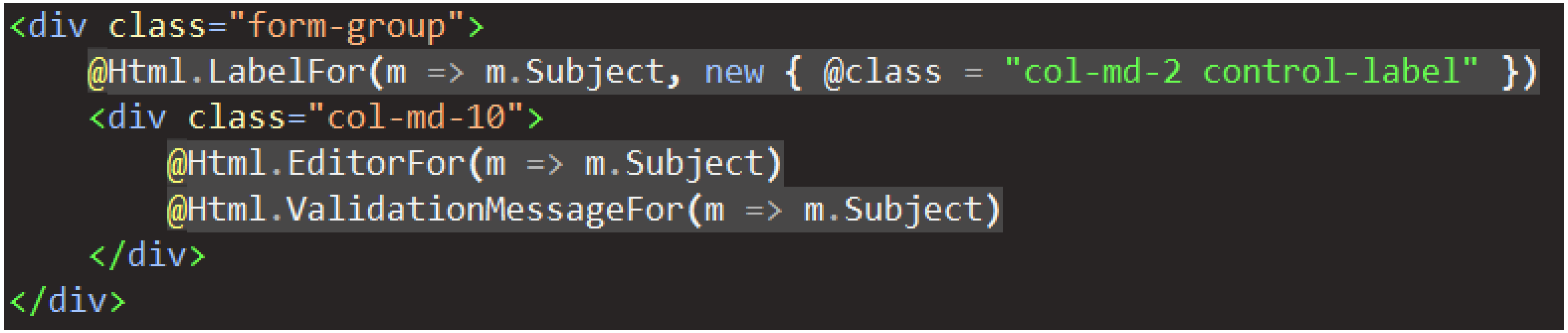


```
<h2>Edit Issue</h2>
@using(Html.BeginForm("Edit", "Issue", FormMethod.Post, new{@class="form-horizontal"}))
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)
    @Html.EditorForModel()

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Save Changes" class="btn btn-default" />
        </div>
    </div>
}
```

What About Labels?

Problems

- Classes Required Everywhere
- Not Future-Proof

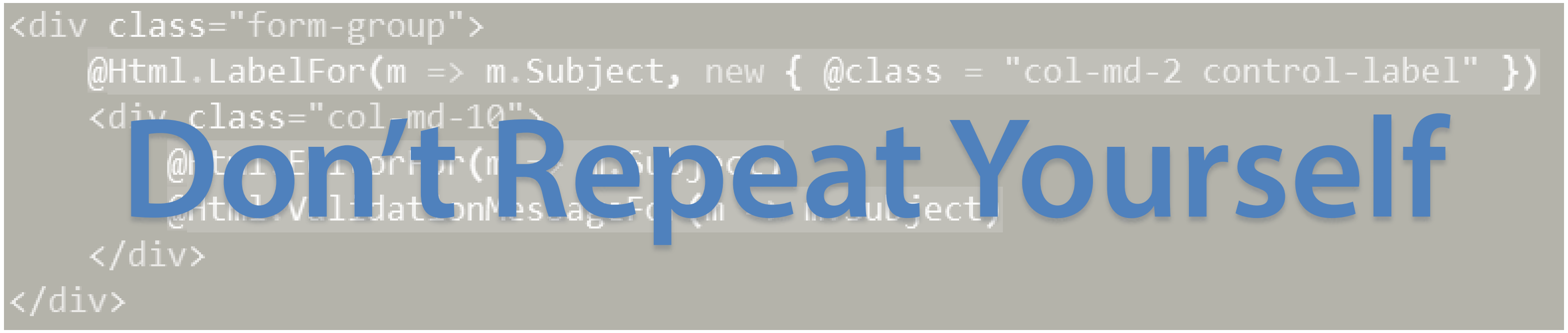


```
<div class="form-group">
    @Html.LabelFor(m => m.Subject, new { @class = "col-md-2 control-label" })
    <div class="col-md-10">
        @Html.EditorFor(m => m.Subject)
        @Html.ValidationMessageFor(m => m.Subject)
    </div>
</div>
```

What About Labels?

Problems

- Classes Required Everywhere
- Not Future-Proof



```
<div class="form-group">
    @Html.LabelFor(m => m.Subject, new { @class = "col-md-2 control-label" })
    <div class="col-md-10">
        @Html.EditorFor(m => m.Subject)
        @Html.ValidationMessageFor(m => m.Subject)
    </div>
</div>
```

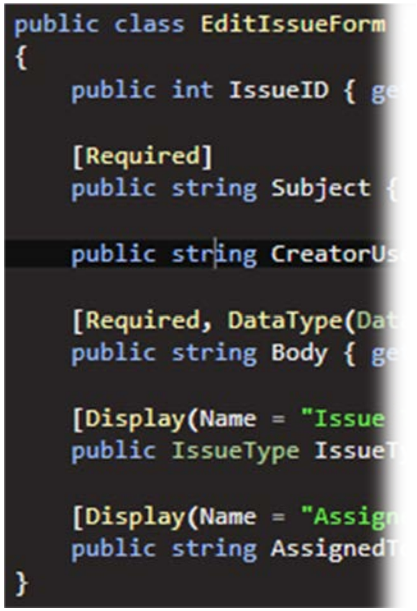
Don’t Repeat Yourself

Solutions

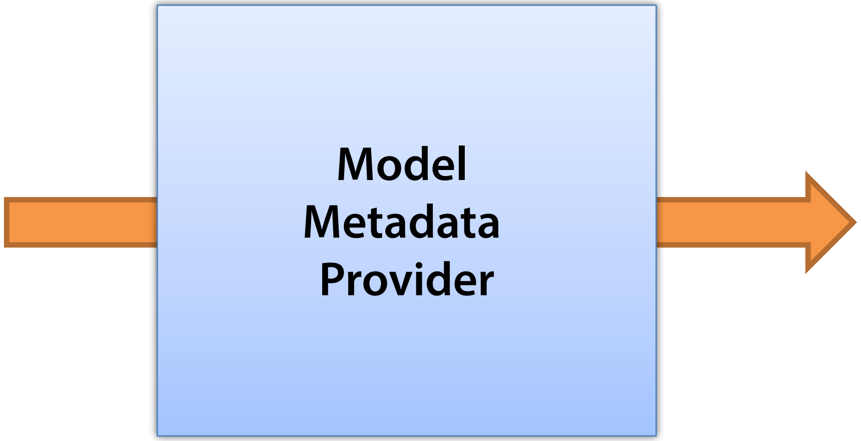
- ~~Label Templates~~
- HtmlHelpers

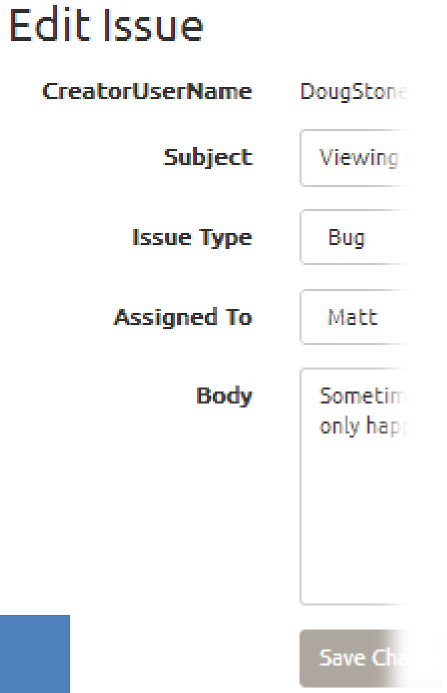
MVC Fundamentals: http://goo.gl/Mmz9

Model Metadata



```
public class EditIssueForm
{
    public int IssueID { ge

    [Required]
    public string Subject {

    public string CreatorUs

    [Required, DataType(Dat
    public string Body { ge

    [Display(Name = "Issue
    public IssueType IssueT

    [Display(Name = "Assign
    public string AssignedT
}
```





Edit Issue

CreatorUserName DougStone

Subject Viewing

Issue Type Bug

Assigned To Matt

Body Sometim
only happ

Model Metadata

Data Type	Display Hint	Display Name

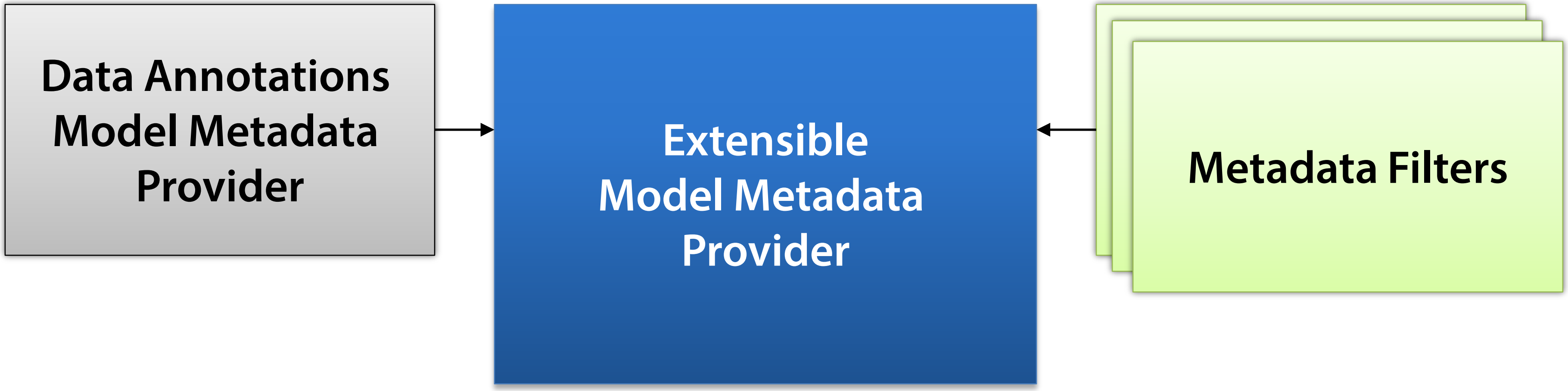
Model Metadata Conventions

Data Annotation Attributes

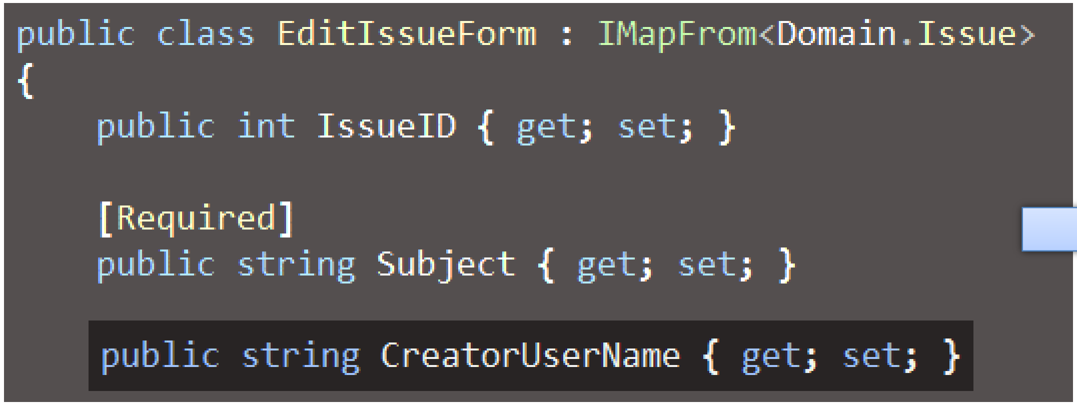
- Simple
- Must be applied where needed

Custom Model Metadata Provider

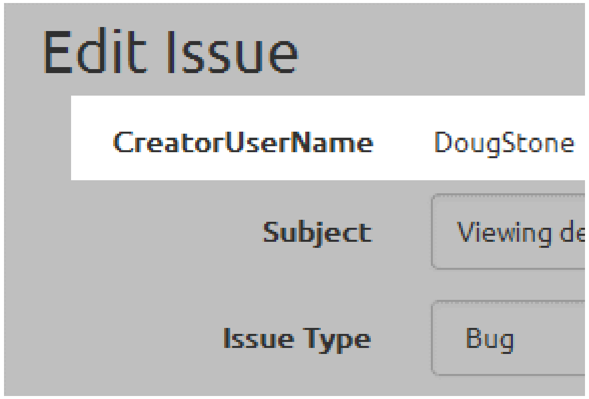
- Very flexible
- Complex


Data Annotations Model Metadata Provider
Extensible Model Metadata Provider
Metadata Filters

Labels (Again!)



```
public class EditIssueForm : IMapFrom<Domain.Issue>
{
    public int IssueID { get; set; }

    [Required]
    public string Subject { get; set; }

    public string CreatorUserName { get; set; }
```

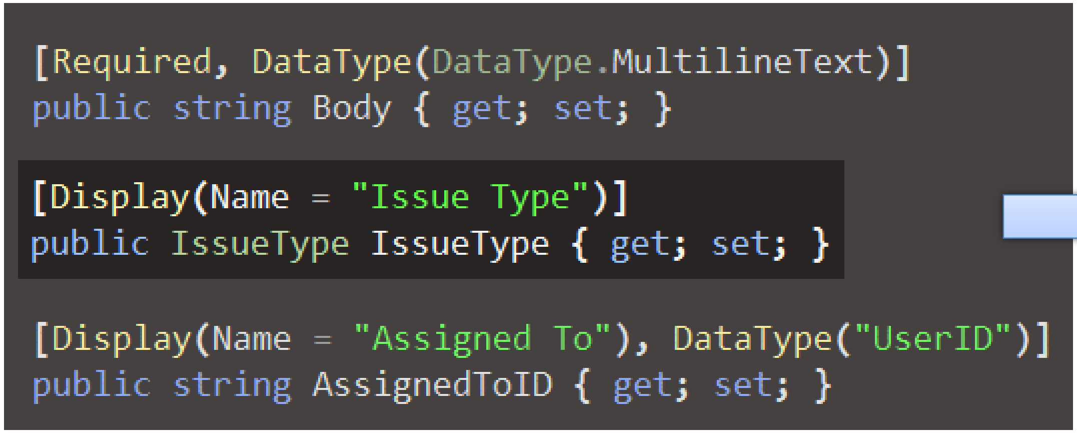


Edit Issue

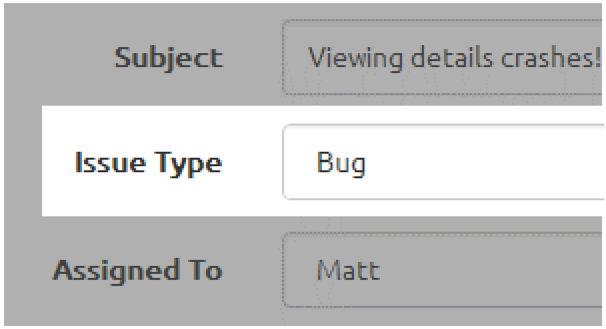
CreatorUserName DougStone

Subject Viewing de

Issue Type Bug



```
[Required, DataType(DataType.MultilineText)]
public string Body { get; set; }

[Display(Name = "Issue Type")]
public IssueType IssueType { get; set; }

[Display(Name = "Assigned To"), DataType("UserID")]
public string AssignedToID { get; set; }
```



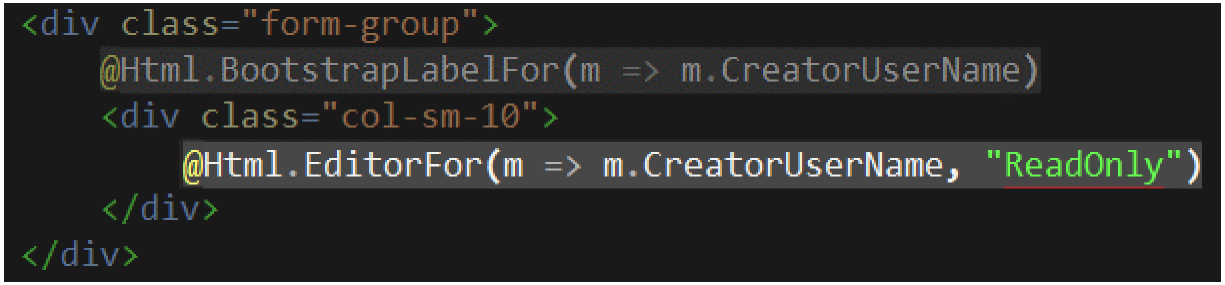
Subject Viewing details crashes!

Issue Type Bug

Assigned To Matt

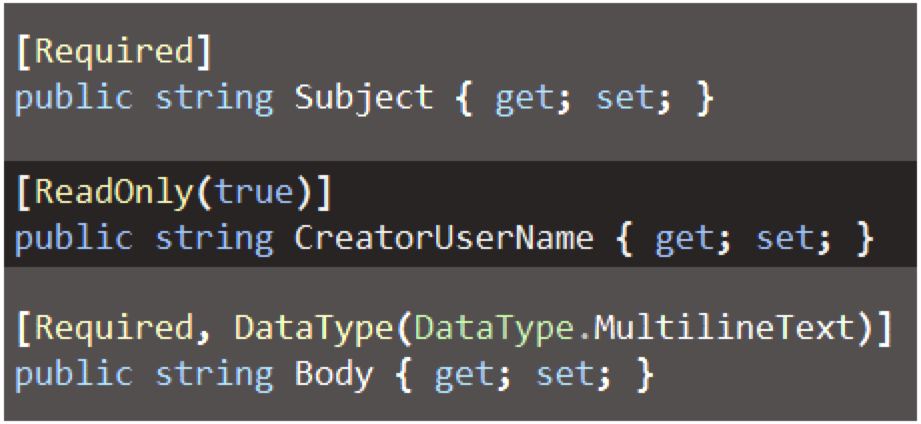
Read-Only Editor Selection

View-Based Selection



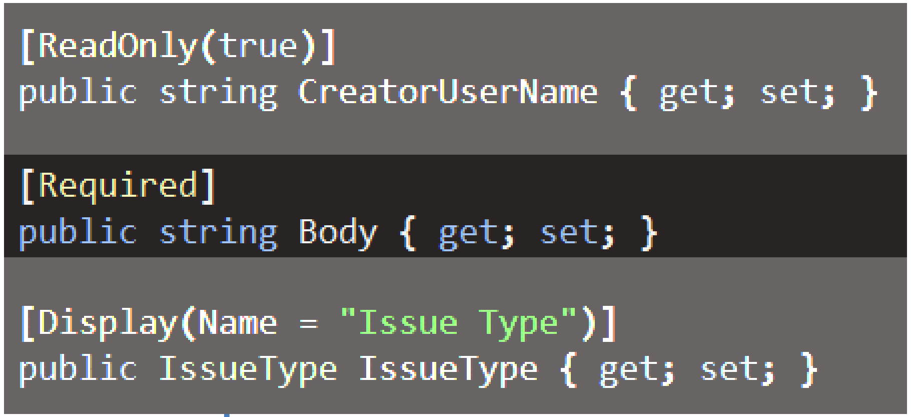
```
<div class="form-group">
    @Html.BootstrapLabelFor(m => m.CreatorUserName)
    <div class="col-sm-10">
        @Html.EditorFor(m => m.CreatorUserName, "ReadOnly")
    </div>
</div>
```

Better: Model-Based Selection



```
[Required]
public string Subject { get; set; }

[ReadOnly(true)]
public string CreatorUserName { get; set; }

[Required, DataType(DataType.MultilineText)]
public string Body { get; set; }
```

Choose Templates Based On Naming Conventions

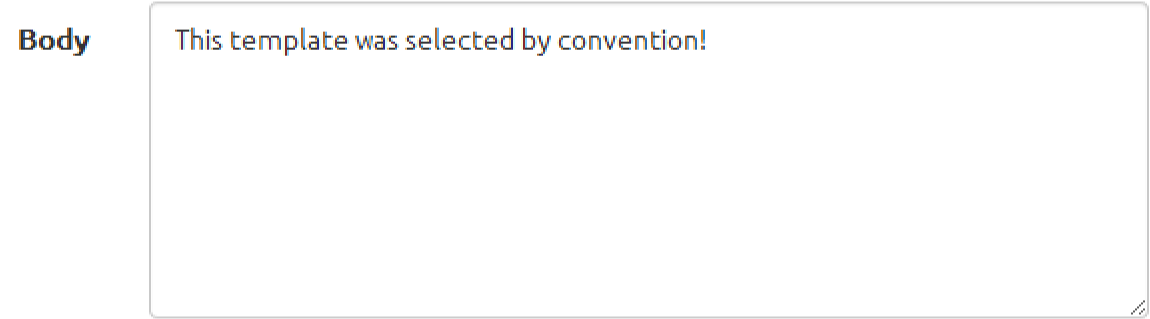


```
[ReadOnly(true)]
public string CreatorUserName { get; set; }

[Required]
public string Body { get; set; }

[Display(Name = "Issue Type")]
public IssueType IssueType { get; set; }
```

Filter

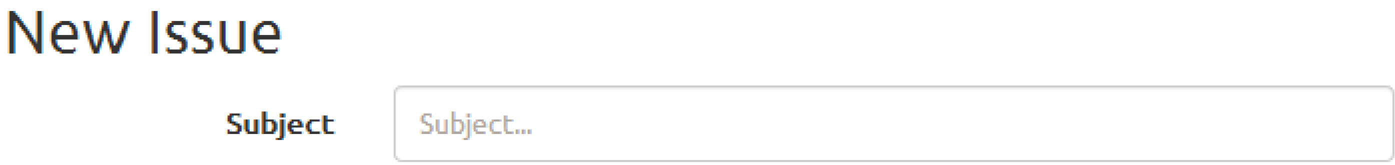
metadata.PropertyName.Contains(*“Body”*) ?

DataType = “ *MultilineText*”



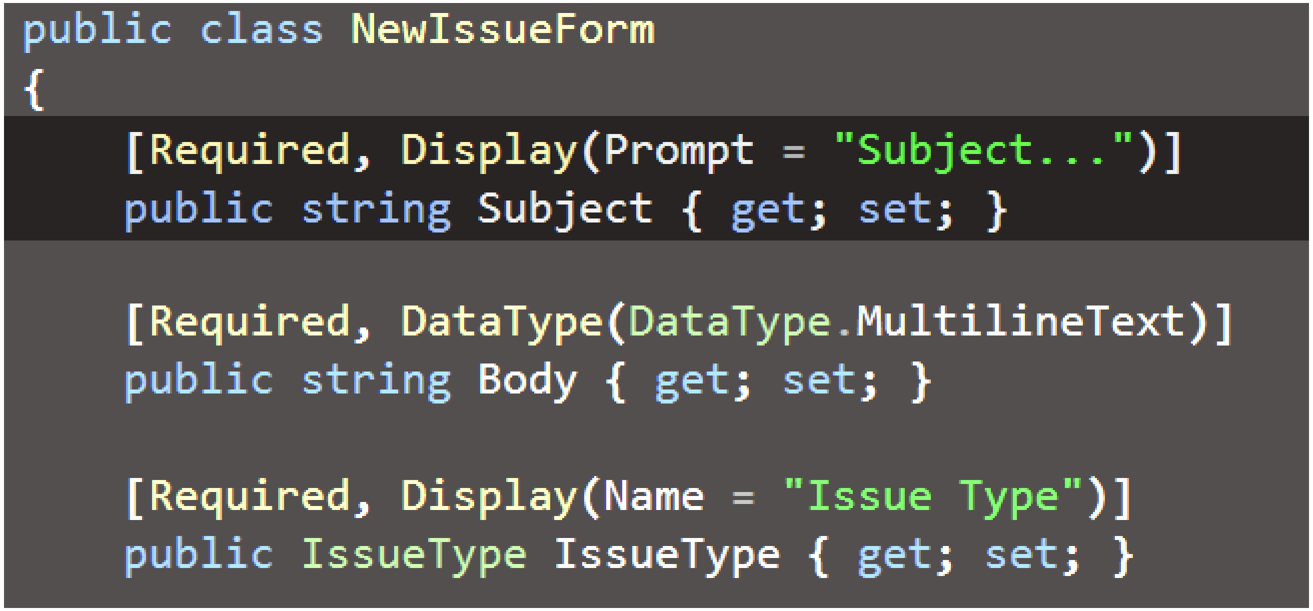
This template was selected by convention!

Automatic Field Watermarks



New Issue

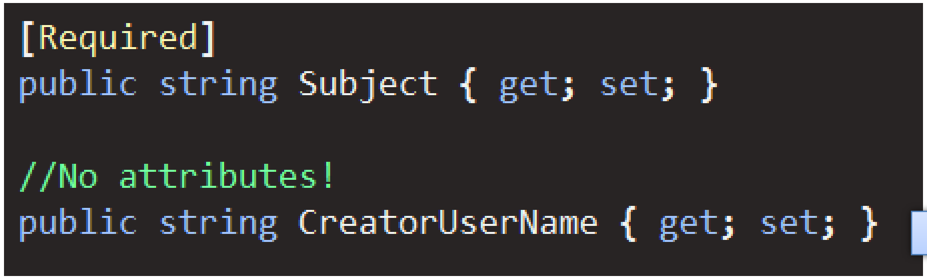
Subject: Subject...



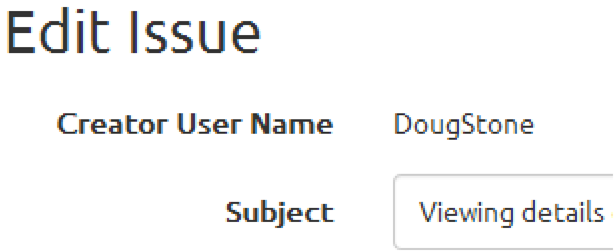
```
public class NewIssueForm
{
    [Required, Display(Prompt = "Subject...")]
    public string Subject { get; set; }

    [Required, DataType(DataType.MultilineText)]
    public string Body { get; set; }

    [Required, Display(Name = "Issue Type")]
    public IssueType IssueType { get; set; }
```

“Overridden Progressively”

Convention: “Labels are built from Pascal-cased property names”



```
[Required]
public string Subject { get; set; }

//No attributes!
public string CreatorUserName { get; set; }
```

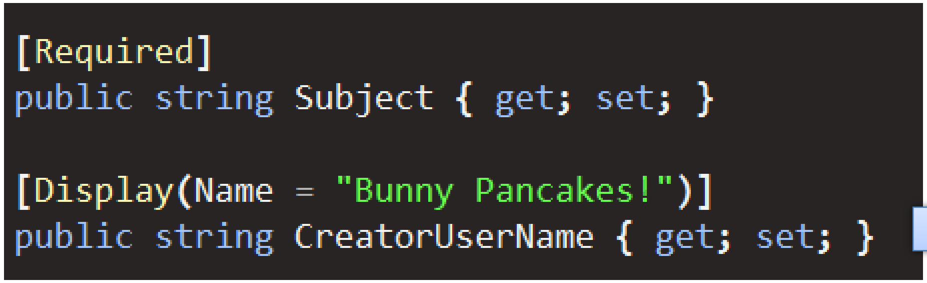


Edit Issue

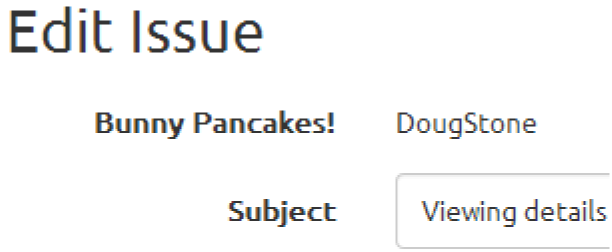
Creator User Name DougStone

Subject Viewing details

Override: “I want the label for *this* property to be ‘Bunny Pancakes!’”



```
[Required]
public string Subject { get; set; }

[Display(Name = "Bunny Pancakes!")]
public string CreatorUserName { get; set; }
```



Edit Issue

Bunny Pancakes! DougStone

Subject Viewing details

Conventional UI

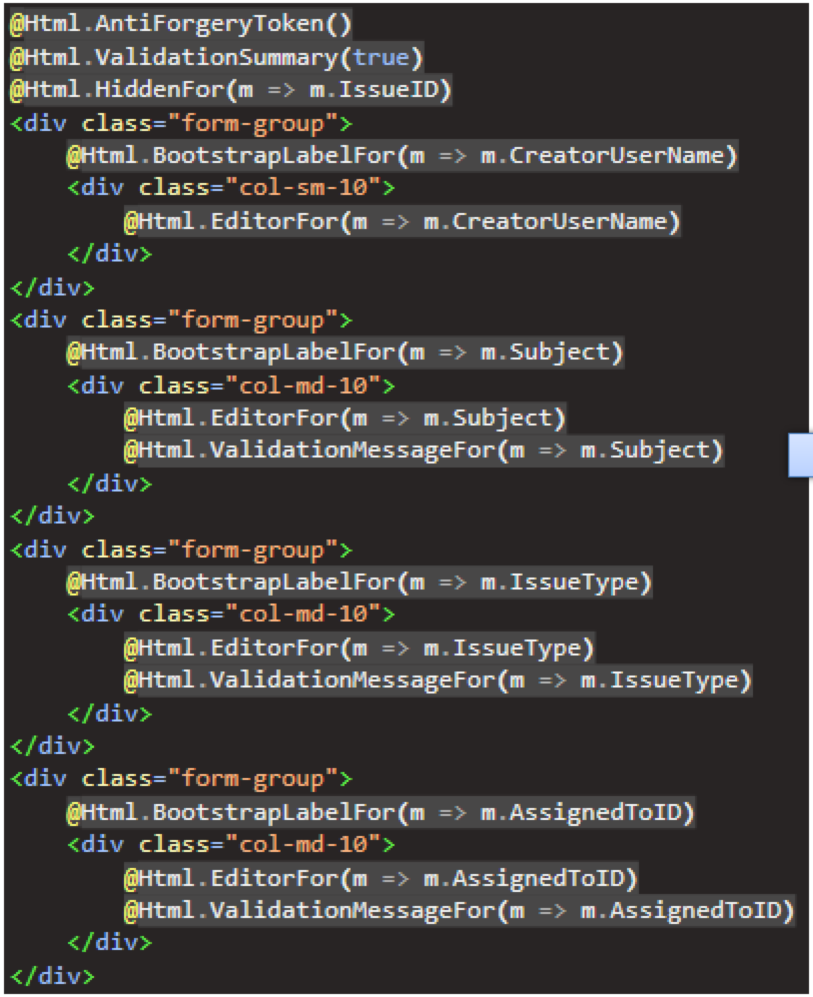
HTML Helpers

Editor Templates

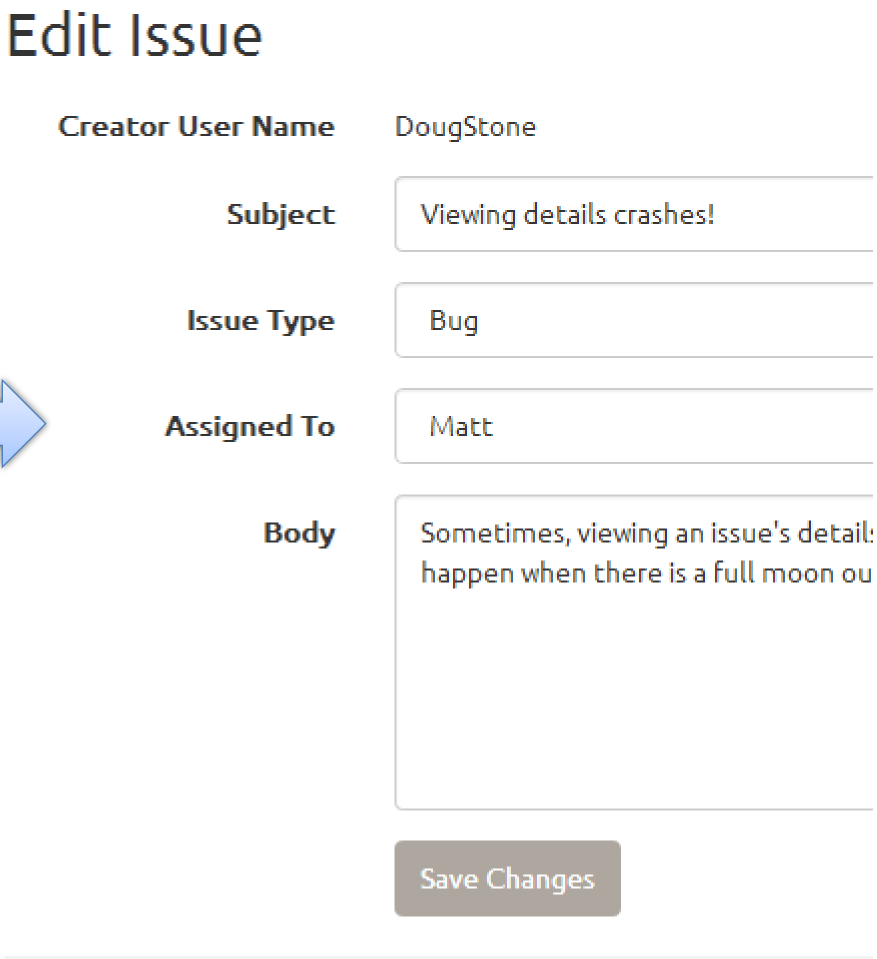
Model Metadata

Convention-Driven UI!

An Object Template



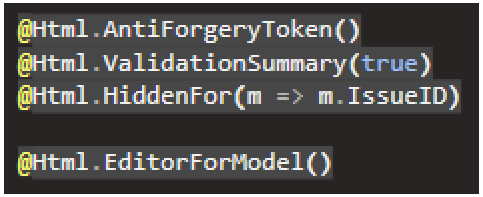
```
@Html.AntiForgeryToken()
@Html.ValidationSummary(true)
@Html.HiddenFor(m => m.IssueID)
<div class="form-group">
    @Html.BootstrapLabelFor(m => m.CreatorUserName)
    <div class="col-sm-10">
        @Html.EditorFor(m => m.CreatorUserName)
    </div>
</div>
<div class="form-group">
    @Html.BootstrapLabelFor(m => m.Subject)
    <div class="col-md-10">
        @Html.EditorFor(m => m.Subject)
        @Html.ValidationMessageFor(m => m.Subject)
    </div>
</div>
<div class="form-group">
    @Html.BootstrapLabelFor(m => m.IssueType)
    <div class="col-md-10">
        @Html.EditorFor(m => m.IssueType)
        @Html.ValidationMessageFor(m => m.IssueType)
    </div>
</div>
<div class="form-group">
    @Html.BootstrapLabelFor(m => m.AssignedToID)
    <div class="col-md-10">
        @Html.EditorFor(m => m.AssignedToID)
        @Html.ValidationMessageFor(m => m.AssignedToID)
    </div>
</div>
```



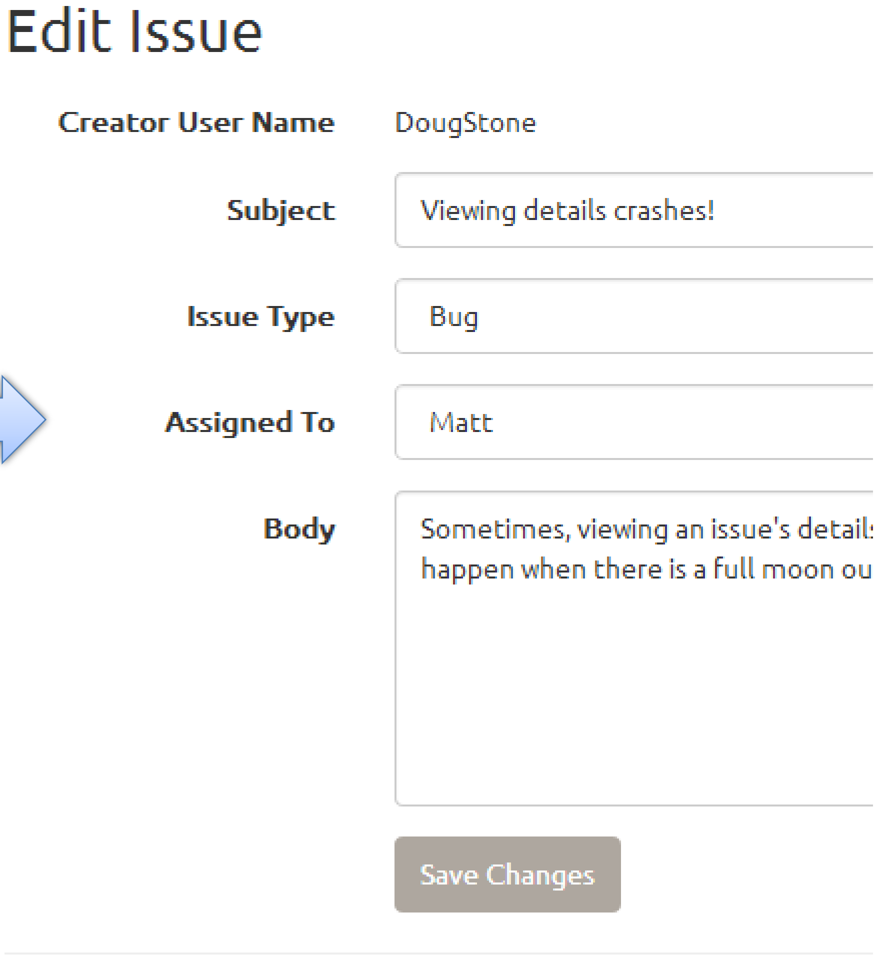
Edit Issue

Creator User Name	DougStone
Subject	Viewing details crashes!
Issue Type	Bug
Assigned To	Matt
Body	Sometimes, viewing an issue's detail happen when there is a full moon ou

An Object Template



```
@Html.AntiForgeryToken()
@Html.ValidationSummary(true)
@Html.HiddenFor(m => m.IssueID)

@Html.EditorForModel()
```



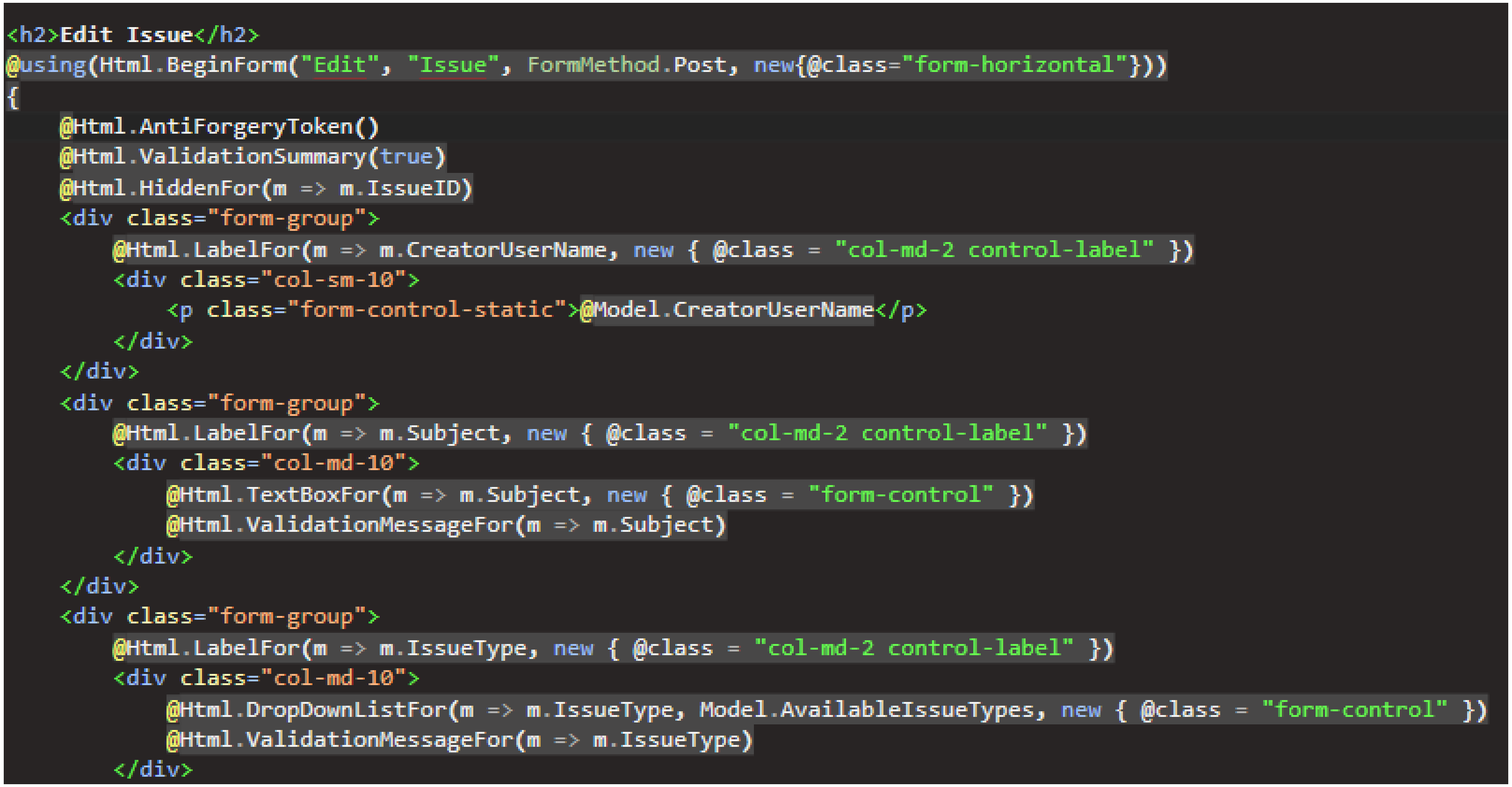
Edit Issue

Creator User Name	DougStone
Subject	Viewing details crashes!
Issue Type	Bug
Assigned To	Matt
Body	Sometimes, viewing an issue's detail happen when there is a full moon ou

Summary

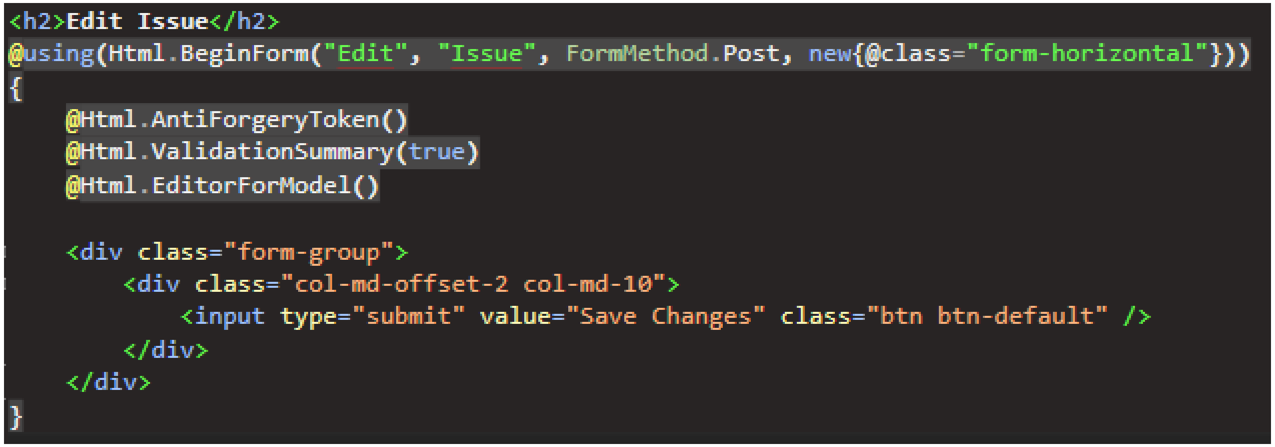
- ☑ Editor Templates
- ☑ Custom HtmlHelpers
- ☑ Model Metadata System
- ☑ Custom Conventions
- ☑ Object Editor Template

Before & After



```
<h2>Edit Issue</h2>
@using(Html.BeginForm("Edit", "Issue", FormMethod.Post, new{@class="form-horizontal"}))
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)
    @Html.HiddenFor(m => m.IssueID)
    <div class="form-group">
        @Html.LabelFor(m => m.CreatorUserName, new { @class = "col-md-2 control-label" })
        <div class="col-sm-10">
            <p class="form-control-static">@Model.CreatorUserName</p>
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.Subject, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Subject, new { @class = "form-control" })
            @Html.ValidationMessageFor(m => m.Subject)
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.IssueType, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.DropDownListFor(m => m.IssueType, Model.AvailableIssueTypes, new { @class = "form-control" })
            @Html.ValidationMessageFor(m => m.IssueType)
        </div>
```

Before & After



```
<h2>Edit Issue</h2>
@using(Html.BeginForm("Edit", "Issue", FormMethod.Post, new{@class="form-horizontal"}))
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)
    @Html.EditorForModel()

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Save Changes" class="btn btn-default" />
        </div>
    </div>
}
```

Up next: Streamline your JavaScrip