Building for Reusability with View Components

Alex Wolf

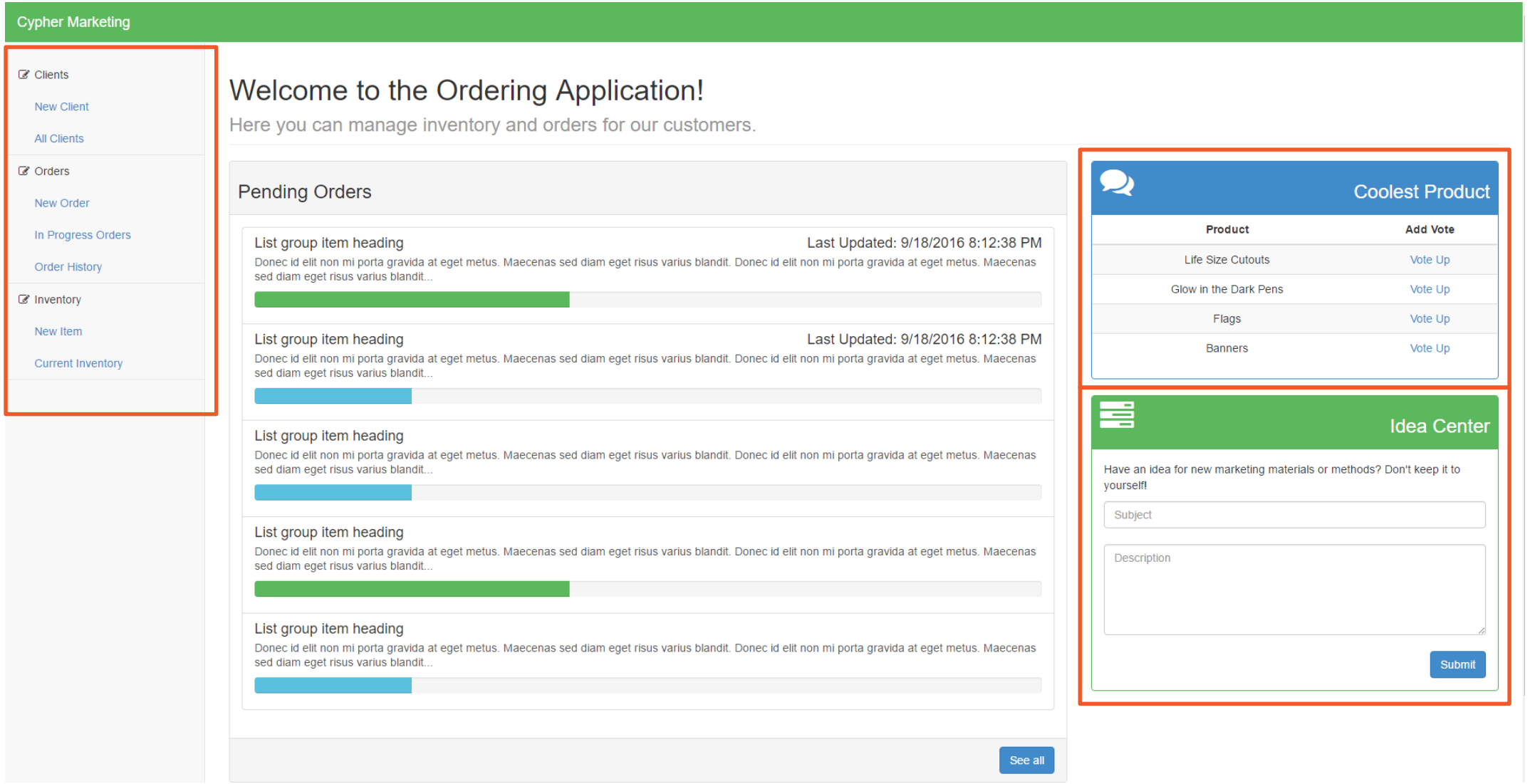
www.alexwolfthoughts.com

To-Do List

- Understanding Partial Views in Razor
- Demo: Working with Partial Views
- The Evolution of View Components
- Demo: Building a Simple View Component
- Demo: Understanding Reusability
- Demo: Understanding Isolation
- Demo: Adding User Interaction and Ajax

Working with Partial Views in Razor

Managing Markup with Partial Views



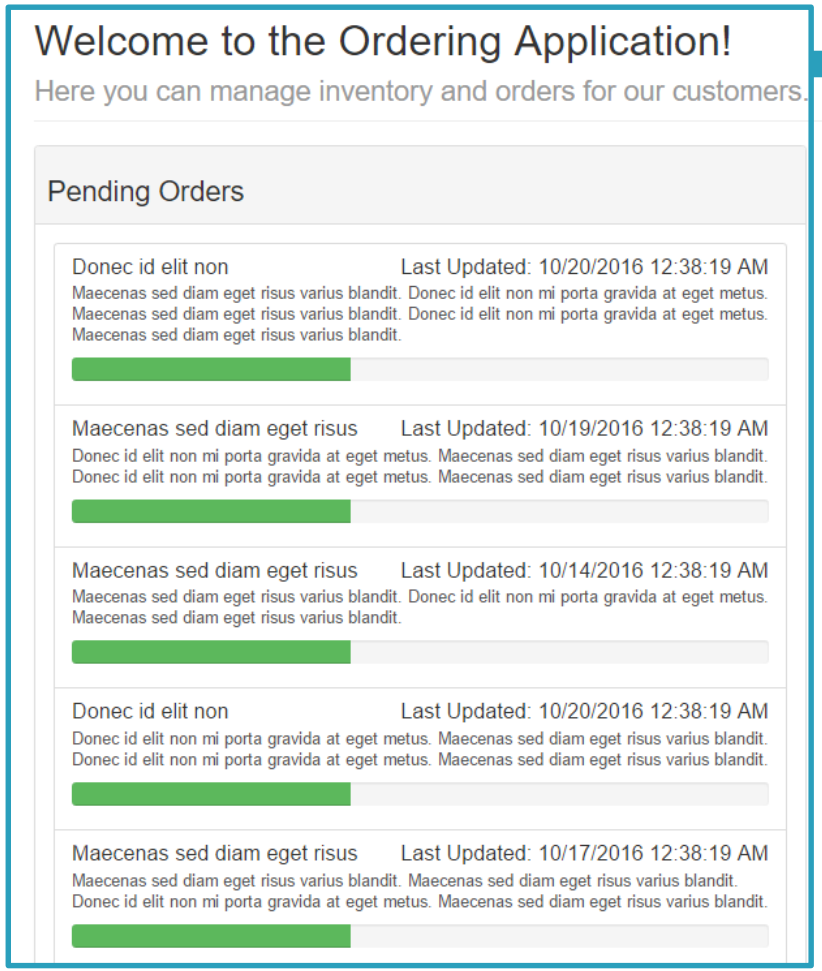
```
<div id="widgets">
    <div class="widget-container">
        @Html.Partial("FeedbackWidget")
    </div>

    <div class="widget-container">
        @Html.Partial("TagCloud", tagModel)
    </div>
</div>
```

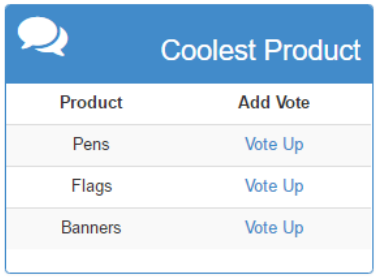
- Renders a Partial View

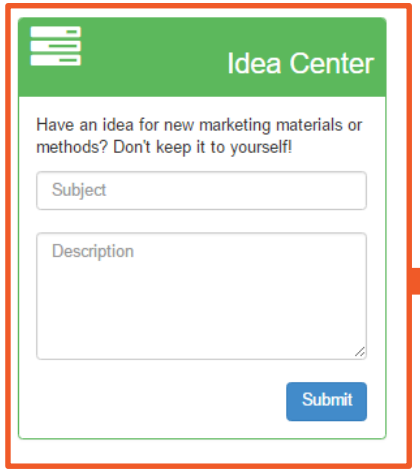
- Renders a Partial View and passes in a Model instance

Partial Views and Action Methods

Index.cshtml







HomeController.cs

```
public IActionResult Index()
{
        return View();
}
```

```
// No Action Method
```

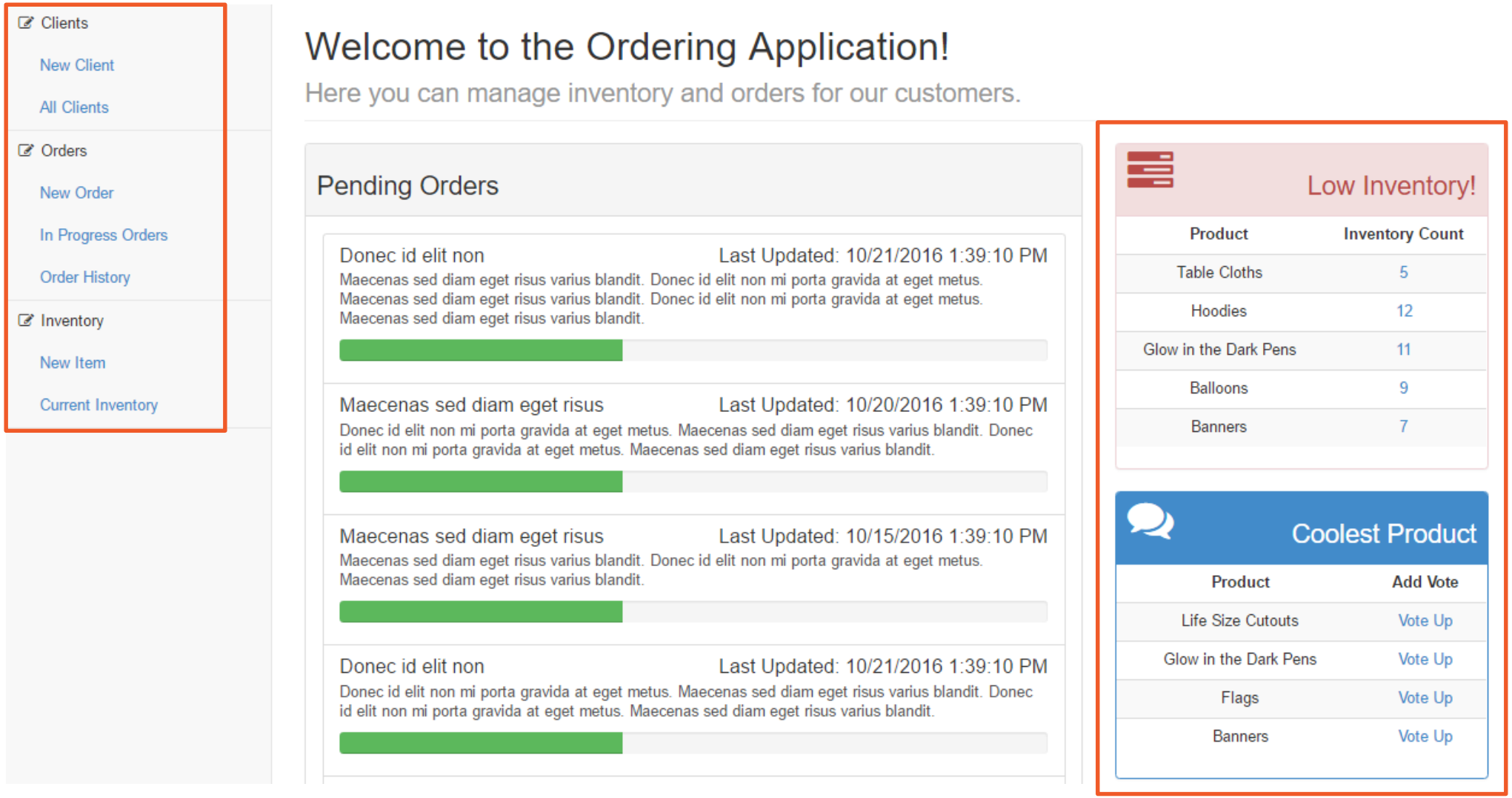
Demo

Identifying areas of an application that are appropriate for Partial Views

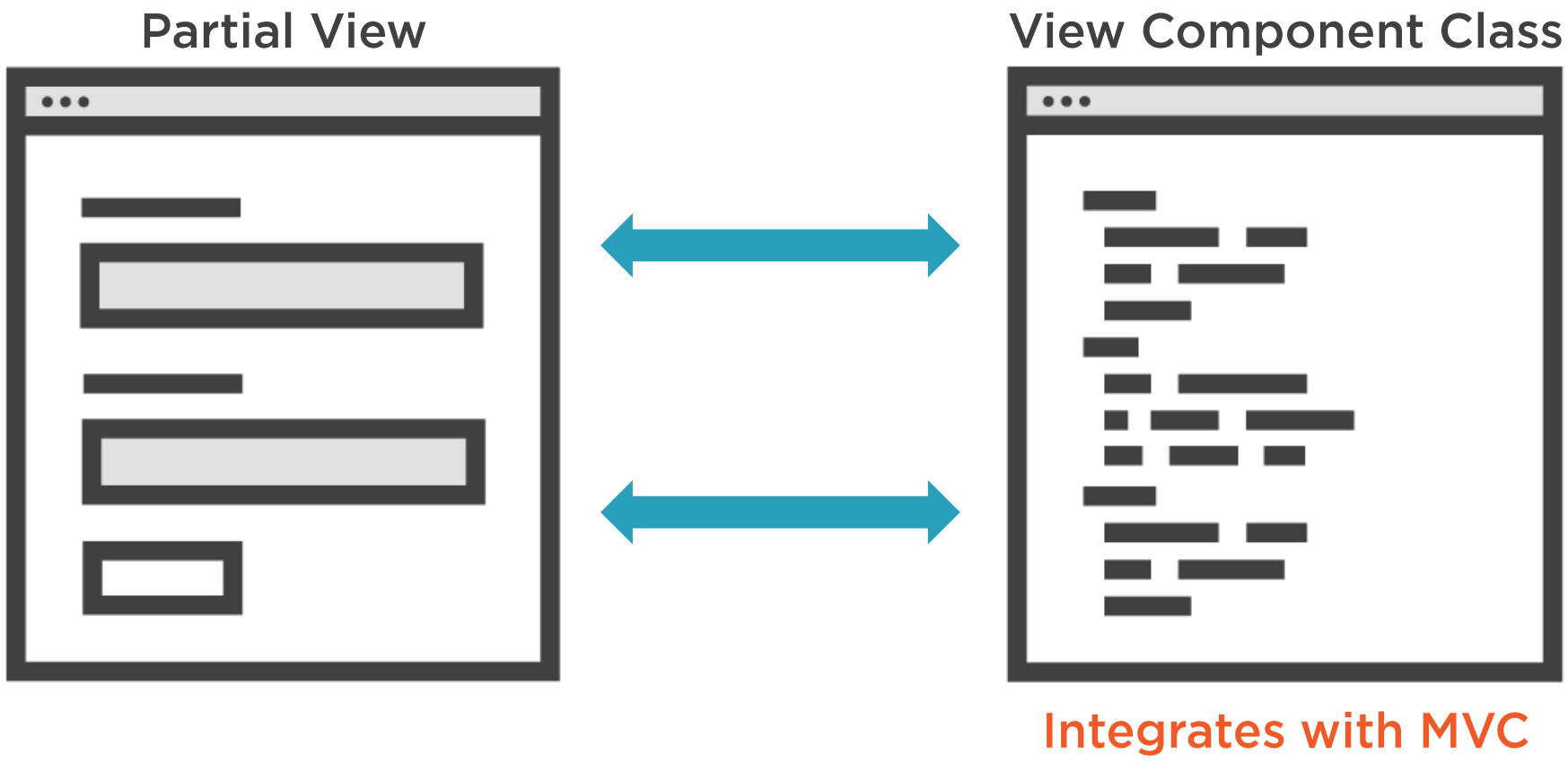
Extracting sections of markup into reusable Partial Views

The Evolution of View Components

The Need for Isolated Components



Empowering Partial Views with Logic



```
public class SurveyViewComponent
{

    public void InvokeAsync()
    {
        // View Component Logic
    }

}
```

- Create a simple View Component using the ViewComponent suffix

- InvokeAsync is the main execution method of a View Component

```
public class SurveyWidget : ViewComponent
{
    public void InvokeAsync()
    {
        var headers = Request.Headers;
        ViewBag.Survey = GetSurveyItems();


        return View();
    }
}
```

- Create a more robust ViewComponent through inheritence
- Inheritence provides access to MVC Framework data objects
- Inheritence also offers helper methods, similar to those in a Controller

```
[View Component]
public class SurveyWidget
{

    public void InvokeAsync()
    {
        // View Component Logic
    }

}
```

- Can also create a simple View Component using the ViewComponent attribute

- This type of View Component does not have access to MVC helpers and context objects

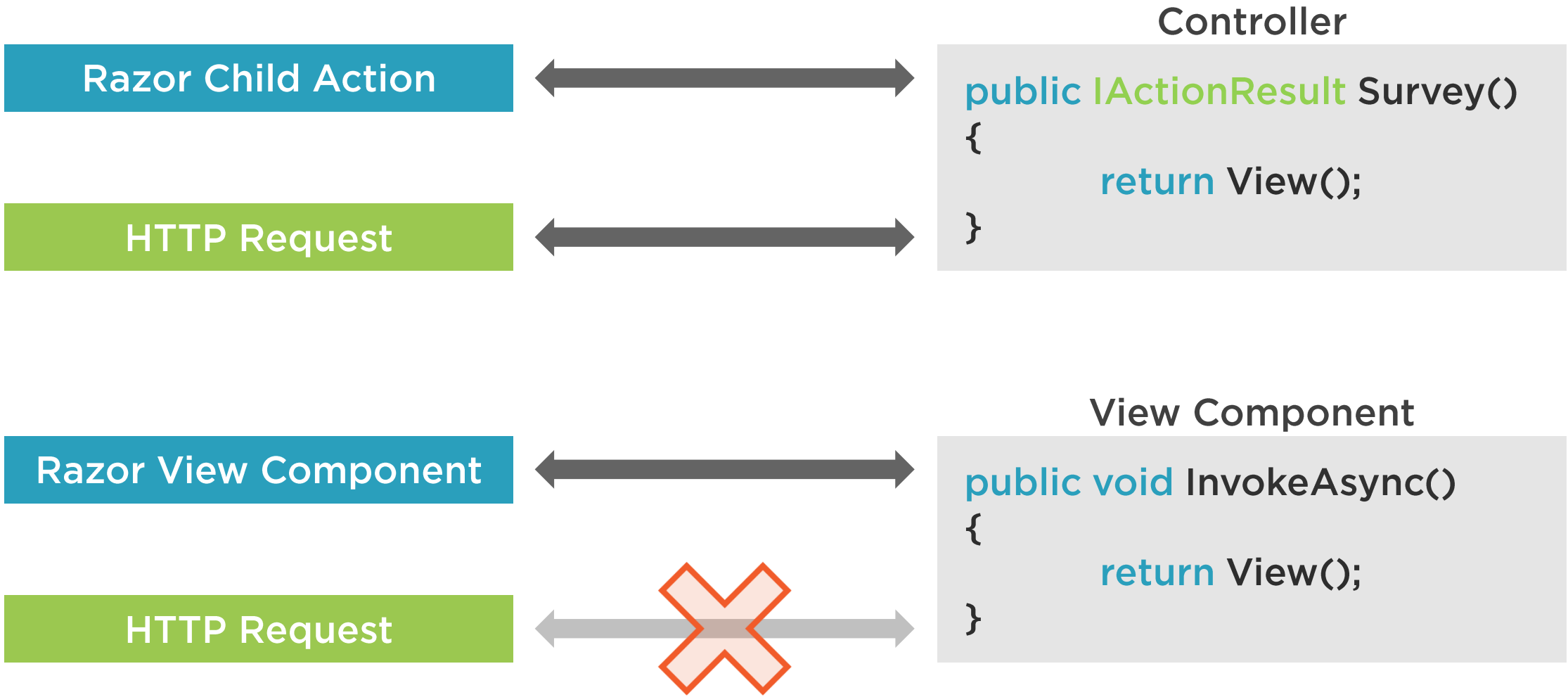
```
<div id="widgets">
    <div class="widget-wrapper">
        @await Component.InvokeAsync("SurveyWidget")
    </div>
</div>
```

Rendering View Components with Razor

View Components can be rendered by Razor from inside of Views

Can be invoked with or without Model parameters

Child Actions and View Components



Template Search Paths for View Components

	Child Actions	View Components
Razor Search Path	Views/Shared/{ViewName}	Views/Shared/Components/{ComponentName}/{ViewName}

Demo

Creating a View Component class through inheritance

Working with View Components and Views

Invoking a View Component

Demo

Exploring the benefits of reusable code

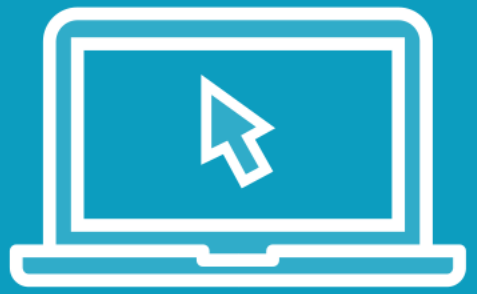
Using parameters to make View Components more flexible

Demo

Exploring isolation in the context of View Components

Understanding how View Components can influence forms and Model Binding

Demo



Adding user interaction and Ajax functionality to a View Component

Invoking a View Component from an Action Method

Summary

View Components allow for partial views that are supported by complex logic

Inheritance can provide View Components with convenience properties and methods

View Components provide reusability and isolation to protect other parts of MVC

HTTP Requests cannot be directly serviced by View Components

Controllers can also execute and return View Components to enable request management