Rebuilding Web Forms Applications in MVC

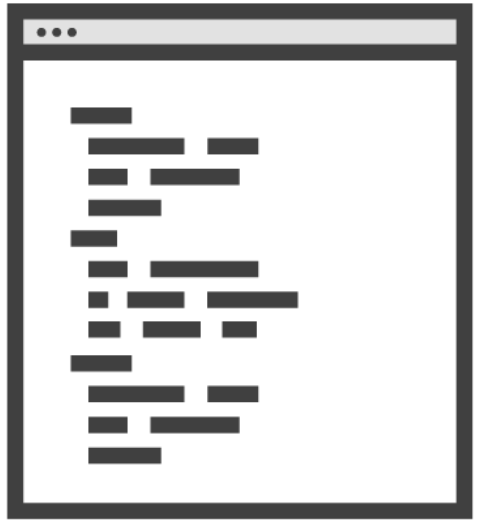
PUTTING WEB FORMS AND MVC IN PERSPECTIVE

Alex Wolf

www.alexwolfthoughts.com

Rebuilding and Improving

Transitioning Between Web Forms and MVC



High Level Concepts

Hey, we have things in common!

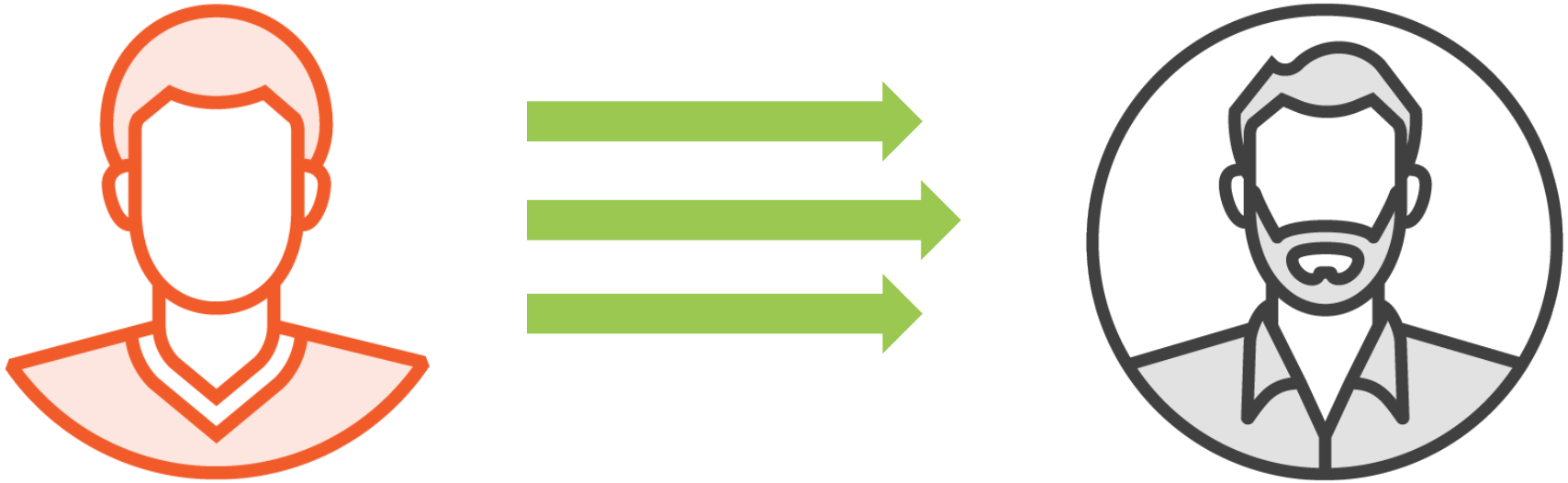
Implementation Details

We're pretty different...

Moving Forward...

Are You in the Right Place?

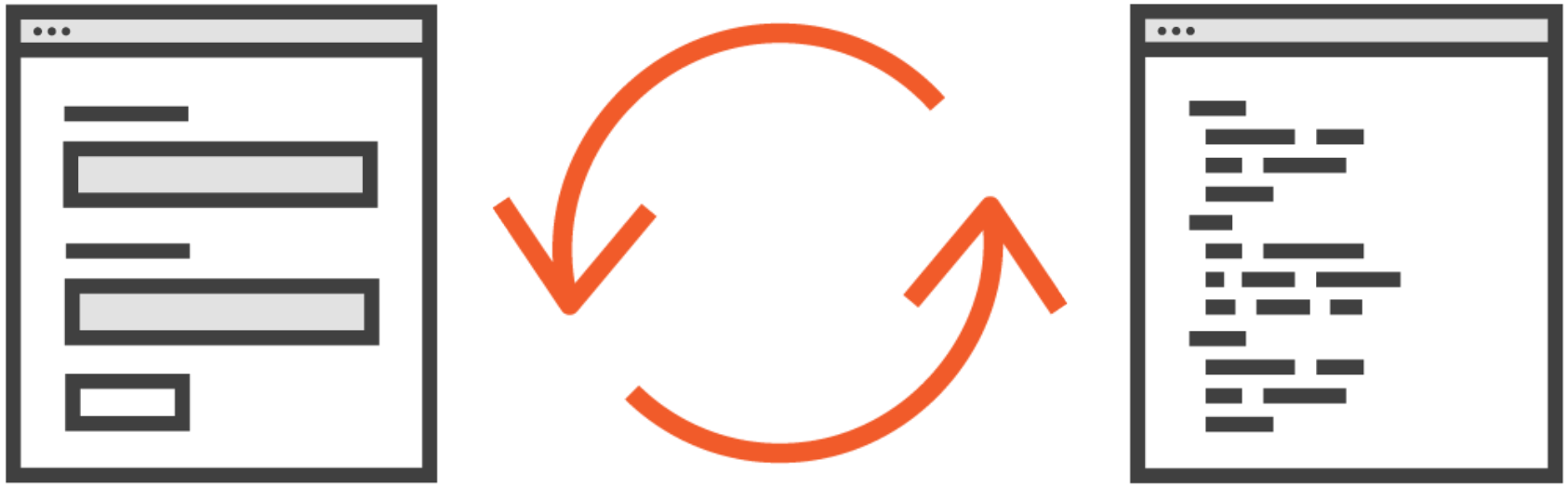
The Transitional Developer



Web Forms Developer

MVC Developer

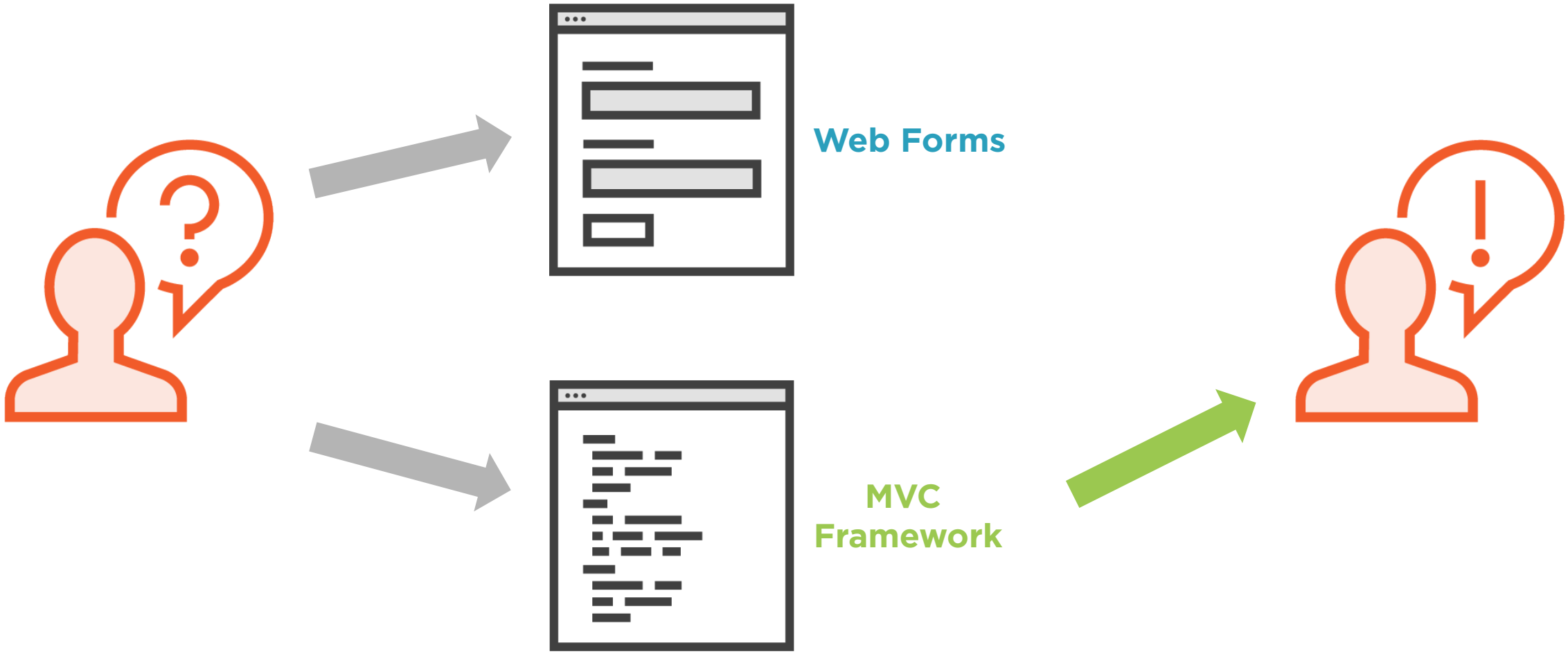
The Reverse Engineer



Web Forms Concepts

MVC Concepts

The Empowered Decision Maker



Managing Expectations

Need to Know

General Web Development

HTML

CSS

HTTP Requests

ASP.NET

General Understanding of the Platform

Some Web Forms or Some MVC

Nice to Know

Other Technologies

Entity Framework

jQuery Basics

Creating Context

Understanding Web Forms

Understanding Web Forms

Built on ASP.NET

- HTTP Modules and Handlers
- Security and User Roles
- Session
- Caching

Markup Generation

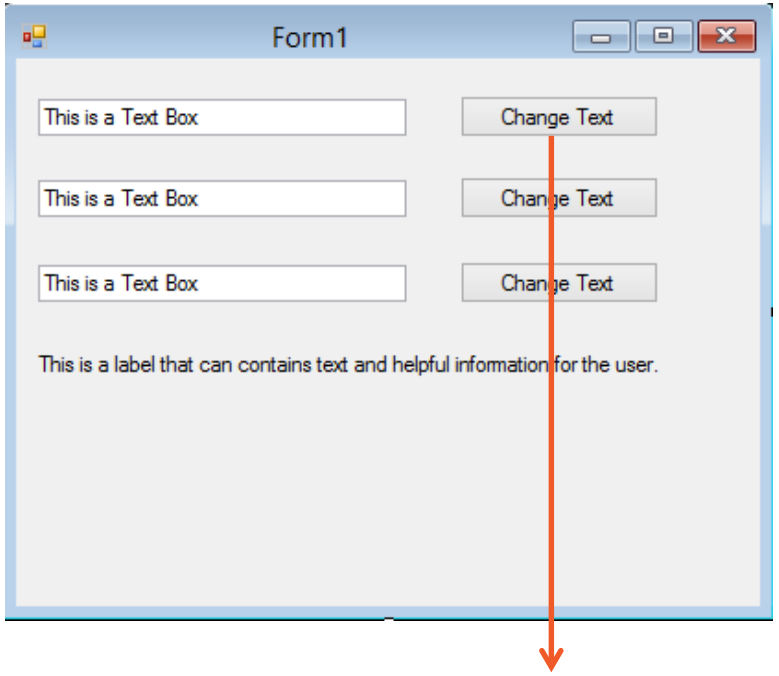
Illusion of State

Code Separation

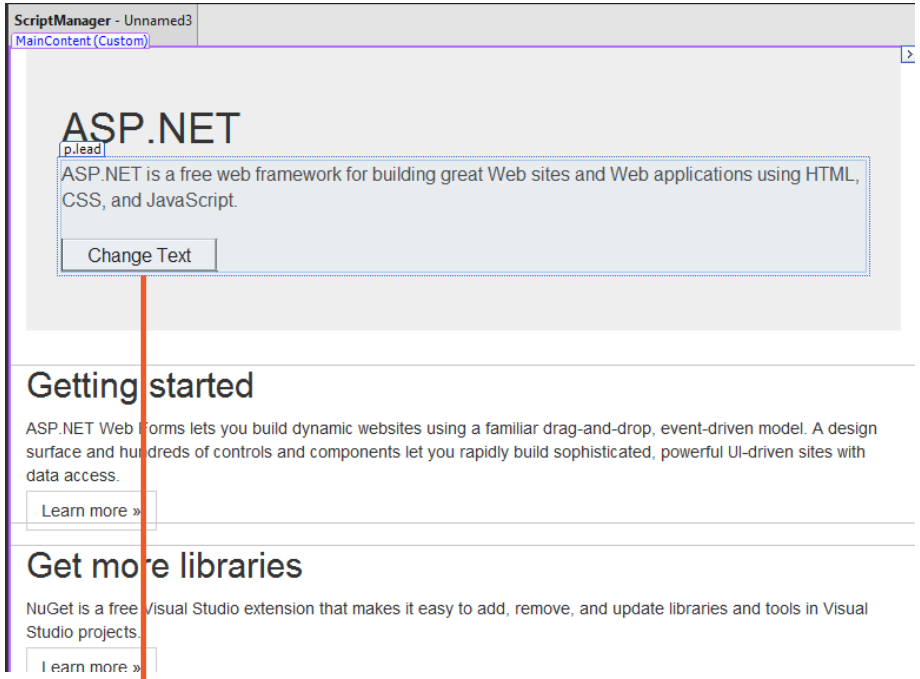
Reusable Components

From Windows Forms to Web Forms

Windows Forms



Web Forms




```
protected void Btn1_Click(object sender, EventArgs args)
{
        textBox1.Text = "This is a Windows Form";
}
```

```
protected void Btn1_Click(object sender, EventArgs args)
{
        label1.Text = "This is a Web Form";
}
```

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="Data
Source=(localdb)\v11.0;Initial Catalog=CypherMVC;Integrated Security=True"
OnSelecting="SqlDataSource1_Selecting" ProviderName="System.Data.SqlClient“
SelectCommand="SELECT Subject, Author, Created FROM Messages"></asp:SqlDataSource>

<asp:ListView ID="ItemList" runat="server"
OnSelectedIndexChanged="ItemList_SelectedIndexChanged"
DataSourceID="SqlDataSource1"></asp:ListView>
```

Web Forms Control Mark Up

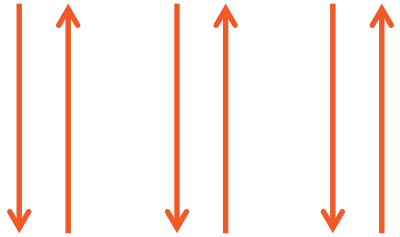
Often troubling, both before and after rendering

From Windows Forms to Web Forms

Web Applications

Remote Client

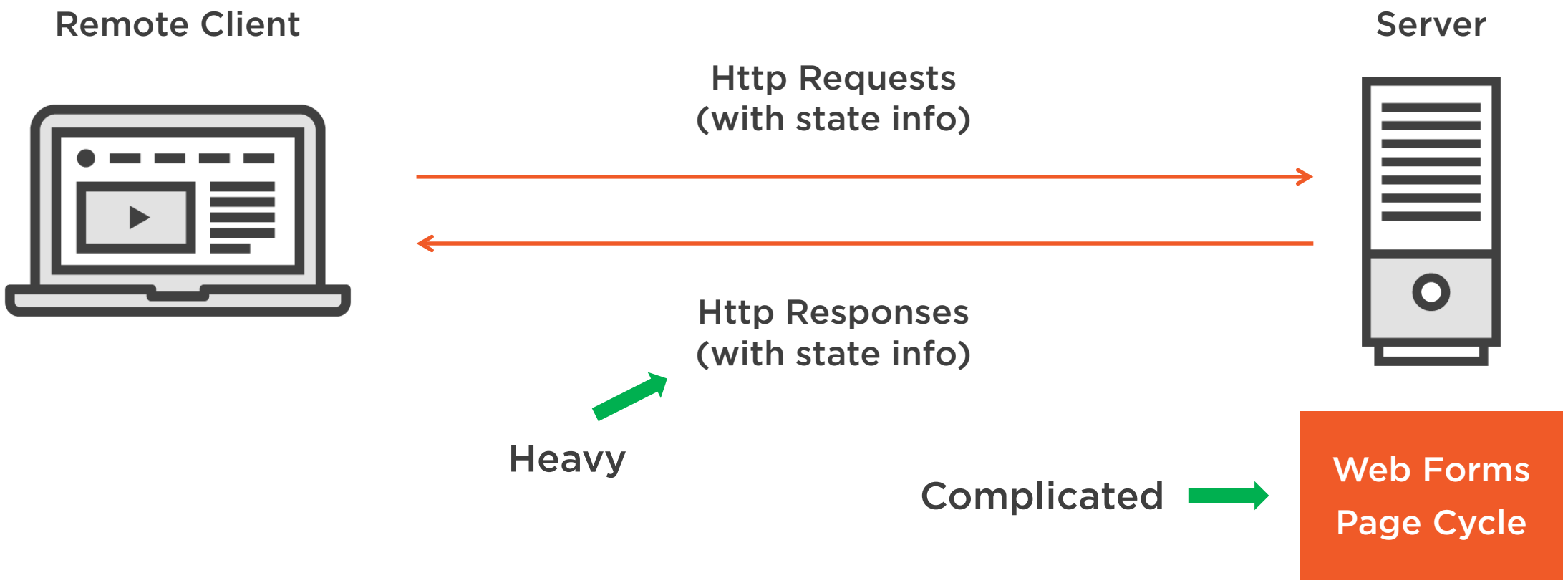
Http Requests



Server

Desktop Client Applications

Enhanced Web Application Requests



Separation of Concerns

Classic Applications
(“spaghetti” code)

```
<div class="row section-1">
(server side logic....)
</div>

<div class="row section-2”>
(server side logic....)
</div>

<div class="row section-3”>
(server side logic....)
</div>
```

Modern Applications
(separated code)

Client Side Code

```
<div class="row section-1"></div>

<div class="row section-2”></div>
```

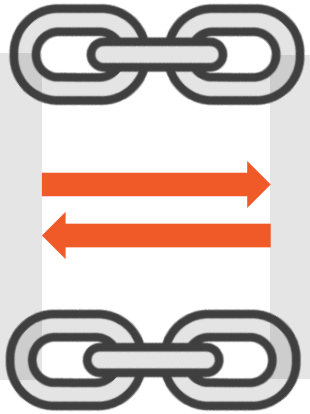
Server Side Code

```
(server side logic....)
```

Tightly Coupled Components

Client Side Code
(Home.aspx)

```
<div class="row section-1"></div>

<div class="row section-2”></div>
```



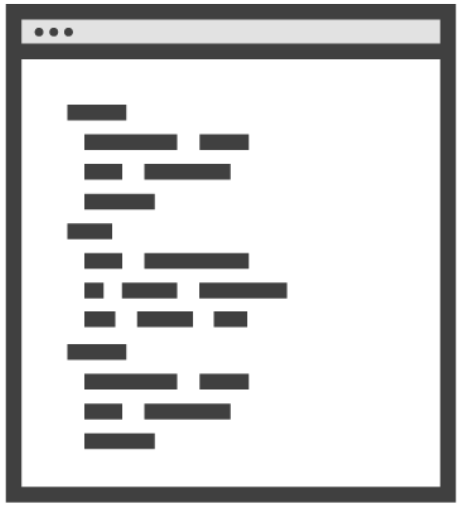
Server Side Code
(Home.aspx.cs)

(server side logic....)

Action Oriented Design

Hey, I need to check out from my mobile app!

Okay, here's a JSON response



I'm on the full website....

You can have full HTML

Other Considerations

Unit Testing

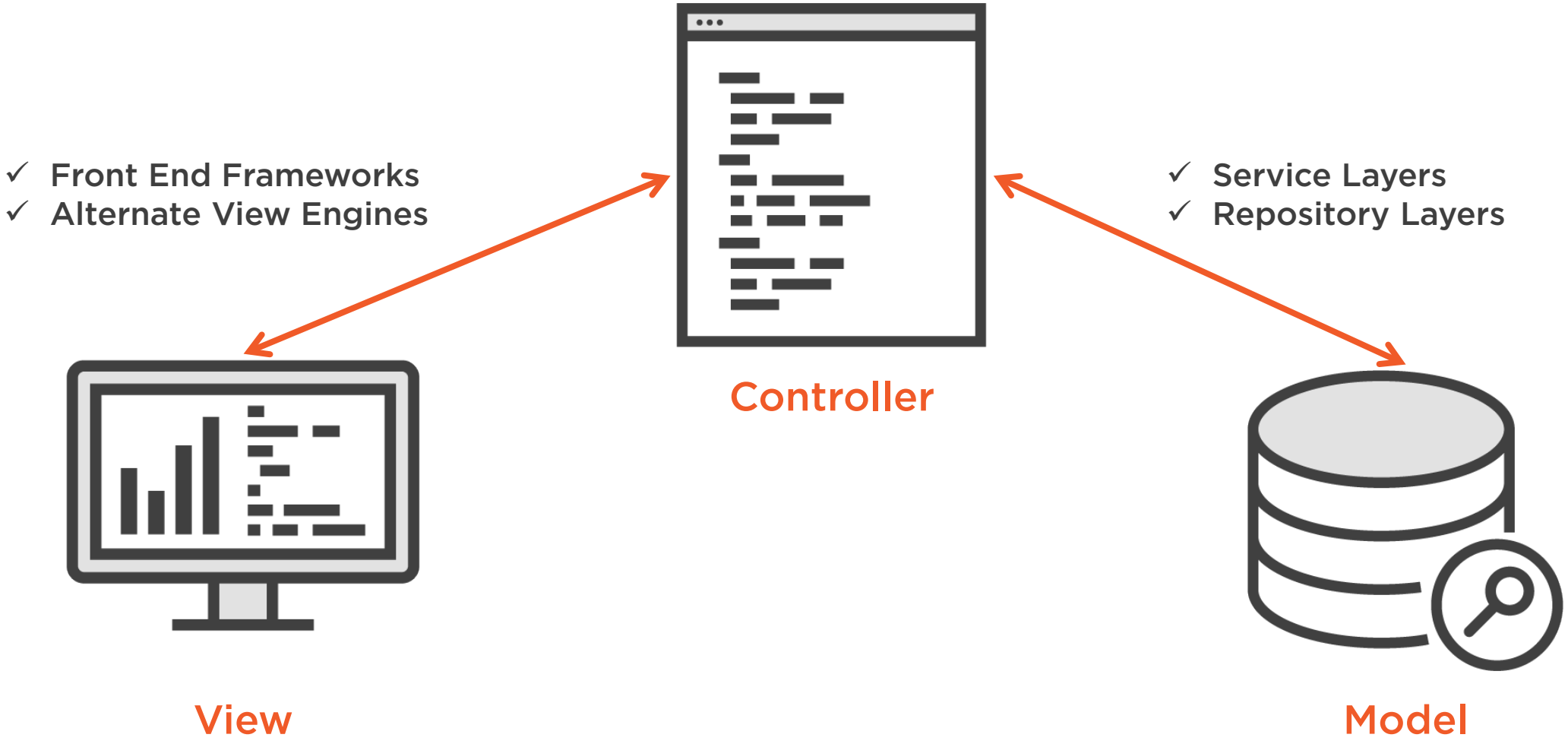
Dependency Injection

Flexible Web Service End Points

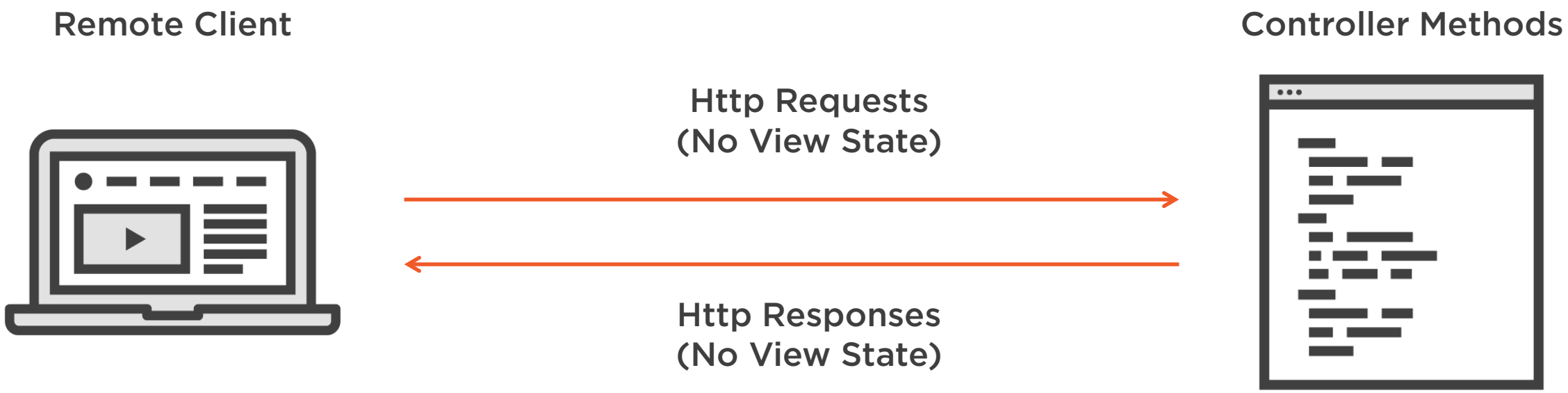
A New Destination

Introducing MVC

The MVC Pattern



Embracing the Nature of HTTP



Flexibility Through Extensibility

Standard Behavior

Use the Default Conventions

Tweaked Behavior

Extend the Existing Components

Fully Custom Behavior

Rewrite and Replace the Provided Classes

Controlling Your Markup

Razor Code		Generated Markup
@Html.TextBoxFor(p => p.Age)	→	<input type="text" name="Age" />
@Html.TextAreaFor(p => p.Comments)	→	<textarea name="Comments"> </textarea>
@Html.ActionLink("Home", "Index", "Go Home")	→	Go Home

```
public class HomeController {

    public ActionResult Index()
    {
        return View();
    }


    public ActionResult GetFeed()
    {
        var feed = db.GetFeed()
        return Json(feed);
    }
}
```

- Responds to home page request with HTML

- Same controller responds with JSON feed

Embracing Modern Design Patterns

- Unit Testing
- Dependency Injection
- Flexible Web Service End Points

MVC and ASP.NET

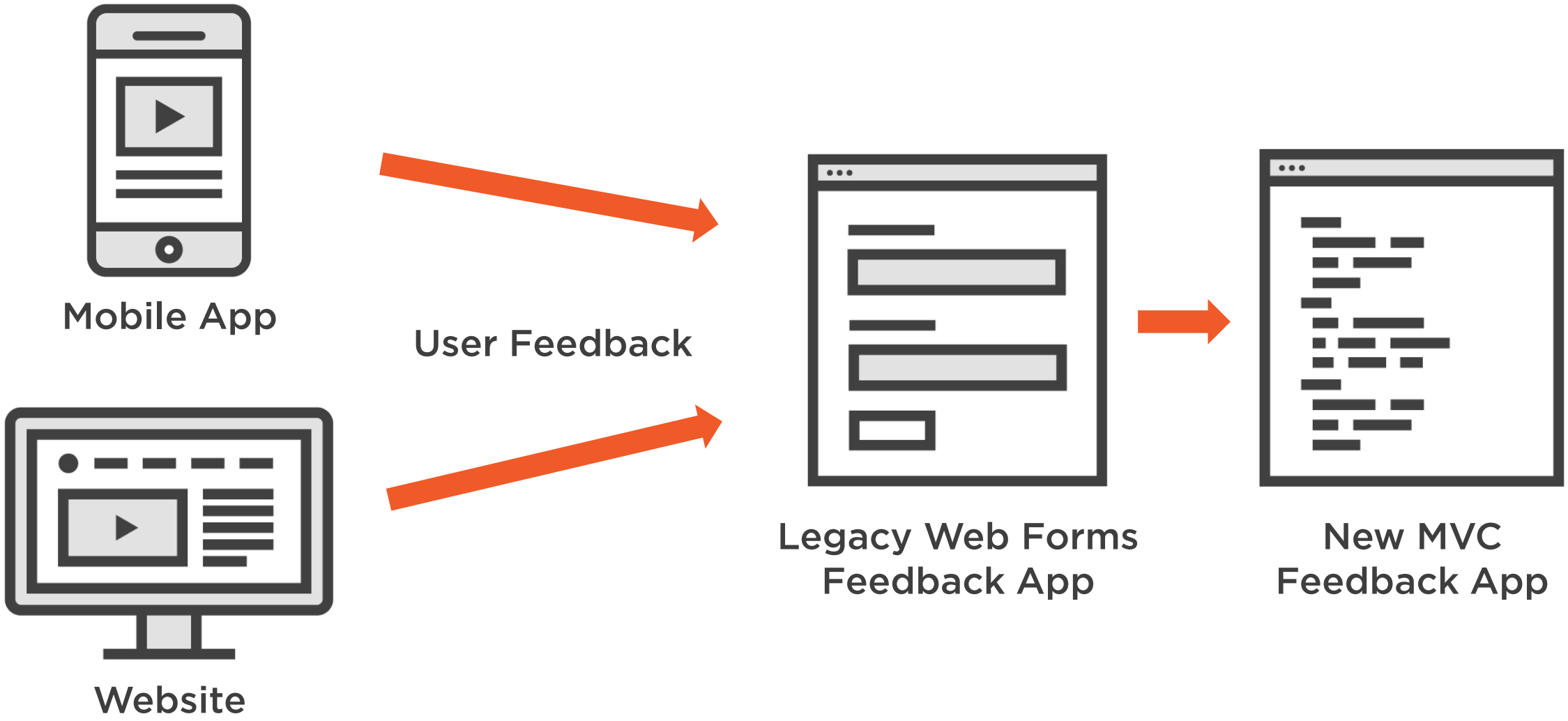
The Agenda

To-Do List

- Request Management
- Designing with Layouts and Views
- Forms and Model Binding
- Validating Form Data
- User Controls and Partial Views
- Handling Ajax and Service Calls
- Displaying and Accessing Data

Building for the Future

Responding to Feedback



Summary

Web Forms Offers Interesting but Challenging Features

- View State
- HTML and HTTP Abstractions
- Split Page Model
- Page and Form Validation

Both Web Forms and MVC are Built on ASP.NET

MVC is Powerful and Extensible

Built for Modern Design Patterns

MVC Embraces Nature of Web

Request Management

Alex Wolf

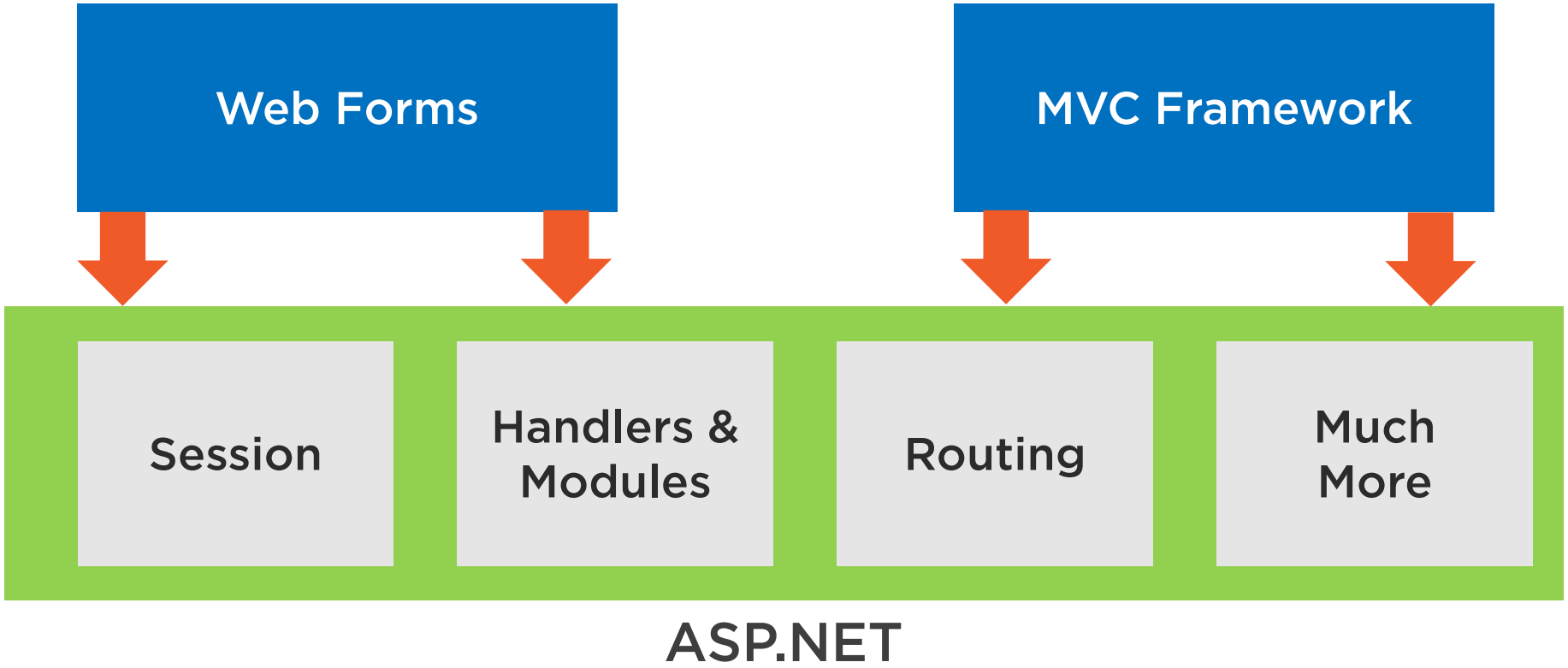
www.alexwolfthoughts.com

To-Do List

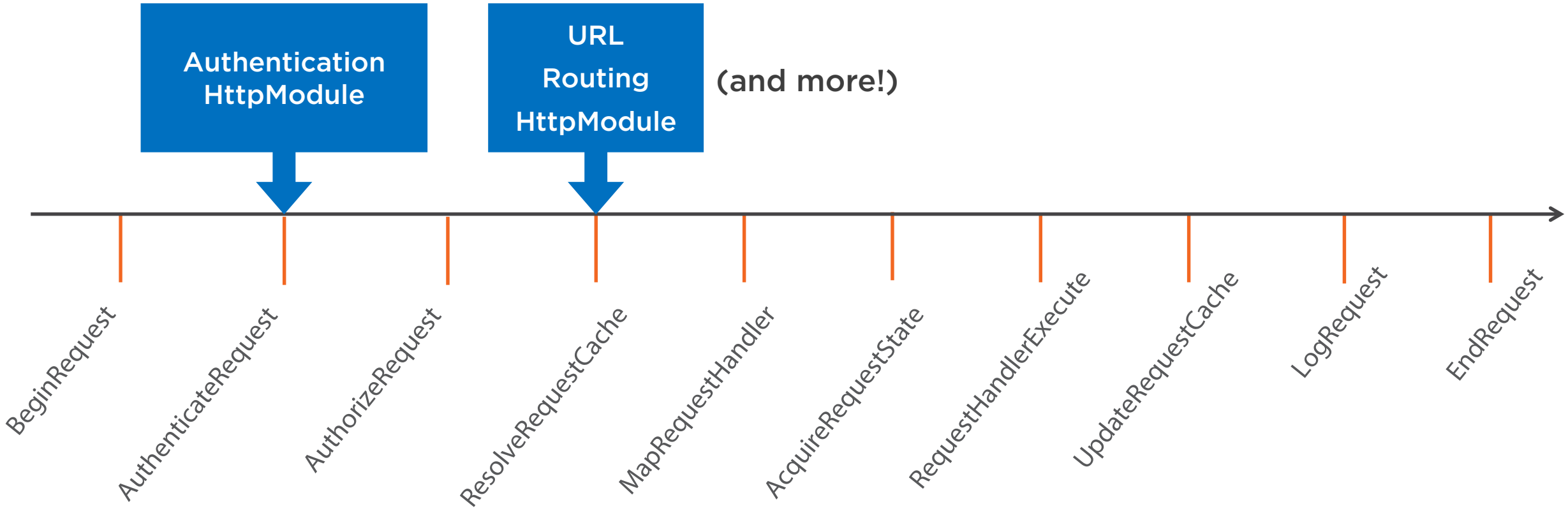
- The Shared ASP.NET Platform
- Demo – Exploring the Application Life Cycle
- Handling Requests with Web Forms
- Demo – Touring the Legacy Application
- Handling Requests with MVC
- Understanding Routing
- Demo – Routing and Requests in MVC
- Demo – Building the Controller Structure

The Shared ASP.NET Platform

A Common Foundation



The ASP.NET Application Life Cycle



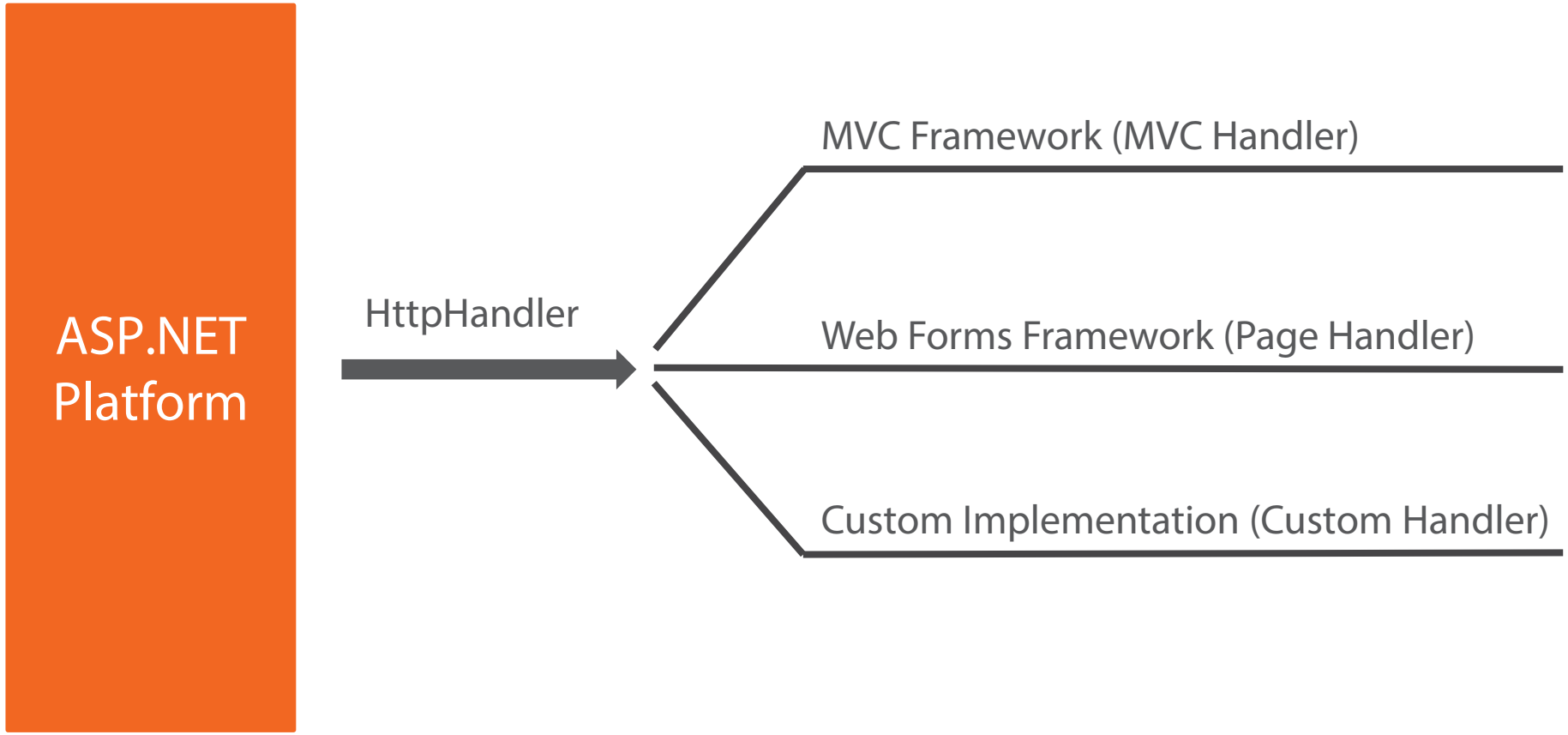
* Event names shown represent both pre and post events

```
public interface IHttpModule {
        void Dispose();
        void Init(HttpApplication application);
}
```

Implementing an HttpModule

Not an overly common task, but just in case!

One Platform, Many Implementations

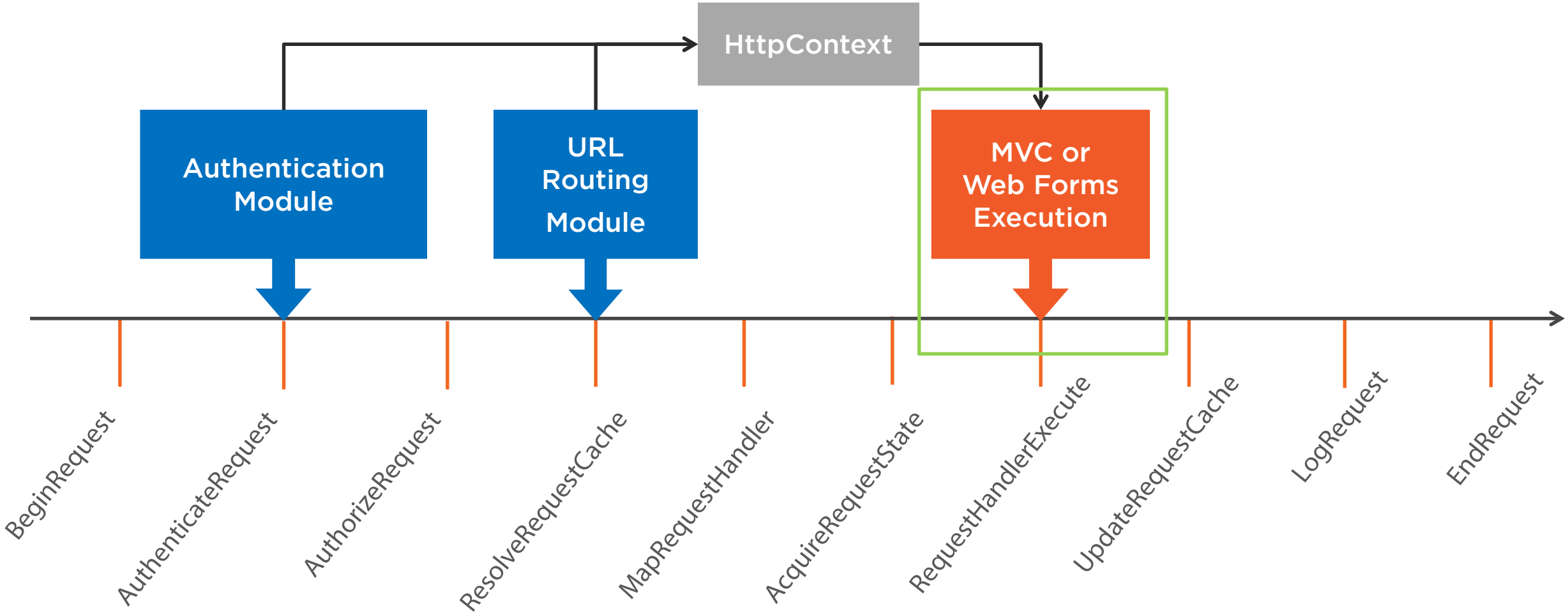


```
public interface IHttpHandler {
    bool IsReusable();
    void ProcessRequest(HttpContext context);
}
```

Implementing an HttpHandler

Remember, ProcessRequest ultimately generates the response

HttpHandlers and the Application Life Cycle



* Event names shown represent both pre and post events

Summarizing Modules and Handlers

HttpModules

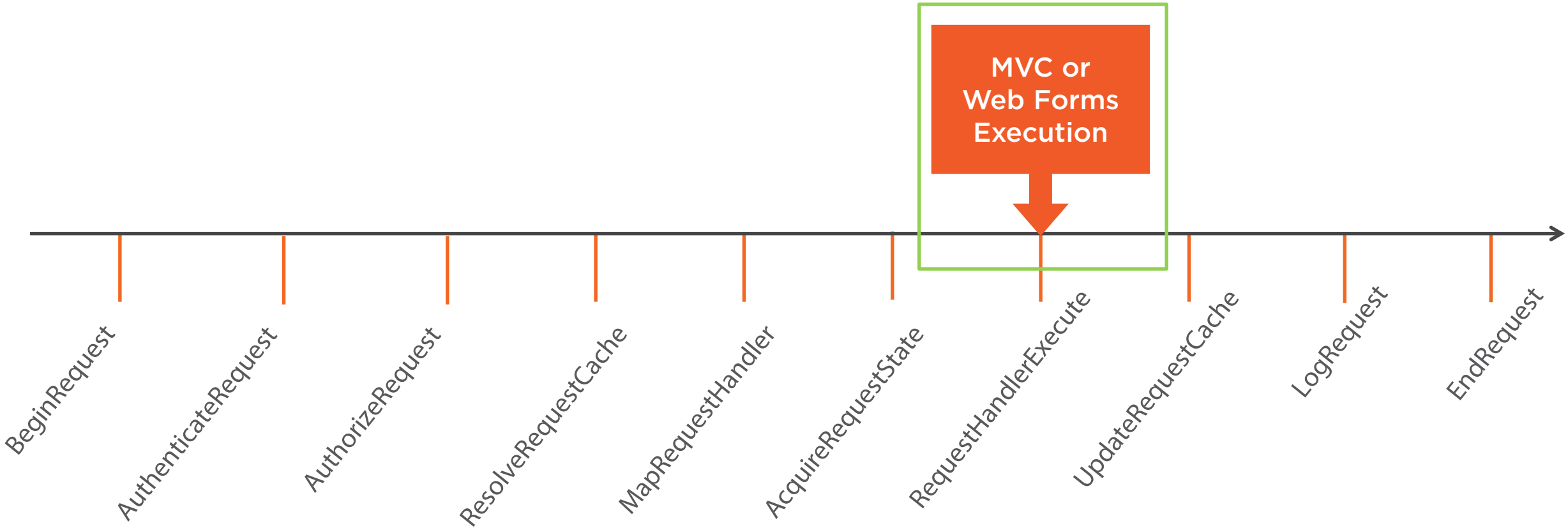
Hook into Application Level Events to provide supporting features

HttpHandlers

Selected and executed by ASP.NET to generate a response for a request

Handling Requests in Web Forms

HttpHandlers and the Application Life Cycle



* Event names shown represent both pre and post events

The Web Forms Page Life Cycle

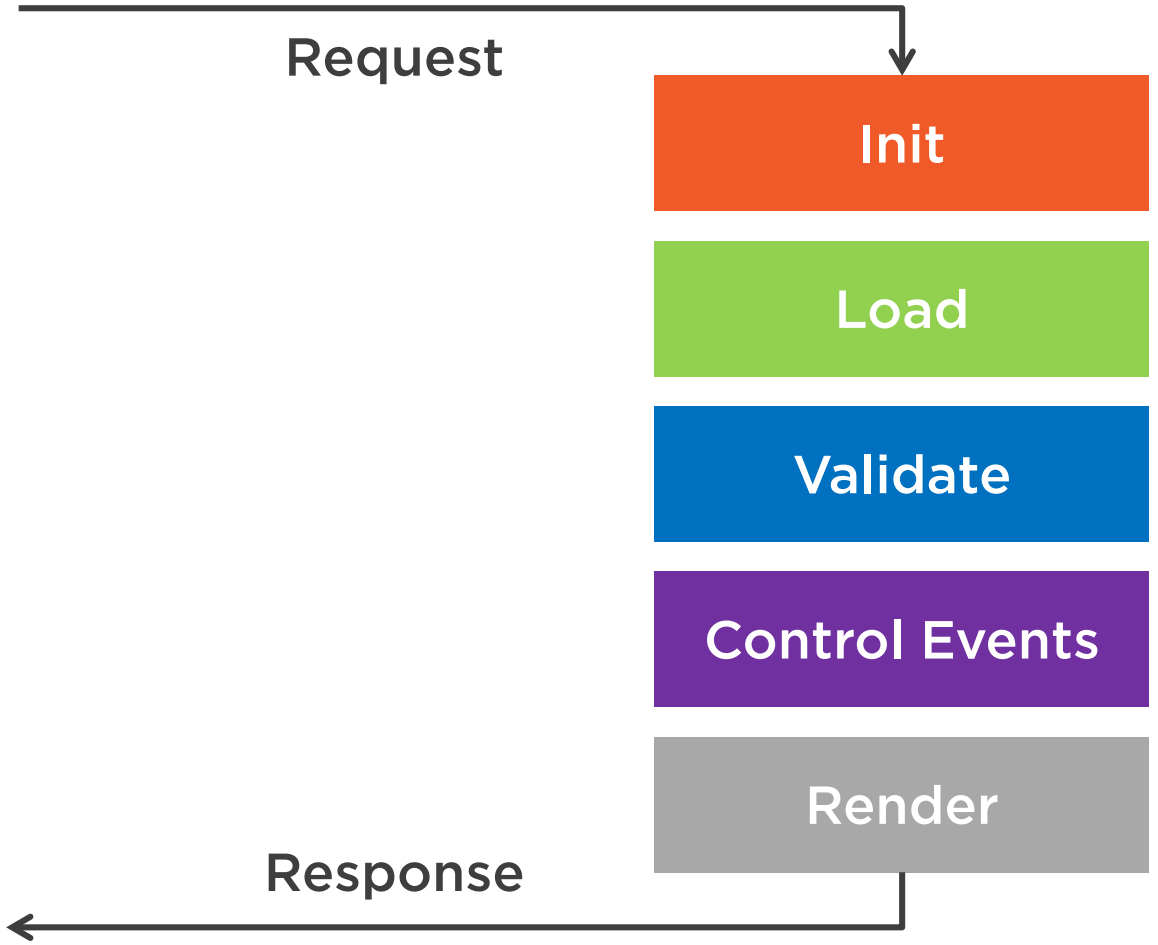
A series of page processing events

Abstraction over Http Requests

Ties together various features such as View State and Validation

Many events, but only a handful used day to day

The (abridged) Web Forms Page Life Cycle



Initialization Code for Controls

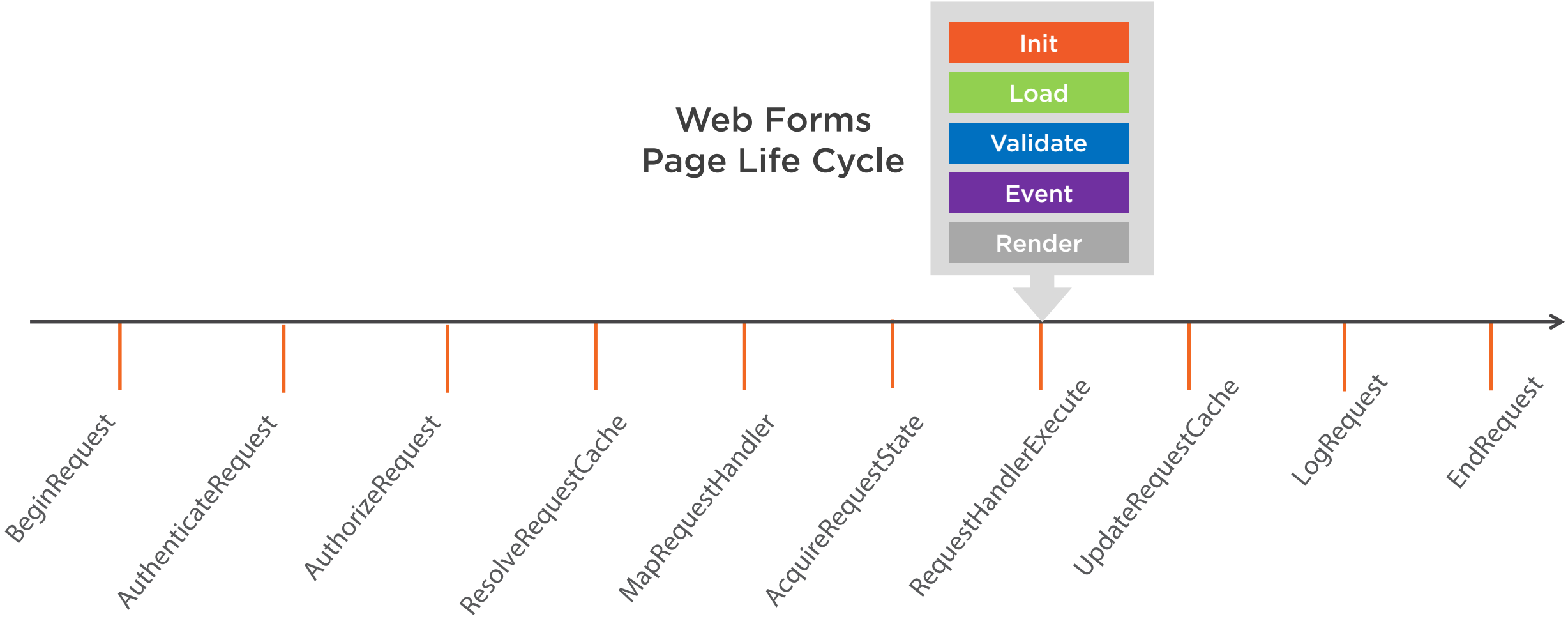
View State, Controls, Form Values Available
Custom logic and code relate to Controls

Page Validation Controls are Run

Event Handlers are Run
(Example: Button Click)

Last chance to make
changes to HTML

Web Forms and the Application Life Cycle

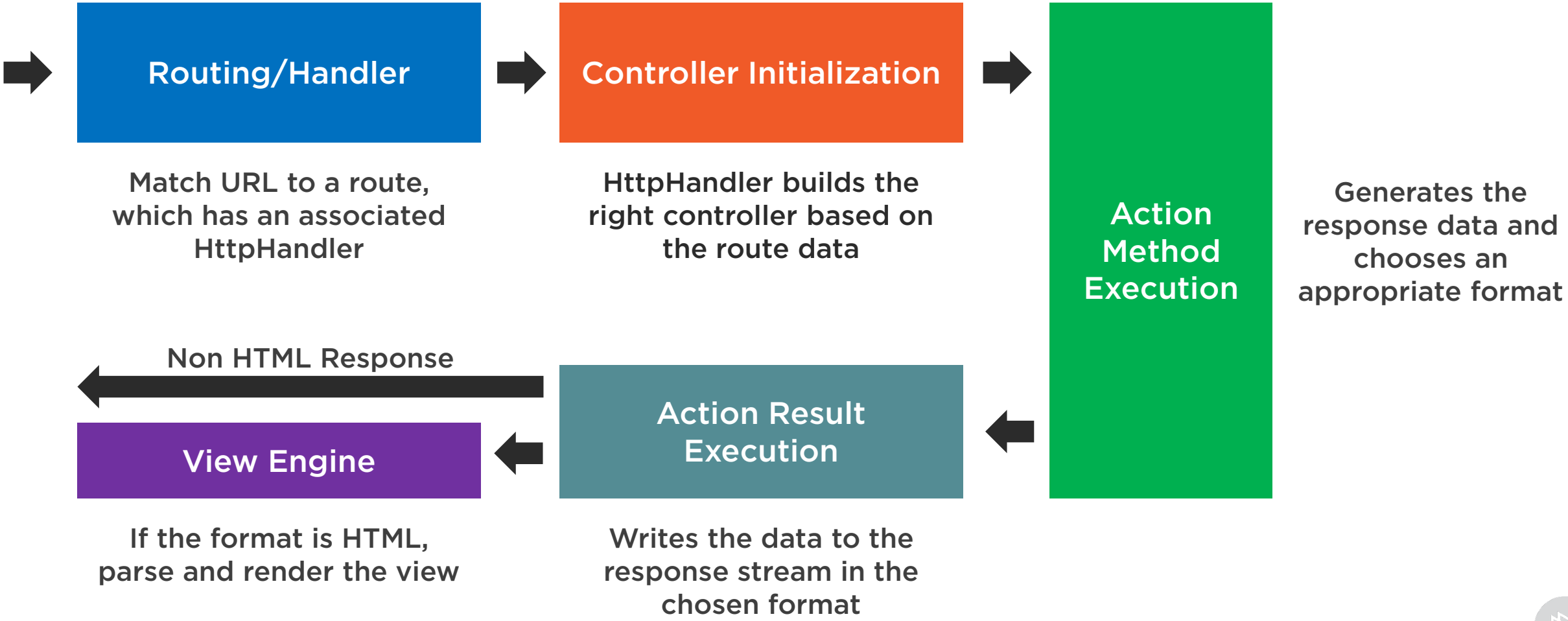


* Event names shown represent both pre and post events

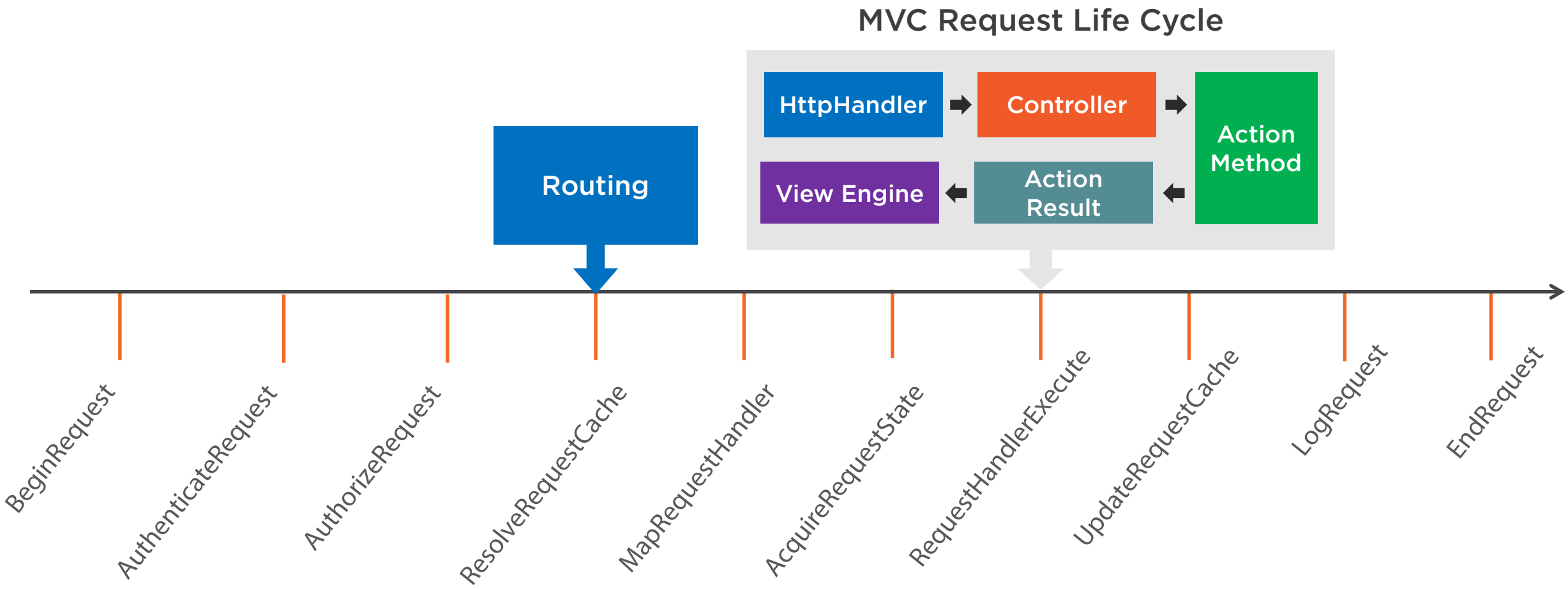
The (abridged) Web Forms Page Life Cycle

Handling Requests in MVC

The MVC Request Life Cycle



MVC and the Application Life Cycle



* Event names shown represent both pre and post events

```
public class MessagesController : IController{
        public void Execute(){  }
}
public class MessagesController : Controller {
        //Action Methods
}
```

Implementing a Controller

Use the interface yourself, or just inherit!

Handling Requests with Controllers

MySite.com/Task/Delete

MySite.com/Task/Create

```
public class TaskController {

  public ActionResult Delete()
  {
        return View();
  }

  public ActionResult Create()
  {
        return View();
  }

}
```

HomeController

MessagesController

WidgetController

Mapping Requests to Action Methods

Site.com/Task/ViewAll

Controller

Action Invoker

Decides which action should handle a request

```
public class TaskController {
    public ActionResult ViewAll
    {
        return View();
    }
    [HttpPost] //Post Requests Only
    public ActionResult Edit(Task task)
    {
        return View();
    }
}
```

Understanding ActionResult Types

ViewResult

Parses Razor Syntax and returns HTML

JsonResult

Formats response as JSON data

ContentResult

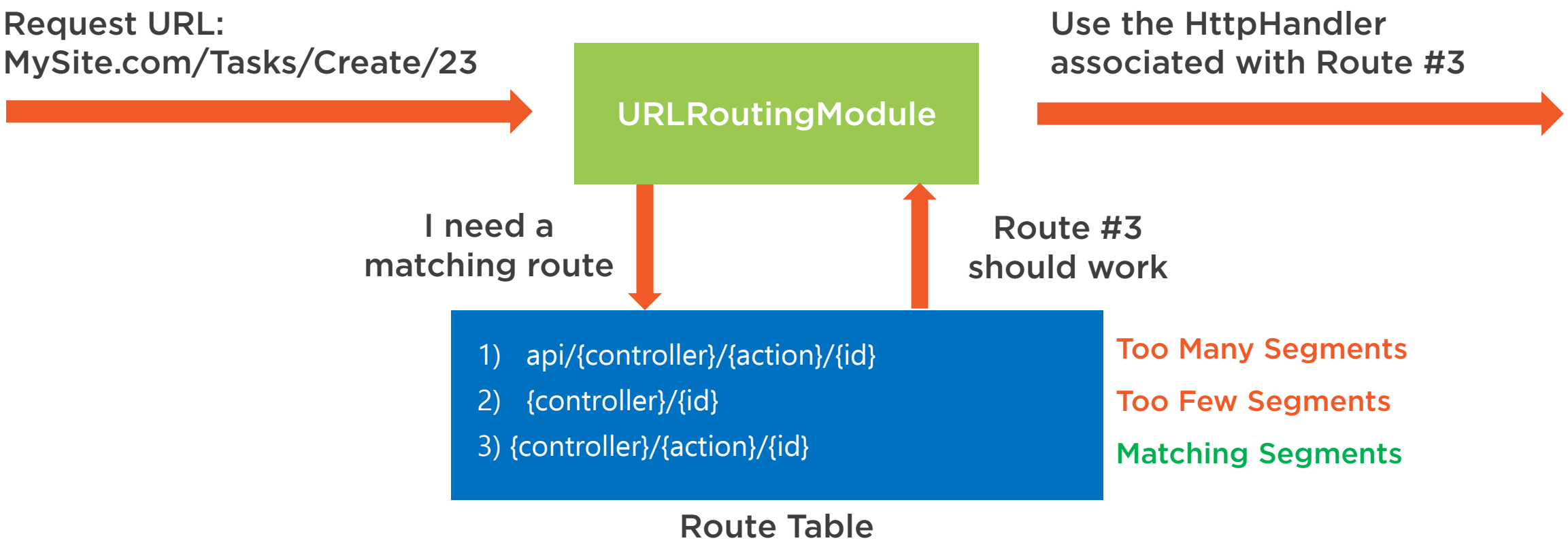
Writes out a string result

ActionResult

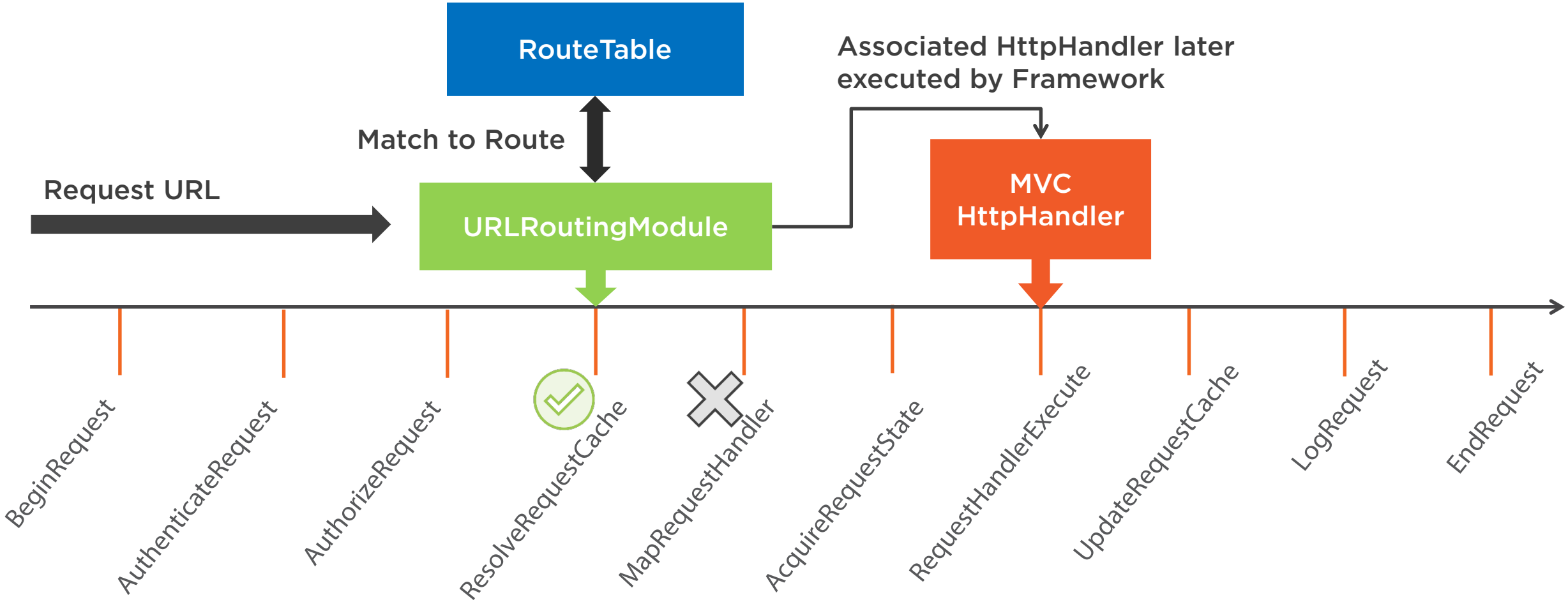
What About Routing?

Routing Requests

Understanding Route and HttpHandler Selection



The URLRoutingModule at Work



* Event names shown represent both pre and post events

```
routes.MapRoute(
  name: "Default",

  url: "{controller}/{action}/{id}",

  defaults: new {
        controller = "Home",
        action = "Index",
        id = UrlParameter.Optional
        }
);
```

- Name of the route for easy referencing
- The URL segment pattern to match
- Default values can be provided as fall backs for missing segments
- Constraints that apply rules for whether a URL segment value is valid

Understanding Route Pattern Matching

URL: Mysite.com/Task/Edit/23

Matched Route: {controller}/{action}/{id}

```
public class TaskController {
  public ActionResult (int Edit ) id
  {
      return View();
  }
}
```

The Route Ahead

Summary

Web Forms maps requests to physical pages in file directories

MVC dynamically handles requests by mapping them to action methods

Action Methods can return whatever data type and format is appropriate

MVC relies heavily on routing to process request information

ASP.NET provides considerable infrastructure for both frameworks

Designing with Layouts and Views

Alex Wolf

www.alexwolfthoughts.com

To-Do List

- Reviewing the Design Features of Web Forms
- Introducing Layouts and Views in MVC
- Demo: Building the Layout in MVC
- Demo: Adding Styles and Scripts in MVC
- Reviewing the Web Forms Server Controls
- Introducing the Razor View Engine
- Demo: Building the Navigation with Razor

Design Features of Web Forms

Understanding Master Pages

Cypher Marketing (MVC)

Messages

View Messages

Tasks

Create Task

View Tasks

Master Page

Welcome to the Feedback Application!

Here you can manage the messages and tasks for our customers.

ASPX Page

Latest Messages

From	Subject	Sent	Action
mark@sample.com	Concerned Customer	1/17/2016 11:01:39 AM	View
joe@sample.com	Question about shipping	1/17/2016 11:01:39 AM	View
sam@sample.com	Mobile app troubles	1/17/2016 11:01:39 AM	View
alex@sample.com	Poor service	1/17/2016 11:01:39 AM	View
john@sample.com	Like the new site!	1/17/2016 11:01:39 AM	View

See all

Admin Tasks

Title	Description	Due Date	Action
Follow up with marketing about materials	A vendor has not received shipments for an order they placed through the online form.	1/24/2016 11:01:39 AM	View
Vendor issues	A vendor has not received their order, please address this and check that the order was placed.	1/24/2016 11:01:39 AM	View
Poor customer relations	Multiple customer complaints have come through about our phone service - please contact customer relations to verify that procedures are being followed.	1/24/2016 11:01:39 AM	View

See all

Rep of the Month

You can only vote once!

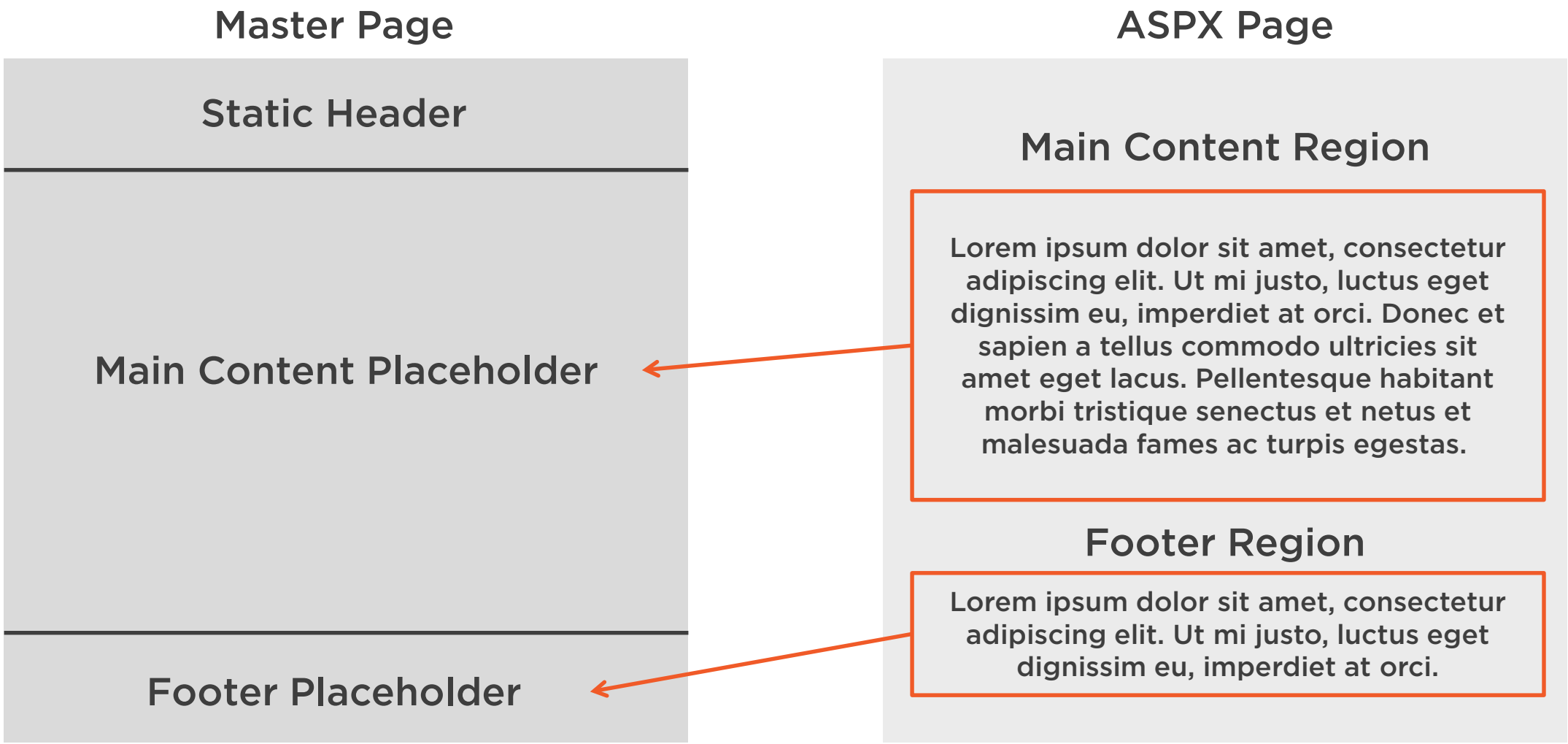
User	Add Vote
Jane	Vote Up
Sarah	Vote Up
Joe	Vote Up
Bob	Vote Up

Continuous Improvement

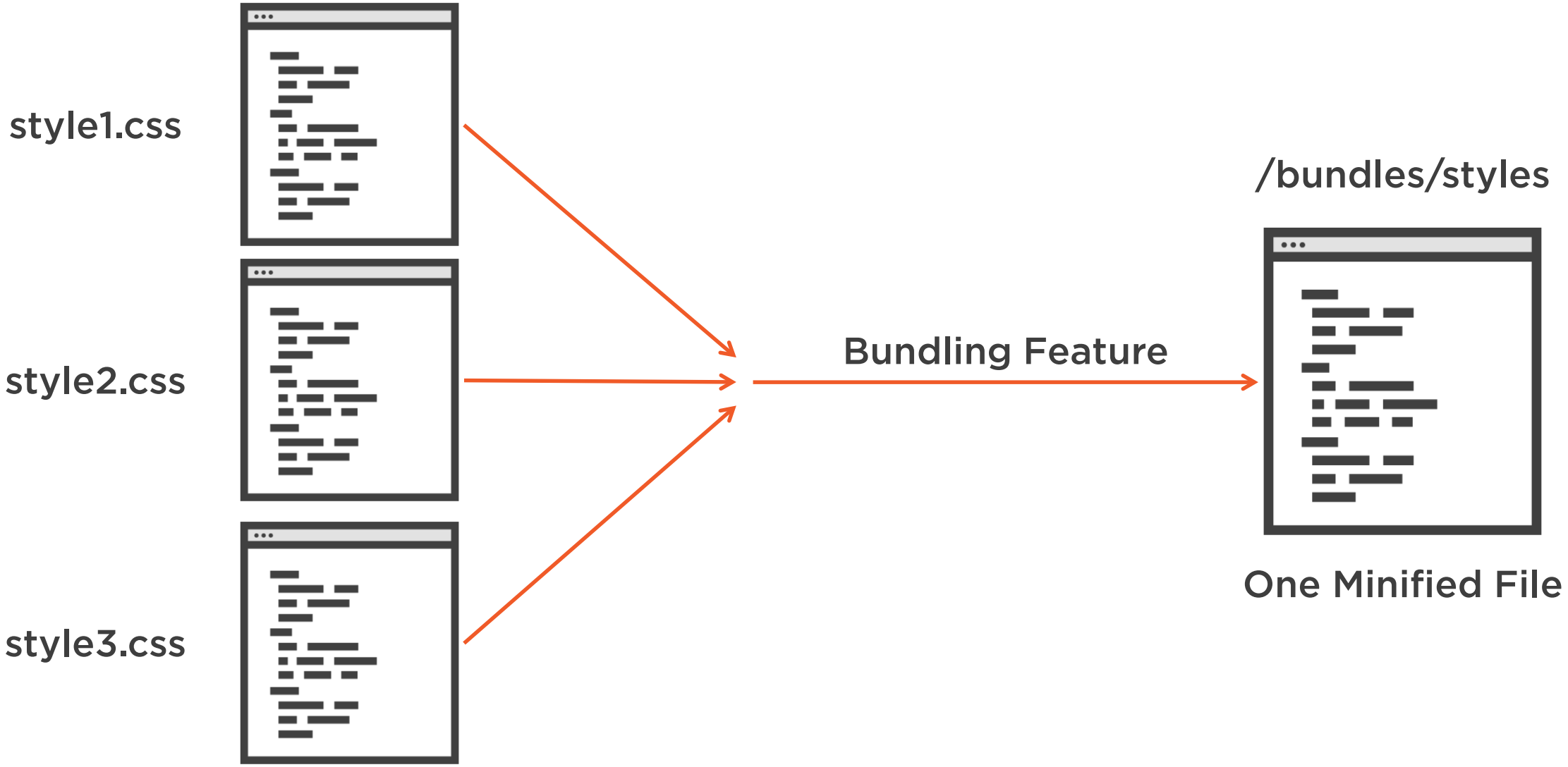
Have a feature request or suggestion for how we can improve this app? Let us know! Send us a suggestion.

Submit

Master Page Content Regions



Scripts, Styles and Bundling



Organizing and Reusing with User Controls

Welcome to the Feedback Application!

Here you can manage the messages and tasks for our customers.

Latest Messages

From	Subject	Sent	Action
mark@sample.com	Concerned Customer	1/17/2016 11:01:39 AM	View
joe@sample.com	Question about shipping	1/17/2016 11:01:39 AM	View
sam@sample.com	Mobile app troubles	1/17/2016 11:01:39 AM	View
alex@sample.com	Poor service	1/17/2016 11:01:39 AM	View
john@sample.com	Like the new site!	1/17/2016 11:01:39 AM	View

See all

Admin Tasks

Title	Description	Due Date	Action
Follow up with marketing about materials	A vendor has not received shipments for an order they placed through the online form.	1/24/2016 11:01:39 AM	View
Vendor issues	A vendor has not received their order, please address this and check that the order was placed.	1/24/2016 11:01:39 AM	View
Poor customer relations	Multiple customer complaints have come through about our phone service - please contact customer relations to verify that procedures are being followed.	1/24/2016 11:01:39 AM	View

See all

Rep of the Month

You can only vote once!

User	Add Vote
Jane	Vote Up
Sarah	Vote Up
Joe	Vote Up
Bob	Vote Up

Continuous Improvement

Have a feature request or suggestion for how we can improve this app? Let us know! Send us a suggestion.

Submit

Survey Widget.ascx

Feedback Widget.ascx

Revisiting the Legacy Application Design

Layouts, Views, and Partial Views

Understanding Layouts

Cypher Marketing (MVC)

Messages

View Messages

Tasks

Create Task

View Tasks

Layout

View

Welcome to the Feedback Application!

Here you can manage the messages and tasks for our customers.

Latest Messages

From	Subject	Sent	Action
mark@sample.com	Concerned Customer	1/17/2016 11:01:39 AM	View
joe@sample.com	Question about shipping	1/17/2016 11:01:39 AM	View
sam@sample.com	Mobile app troubles	1/17/2016 11:01:39 AM	View
alex@sample.com	Poor service	1/17/2016 11:01:39 AM	View
john@sample.com	Like the new site!	1/17/2016 11:01:39 AM	View

See all

Admin Tasks

Title	Description	Due Date	Action
Follow up with marketing about materials	A vendor has not received shipments for an order they placed through the online form.	1/24/2016 11:01:39 AM	View
Vendor issues	A vendor has not received their order, please address this and check that the order was placed.	1/24/2016 11:01:39 AM	View
Poor customer relations	Multiple customer complaints have come through about our phone service - please contact customer relations to verify that procedures are being followed.	1/24/2016 11:01:39 AM	View

See all

Rep of the Month

You can only vote once!

User	Add Vote
Jane	Vote Up
Sarah	Vote Up
Joe	Vote Up
Bob	Vote Up

Continuous Improvement

Have a feature request or suggestion for how we can improve this app? Let us know! Send us a suggestion.

Submit

```
<div id="wrapper">
    <div id="page-wrapper">
        @RenderBody()
    </div>

@RenderSection("Secondary", false)

@if (!IsSectionDefined("Footer"))
{
<footer>This is the footer.</footer>
}
</div>
```

- Pulls the main view into the Layout
- Optional content section that can be overridden from view
- Optional footer content section that provides fallback content

Organizing and Reusing with User Controls

Welcome to the Feedback Application!

Here you can manage the messages and tasks for our customers.

Latest Messages

From	Subject	Sent	Action
mark@sample.com	Concerned Customer	1/17/2016 11:01:39 AM	View
joe@sample.com	Question about shipping	1/17/2016 11:01:39 AM	View
sam@sample.com	Mobile app troubles	1/17/2016 11:01:39 AM	View
alex@sample.com	Poor service	1/17/2016 11:01:39 AM	View
john@sample.com	Like the new site!	1/17/2016 11:01:39 AM	View

See all

Admin Tasks

Title	Description	Due Date	Action
Follow up with marketing about materials	A vendor has not received shipments for an order they placed through the online form.	1/24/2016 11:01:39 AM	View
Vendor issues	A vendor has not received their order, please address this and check that the order was placed.	1/24/2016 11:01:39 AM	View
Poor customer relations	Multiple customer complaints have come through about our phone service - please contact customer relations to verify that procedures are being followed.	1/24/2016 11:01:39 AM	View

See all

Rep of the Month

You can only vote once!

User	Add Vote
Jane	Vote Up
Sarah	Vote Up
Joe	Vote Up
Bob	Vote Up

Continuous Improvement

Have a feature request or suggestion for how we can improve this app? Let us know! Send us a suggestion.

Submit

Partial View

Partial View

```
<div id="wrapper">

    <div id="widgets">

    @Html.Partial("SurveyWidget")

    @Html.Action("FeedbackWidget",
    "WidgetController")

    </div>

</div>
```

- Inserts a partial view directly
- Specifies an Action Method on a Controller to execute and inserts the result

Comparing Sections and Partial Views

Sections

Content placeholders in the Layout file that enforce document structure

Partial Views

Individual calls inside of Layouts or Views to inject the contents of an additional View

Comparing Views and Partial Views

Views

.cshtml file

==

Technically Equivalent

Partial Views

.cshtml file

Generating Views and Partial Views

Regular Views

From a Controller

```
return View()
```

From another View

Not Applicable

Partial Views

From a Controller

```
return PartialView()
```

From another View

```
@Html.Action
@Html.Partial
```

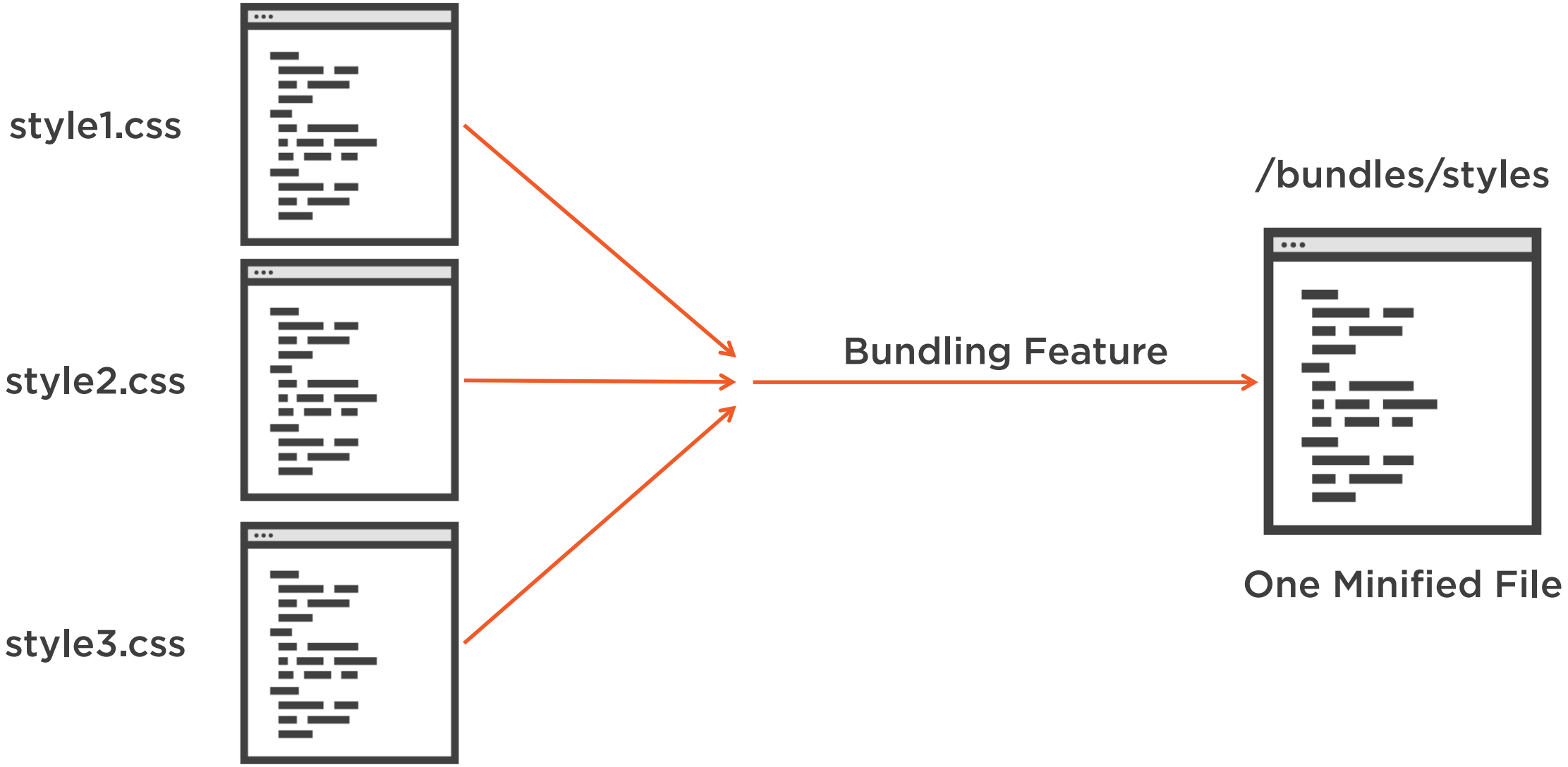
The ViewStart File

ViewStart

```
Layout = "~/Views/Shared/_La

//Other code and markup
```

View

Bundling in MVC



Building HTML Components in Web Forms

The Types of Server Controls

HTML Server Controls

Web Server Controls

Validation Controls

User Controls

```
<div class="wrapper">
    <label ID="subjectLabel" runat="server" for="subject"></label>
</div>
```

HTML Server Controls

Regular HTML empowered by Web Forms attributes

```
<div class="wrapper">

    <asp:TextBox ID="Subject" runat="server" CssClass="form-    control"></asp:TextBox>

    <asp:DropDownList CssClass="form-control" runat="server"
    ID="Category"></asp:DropDownList>

</div>
```

Web Server Controls

Abstraction over HTML, deeply integrated with Web Forms framework

```
<div class="wrapper">

    <asp:RequiredFieldValidator ID="DescriptionValidator"    ControlToValidate="Description"
runat="server">

    </asp:RequiredFieldValidator>

</div>
```

Validation Server Controls

Ensures valid data is contained in other Server Controls

User Controls

```
<div id="wrapper">
        <UserControls:Nav runat="server" id="Nav" />
</div>
```

- User Control embedded in an ASPX page

```
//Separate File
<div class="navbar-header">
        //Navigation markup
</div>
```

- User Control source file

Controlling HTML Through Code

ASPX Page

```
<asp:TextBox ID="Subject" runat="server"
></asp:TextBox>

<asp:TextBox ID="Description"
runat="server”></asp:TextBox>
```

Markup

Code Behind Page

```
Subject.Text = task.Title;

Description.Text = task.Description;
```

C# Logic

The Razor View Engine

```
<div class="wrapper">
    @foreach(var message in messages){
        <span class="topic">@message.Subject</span>
    }
</div>
```

An Intelligent View Engine

Fluid transitions between HTML and Razor syntax

```
<div id="wrapper">

    <div id="widgets">
    @Html.ActionLink("Create Task",    "Create",
"Task")

    @Html.Partial("SurveyWidget")

    @Url.Content("~/images/cat.png")
    </div>

</div>
```

- Renders a link to the create tasks page
- Inserts the contents inside of the SurveyWidget partial view
- Returns a string version of the URL to the specified item

Razor Helpers for Forms Elements

Razor Form Helper	HTML Equivalent
@Html.TextboxFor()	<Input type="text">
@Url.CheckboxFor()	<input type="checkbox">
@Url.DropDownListFor()	<select>[options]</select>

```
@model Task

<div id="wrapper">
    <div id="widgets">
    @Html.TextboxFor(m => m.Subject)

    @Html.TextboxFor(m => m.Date)

    @Html.CheckboxFor(m => m.IsDone)
    </div>
</div>
```

- The Model for the View and Form
- Strongly typed HTML helpers that render form fields for the Model properties

HTML Helpers generate
MVC friendly form fields.

```
<div id="wrapper">
    @foreach(var item in messages){
        <tr>
            <td>@item.Subject</td>
        </tr>
    }
    @using(Html.BeginForm()) {
        //Form Contents
    }
    @if(@Model.IsDone){
        //Conditionally show
    }
</div>
```

- For Each loop to build elements
- Using statement for easy form creation
- Conditionally render markup

```
<div class="wrapper">
    @{
        var task = new Task();
        task.Subject = "Hello World";
        //Other Logic
    }
</div>
```

C# Code Blocks in Razor

Use with caution – is this really necessary?

Working with Razor

Summary

Web Forms and MVC share similar design concepts, but different implementations

Web Forms offers Master Pages, Pages and User Controls

MVC provides Layouts, Views and Partial Views

Web Forms uses Server Controls to work with HTML and framework features

MVC offers lightweight but useful HTML Helpers through the Razor View Engine

Working with Forms

Alex Wolf

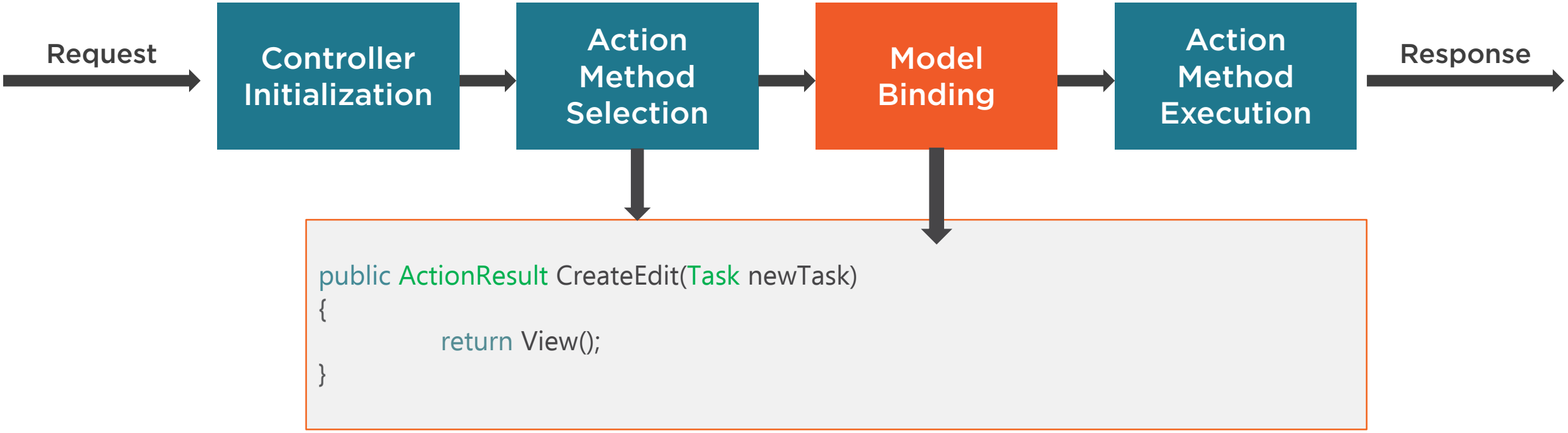
www.alexwolfthoughts.com

To-Do List

- Reviewing Form Submissions in Web Forms
- Introducing Model Binding in MVC
- Demo: Stepping Through the Data Layer
- Demo: Rebuilding the Task Form in Razor
- Demo: Handling Submitted Form Data

Introducing Model Binding

Model Binding and the Request Life Cycle



The Types of Value Providers

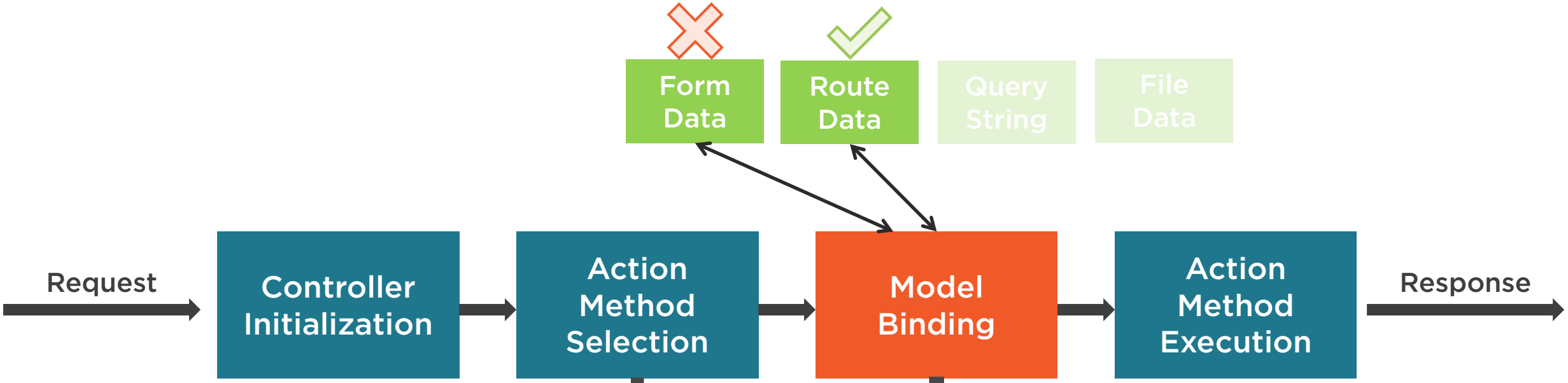
Form Value Provider

Route Data Value Provider

Query String Value Provider

File Value Provider

Supplying Data with Value Providers




```
public ActionResult CreateEdit(int id)
{
        return View();
}
```

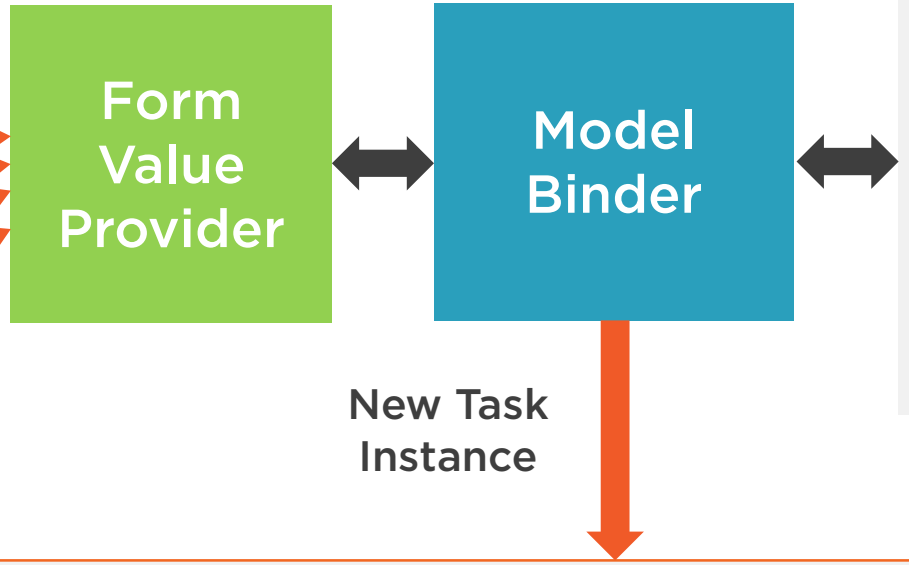
Binding Complex Types

HttpRequest (Simplified)

```
POST /Task/CreateEditTask
Host: localhost:64201

Title=Follow+up
&CategoryId=4
&Notes=Very+Important
&Description=Very+Urgent
```

Task Class

```
public class Task {
  public string Title{}
  public string Category{}
  public string Notes{}
  public string Description{}
}
```




```
public ActionResult CreateEdit(Task id)
{
        return View();
}
```

```
public interface IModelBinder
{
    object BindModel(ControllerContext controllerContext,    ModelBindingContext
bindingContext);
}
```

The IModelBinder Interface

For custom behavior, extend the DefaultModelBinder, or create your own

```
public interface IValueProvider
{
    bool ContainsPrefix(string prefix);
    ValueProviderResult GetValue(string key);
}
```

The IValueProvider Interface

Add your own data sources to the Model Binding process

Applying Model Binding Concepts

Summary

The Model Binder maps request data to Action Method parameters

Value Providers extract data from the request for the Model Binder to use

Razor Provides helpers to streamline model binding and creating forms

Implementing Data Validation

Alex Wolf

www.alexwolfthoughts.com

To-Do List

- Validating Form Inputs in Web Forms
- Understanding Data Validation in MVC
- Demo: Adding Validation to the Task Form
- Implementing Custom Validation in MVC
- Demo: Creating Custom Data Attributes
- Demo: Customizing Model Level Validation

Exploring Validation in MVC

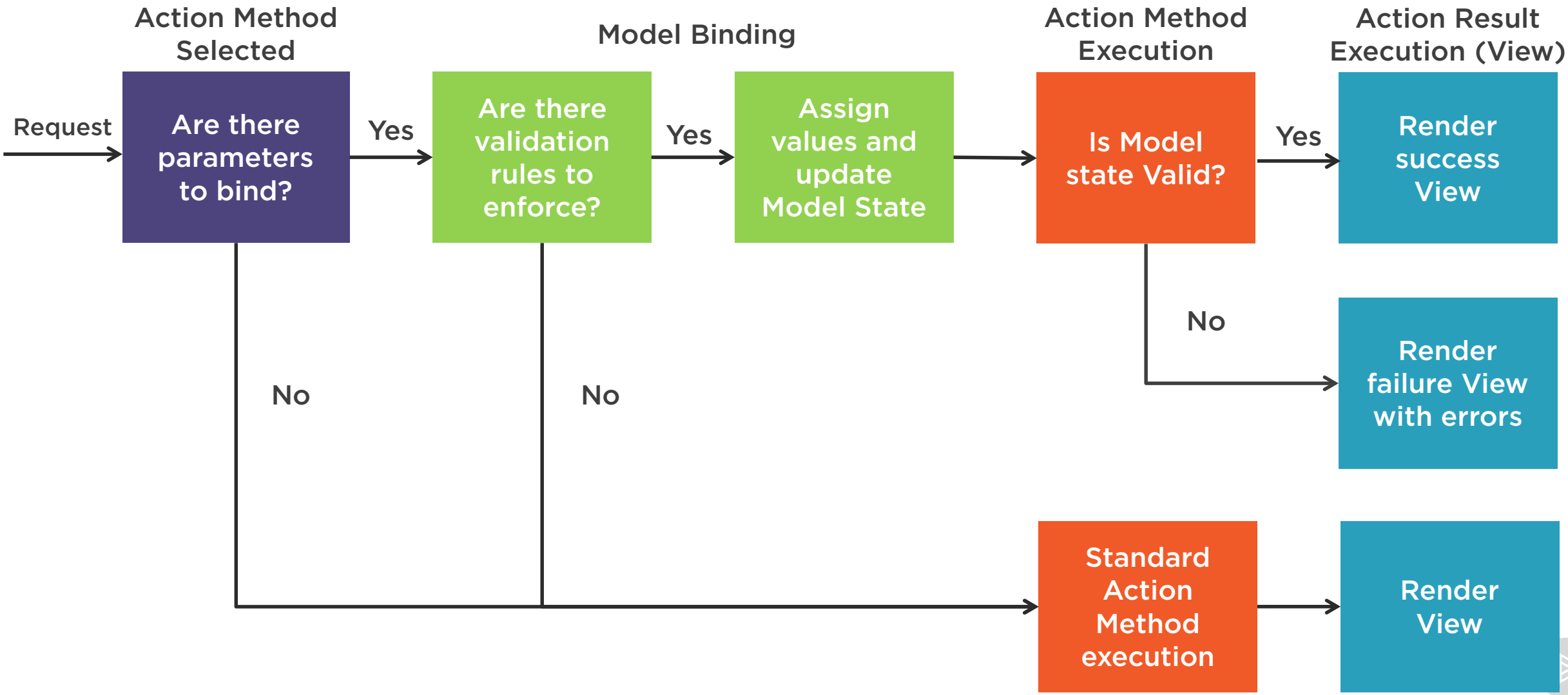
Understanding Data Attributes

```
public class Task
{
        public int Id { get; set; }

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

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

        // Other Properties
}
```

The Default Validation Attributes

Required	StringLength	RegularExpression
Range	Compare	Remote

Data Validation and the MVC Pipeline



```
<div class="form-group">
    @Html.LabelFor(x => x.Description, "Description")
    @Html.TextAreaFor(x => x.Description)
    @Html.ValidationMessageFor(x => x.Description)
</div>
```

Razor Validation Helpers

***ValidationMessageFor* renders Property level errors**

```
@Html.ValidationSummary()
<div class="form-group">
    @Html.LabelFor(x => x.Description, "Description")
    @Html.TextAreaFor(x => x.Description)
</div>
```

Razor Validation Helpers (cont.)

***ValidationSummary* renders Model and (optionally) Property level errors**

```
<div class="form-group">

    <input class="form-control" data-val="true" data-val-required="The Title field is required."
id="Title" name="Title" type="text" value="">

</div>
```

Client Side Validation

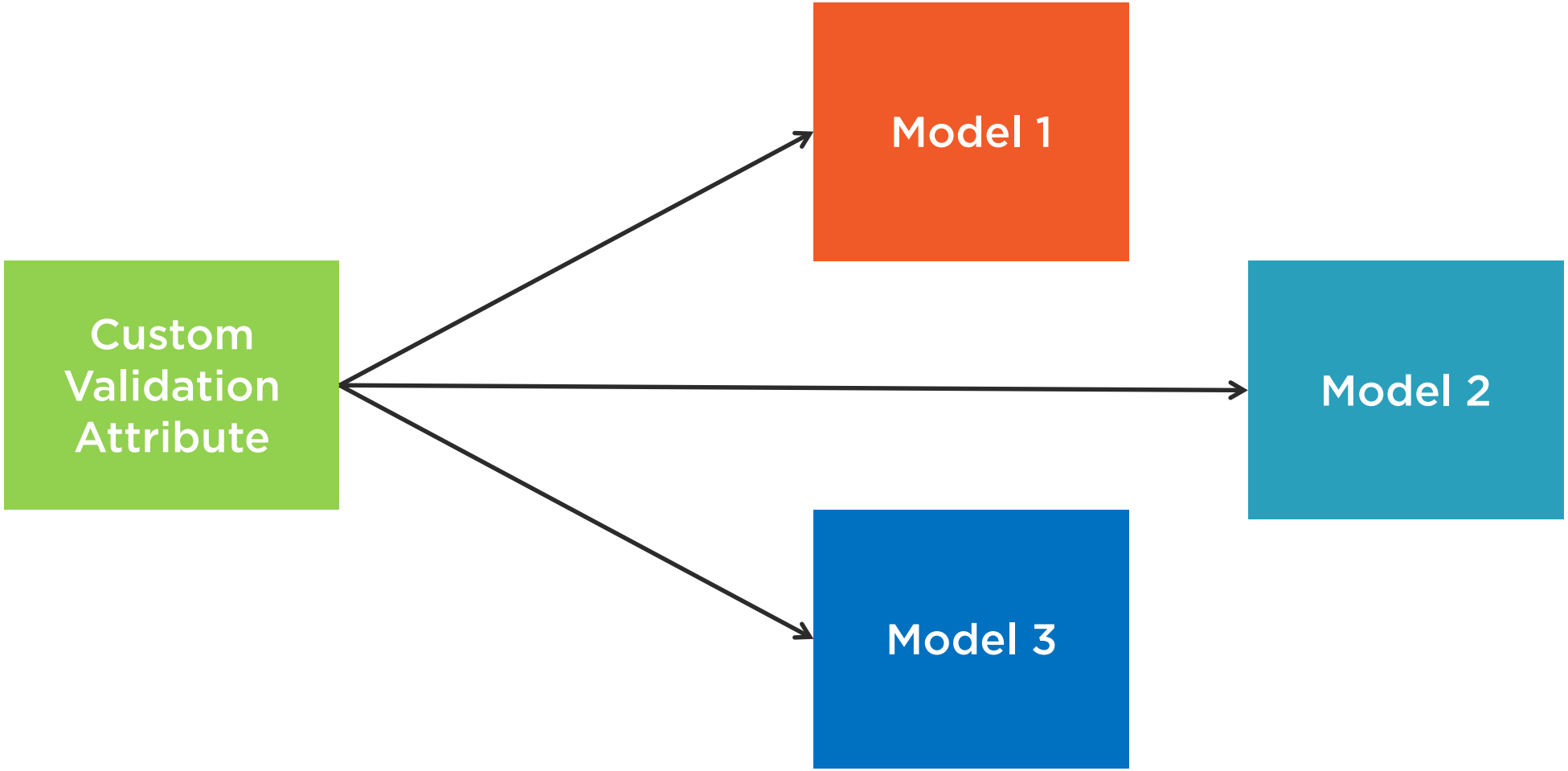
Razor Validation Helpers can auto generate data attributes for validation scripts

Applying Validation in MVC

Customizing Validation in MVC

Creating Custom Data Validation Attributes

Reusable Validation Components



```
public interface IValidatableObject
{
    IEnumerable<ValidationResult> Validate(ValidationContext validationContext)
}
```

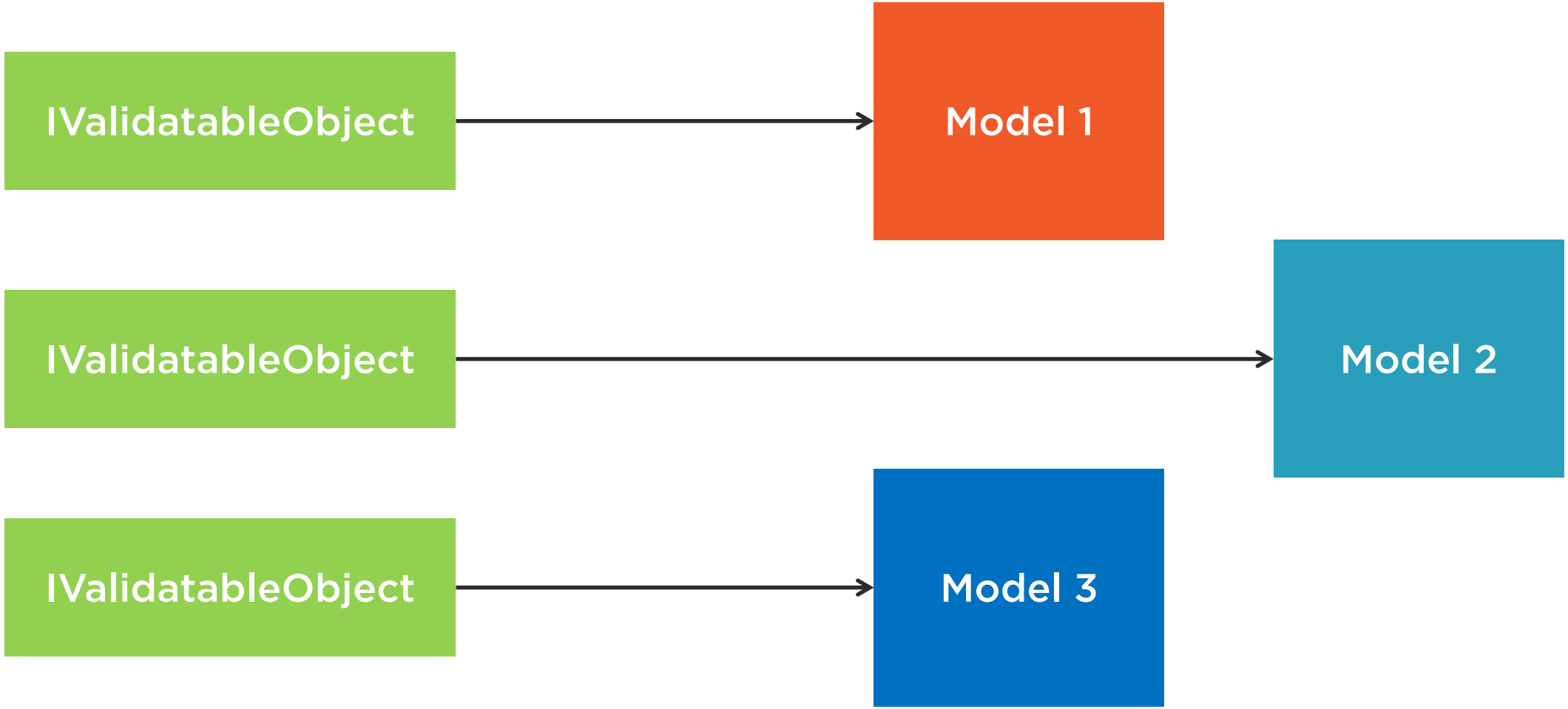
The IValidatableObject Interface

Implemented on the Model rather than a specific Property

Understanding Model Level Validation

```
public class Task
{
        public int Id { get; set; }

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

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

        public string Notes { get; set; }

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

Limited Code Reusability



```
public class Task : IValidatableObject
{
        public int Id { get; set; }
        [Required]
        public string Title { get; set; }
        [Required]
        public string Notes{ get; set; }

        public IEnumerable<ValidationResult>
        Validate(ValidationContext        validationContext)
   {
                //Validation Logic
        }
}
```

- Often not ideal to include logic in models

Summary

Web Forms validates form data through associated server controls

MVC implements validation through the Model Binding process

Data Attributes or the IValidatableObject interface can be used to apply validation

Razor offers helper methods to streamline displaying error messages

Both framework implementations offer client side enhancements as well

A Short Detour

Understanding Partial Views and Child Actions

Alex Wolf

www.alexwolfthoughts.com

To-Do List

- Exploring User Controls in Web Forms
- Introducing Child Actions in MVC
- Demo: Extracting the Widgets into Child Actions
- Demo: Completing the Suggestion Widget
- Demo: Completing the Survey Widget

Introducing Child Actions

Extracting Markup with Partial Views

Welcome to the Feedback Application!

Here you can manage the messages and tasks for our customers.

Latest Messages

From	Subject	Sent	Action
mark@sample.com	Concerned Customer	1/17/2016 11:01:39 AM	View
joe@sample.com	Question about shipping	1/17/2016 11:01:39 AM	View
sam@sample.com	Mobile app troubles	1/17/2016 11:01:39 AM	View
alex@sample.com	Poor service	1/17/2016 11:01:39 AM	View
john@sample.com	Like the new site!	1/17/2016 11:01:39 AM	View

See all

Admin Tasks

Title	Description	Due Date	Action
Follow up with marketing about materials	A vendor has not received shipments for an order they placed through the online form.	1/24/2016 11:01:39 AM	View
Vendor issues	A vendor has not received their order, please address this and check that the order was placed.	1/24/2016 11:01:39 AM	View
Poor customer relations	Multiple customer complaints have come through about our phone service - please contact customer relations to verify that procedures are being followed.	1/24/2016 11:01:39 AM	View

See all

Rep of the Month

You can only vote once!

User	Add Vote
Jane	Vote Up
Sarah	Vote Up
Joe	Vote Up
Bob	Vote Up

Continuous Improvement

Have a feature request or suggestion for how we can improve this app? Let us know! Send us a suggestion.

Submit

Partial View

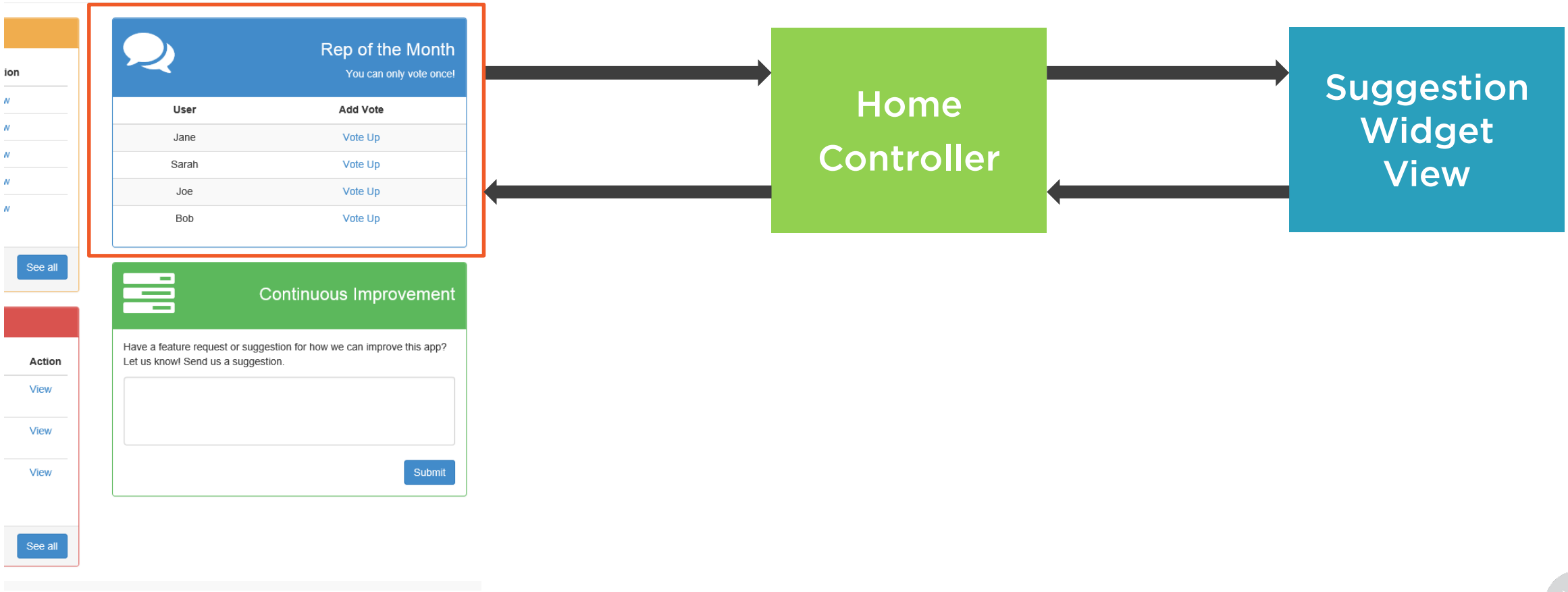
Partial View

Child Actions

Controller Action Methods that are called from within a View

Understanding Child Actions

@Html.Action("SuggestionWidget")



Benefits of Child Actions

- Prevent Code Duplication
- Isolate Complex Sections of Code
- Utilize Multiple Controllers for a Single Page

```
<div class="form-group">
    @Html.Action("Survey", "Home", new { adminId = 3 })
</div>
```

Child Action Parameters

Like standard Action Methods, Child Actions can execute with parameters and Model Binding

Implementing Child Actions in MVC

Summary

Child Actions Are Action Methods Executed from a View

Enable Isolation of Complex Components into Partial Views

MVC Supports Working with Multiple Forms on the Same Page

Child Actions Support Model Binding and Parameters like Any Other Action

A Short Detour

Enhancing the Application with Ajax

Alex Wolf

www.alexwolfthoughts.com

To-Do List

Understanding Ajax in MVC

Demo: Working with Razor Ajax Helpers

Demo: Custom Ajax Features with jQuery

Demo: Enhancing the Task Form Using Ajax

Understanding Ajax in MVC

Ajax in Web Forms and MVC

Handling HttpRequests with Action Methods

localhost/Task/Create

Full Page Request

localhost/Task/GetSuggestions

Ajax Request for Form Auto Complete

```
public class TaskController {

  public ActionResult Create()
  {
        return View();
  }

  public ActionResult GetSuggestions()
  {
        return View();
  }

}
```

The Razor View Engine and Ajax

```
<div class="form-group">
    @{
        var options = new AjaxOptions(){
            HttpMethod = “Post”,
            InsertionMode = InsertionMode.ReplaceWith
        }
    }
    @Ajax.ActionLink("Send Another", "Suggestion", options)
</div>
```

Ajax Options

An object required by Razor Ajax Helpers to configure request and response behavior

```
<div class="form-group">

        <form action="/Home/Suggestion" data-ajax="true" data-ajax-method="Post" data-ajax-
mode="replace-with" data-ajax-update="#suggestionWidget" id="form0" method="post">

</div>
```

Unobtrusive Ajax

Data attributes rendered by Ajax Helpers are utilized by the unobtrusive Ajax libraries

Working with Ajax in MVC

Summary

Razor Ajax Helpers Can Streamline Certain Scenarios

MVC Features Strong Ajax Support with or Without Helpers

Action Methods Can Return Different Types of Results with Ease

The Final Stages

Working with Data

Alex Wolf

www.alexwolfthoughts.com

To-Do List

- Managing and Displaying Data in Web Forms
- Working with Data in MVC
- Demo: Finishing the Home Page Widgets
- Demo: Building the Listing Pages
- Demo: Building the Conversation Page
- Demo: View Models and Model Binding
- Demo: Dependency Injection and Next Steps

Looking Ahead

Working with Data in MVC

Data Workflows in MVC and Web Forms

Data in Web Forms

Server Controls with Various Levels of Abstractions

Data in MVC

Manual Process Using Models and Razor

```
public ActionResult ViewAll()
{
    var context = new FeedbackContext();
    var tasks = context.Tasks.OrderByDescending(x => x.Created).ToList();
    return View(tasks);
}
```

Displaying Data through Models

Models should provide all of the necessary data for display in a View

```
<div class="form-group">
    @foreach (var message in Model)
        {
            <tr>
                <td>@message.Author</td>
                <td>@message.Content</td>
            </tr>
        }
    @Html.DisplayFor(x => x.Subject)
</div>
```

Accessing Data Through Razor

Razor Helpers and syntax provide useful workflows for displaying data

Multiple Sets of Data in Views

List<Message>

List<Task>

Welcome to the Feedback Application!

Here you can manage the messages and tasks for our customers.

Latest Messages

From	Subject	Sent	Action
mark@sample.com	Concerned Customer	1/17/2016 11:01:39 AM	View
joe@sample.com	Question about shipping	1/17/2016 11:01:39 AM	View
sam@sample.com	Mobile app troubles	1/17/2016 11:01:39 AM	View
alex@sample.com	Poor service	1/17/2016 11:01:39 AM	View
john@sample.com	Like the new site!	1/17/2016 11:01:39 AM	View

See all

Admin Tasks

Title	Description	Due Date	Action
Follow up with marketing about materials	A vendor has not received shipments for an order they placed through the online form.	1/24/2016 11:01:39 AM	View
Vendor issues	A vendor has not received their order, please address this and check that the order was placed.	1/24/2016 11:01:39 AM	View
Poor customer relations	Multiple customer complaints have come through about our phone service - please contact customer relations to verify that procedures are being followed.	1/24/2016 11:01:39 AM	View

See all

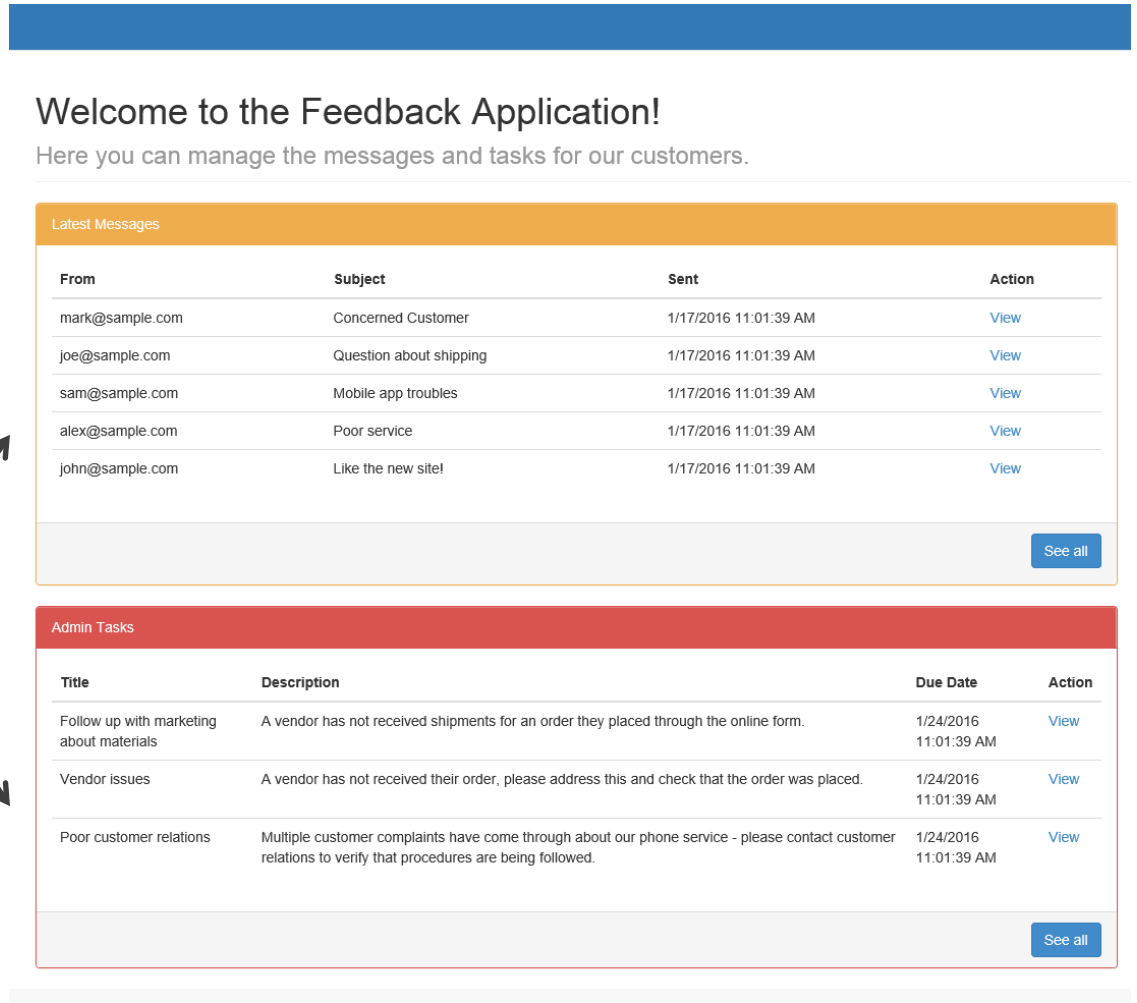
Introducing View Models

List<Message>

List<Task>

View Model

List<Message>
List<Task>



Displaying and Managing Data in MVC

Moving Forward

Other Considerations

Summary

Web Forms Offers Data Controls with Various Levels of Abstractions

MVC Uses Techniques Built Around Models and Razor for Managing Data

View Models Can Provide Benefits for Both Displaying and Binding Data

Dependency Injection Can Improve the Implementation of Data Driven Classes

Thank You, and Best of Luck!