

# Introduction to MVC

# About DevelopMentor



DevelopMentor provides solutions for all professionals involved in the lifecycle of software projects via our consulting, mentoring & learning services.

We're committed to ...

*“Developing People Who Develop Software”*



# About your presenter

- **Brock Allen**
  - ballen@develop.com
- **Author & Instructor:**
  - Essential ASP.NET WebForms
  - Essential ASP.NET MVC
- **Likes:**
  - MVC, jQuery, Ajax
- **Dislikes:**
  - ties, button-down shirts, black & white photos



# Objectives

- **Examine MVC project architecture**
  - routing
  - controllers
  - models
  - views
- **Other features**
  - html helpers
  - model binding
  - validation
  - Ajax



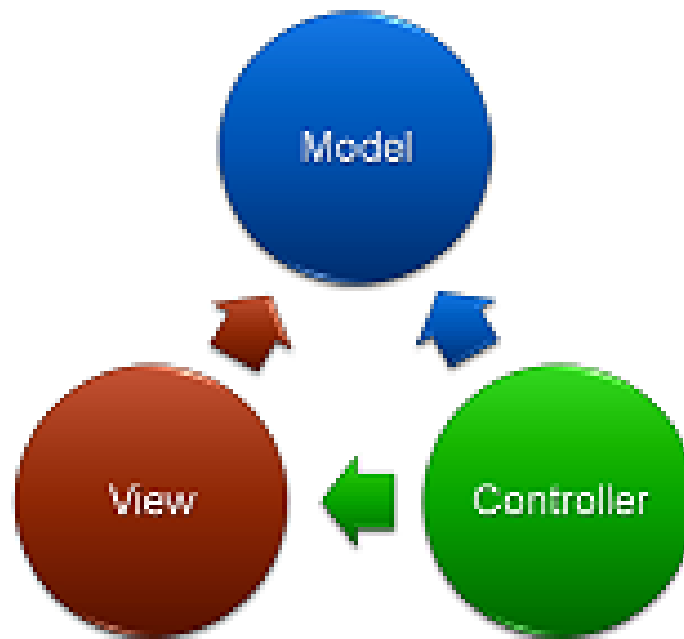
# Problems with ASP.NET Web Forms [motivation]

- **Lack of control over rendered markup**
  - many server controls generate ugly HTML
  - client side CSS and JavaScript sometimes cumbersome
- **Lack of modularity**
  - pages contain input processing and rendering logic
  - pages sometimes even contain data access logic
- **Lack of control over runtime environment dependencies**
  - applications are difficult to test without live web server



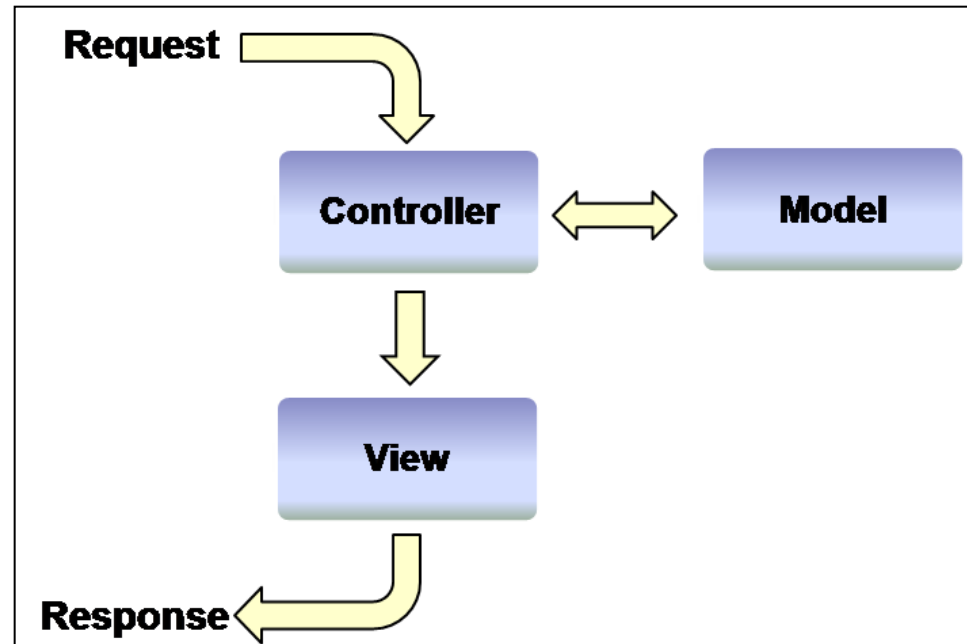
# What is MVC?

- **Model-View-Controller architecture**
  - sometimes called front-controller
  - deliberate separation of concerns



# What is MVC?

- **Controller receives all requests**
  - processes all input
  - accesses model
  - chooses response or view
- **Model is data access layer**
  - same as WebForms (but more explicit)
  - no data access in pages
- **View emits response**
  - only contains rendering logic
  - passed data from controller



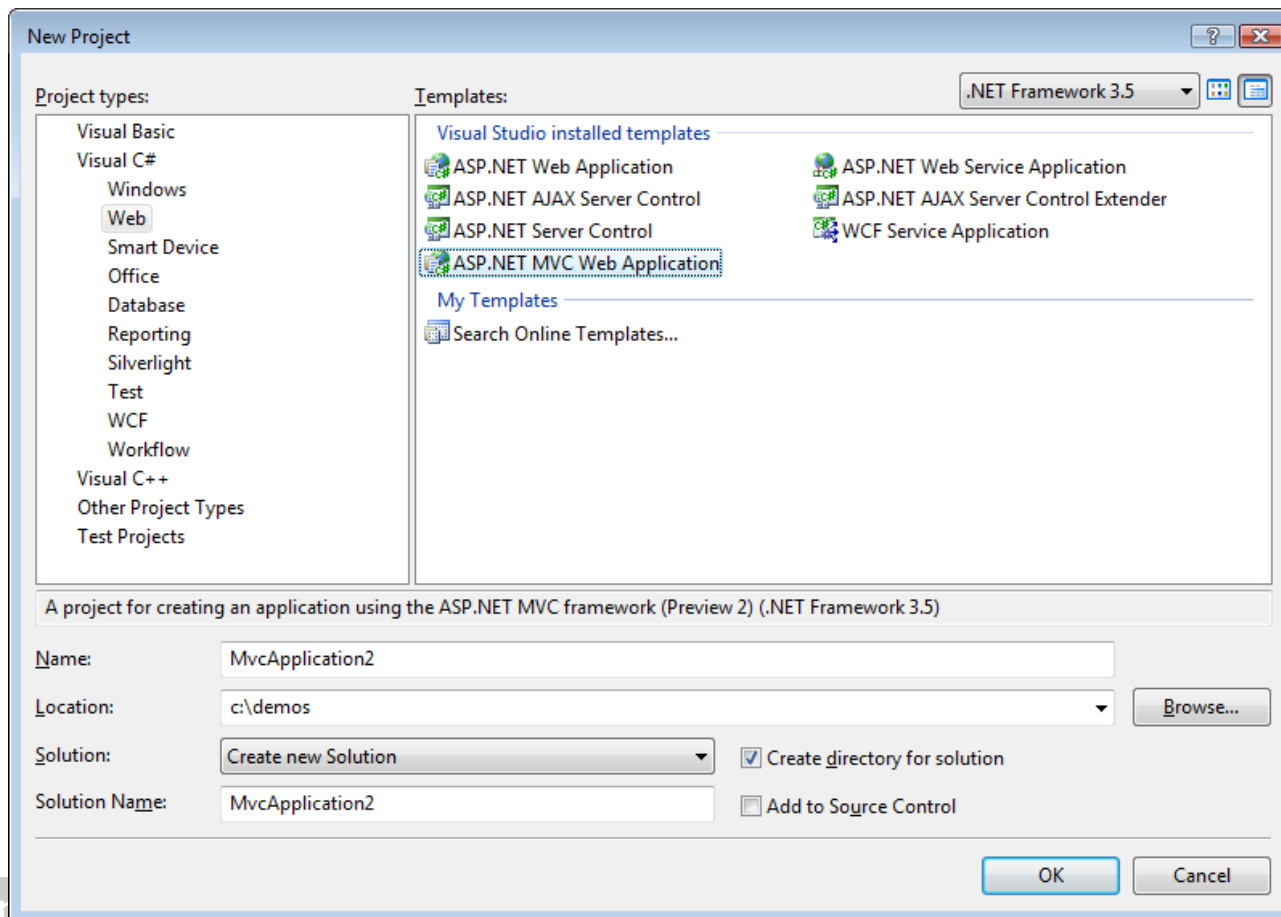
# What is ASP.NET MVC?

- **Framework**
  - maps requests onto controller classes and methods
  - WebForms view engine for rendering ASPX pages
  - extensible, customizable and designed for testing
- **Assemblies**
  - System.Web.Abstractions
  - System.Web.Mvc



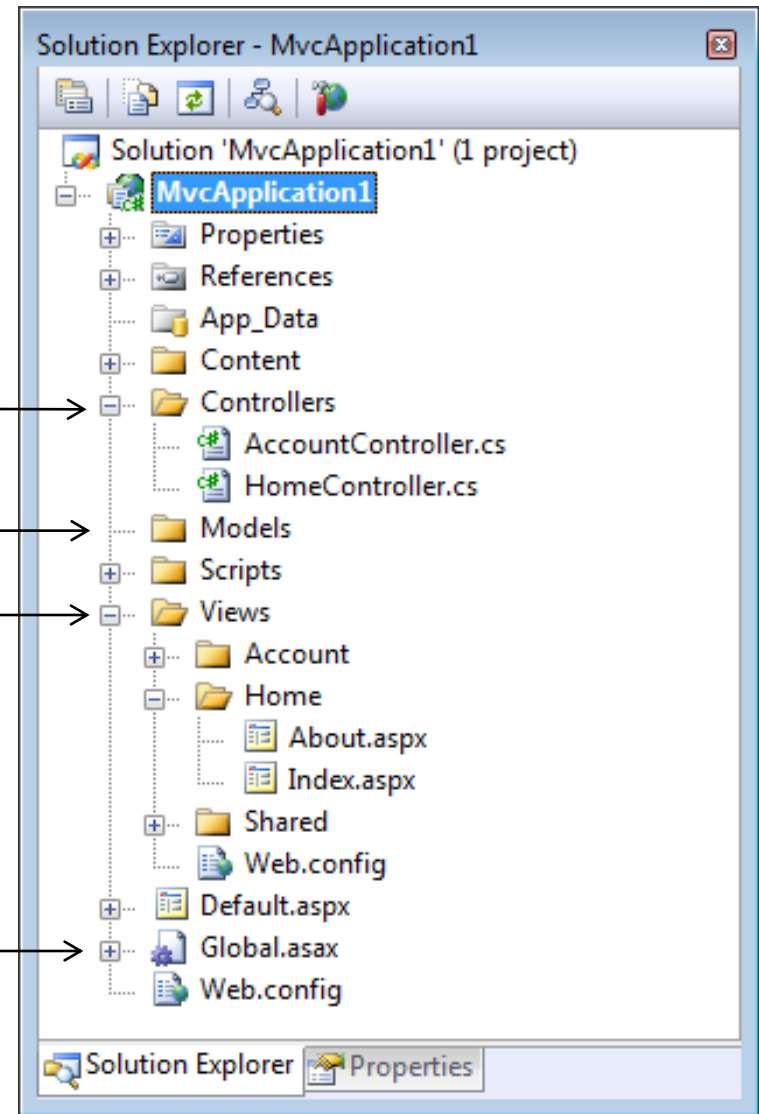
# ASP.NET MVC and Visual Studio

- **ASP.NET MVC installs a web application project template**
  - web site project types not supported



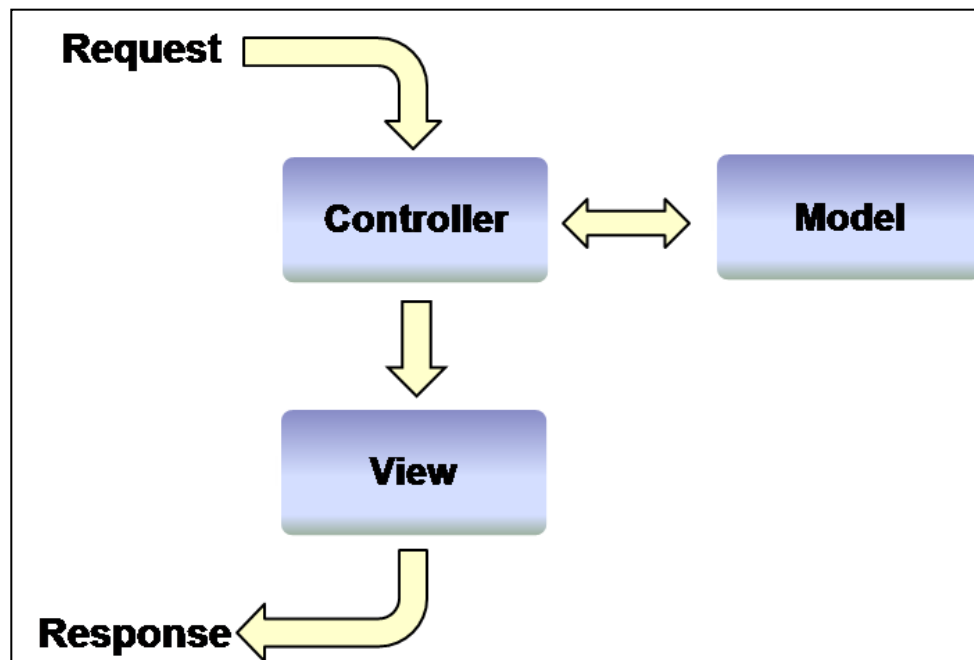
# ASP.NET MVC project files

- **Controllers live in "Controllers"**
  - inherit **Controller** base
- **Models directory for data access code**
  - no special requirements
- **Views live in "Views"**
  - subdirectory with same name as controller
  - classes inherit ViewPage base
- **Routes defined in global.asax**
  - define URL to controller mapping



# Controllers

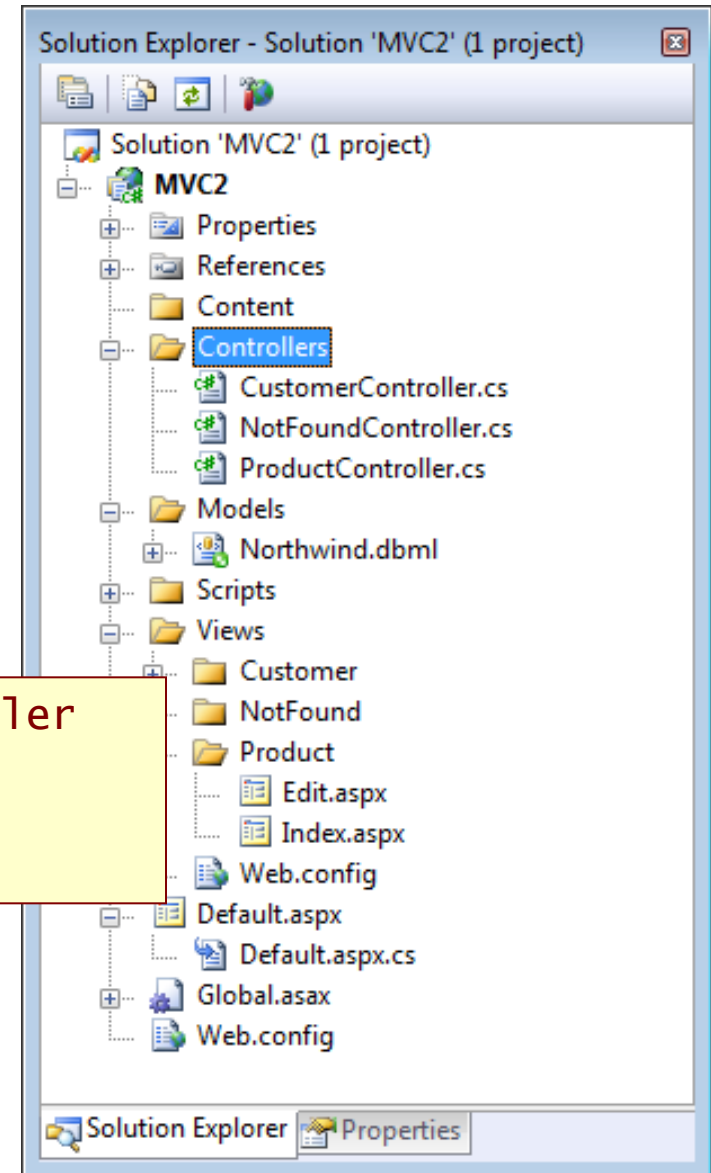
- **Controller is class that receives all requests**
  - handle all input
  - access model for data access
  - determine result to return to client



# Controller class

- **Controller conventions**
  - class resides in "Controllers" directory
  - class name must end with "Controller"
  - derives from **Controller** base class

```
public class ProductController : Controller
{
    // ...
}
```



# Controllers and requests

- **Requests map onto controller methods**
  - methods represent operations user will be performing
  - called "Actions" or "Action Methods"
  - must return **ActionResult** (more on this later)

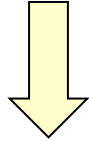
```
public class ProductController : Controller
{
    public ActionResult ShowAll() {
        // ...
    }
    public ActionResult Edit(string id) {
        // ...
    }
    public ActionResult Update(string id,
                               string description, decimal price) {
        // ...
    }
}
```



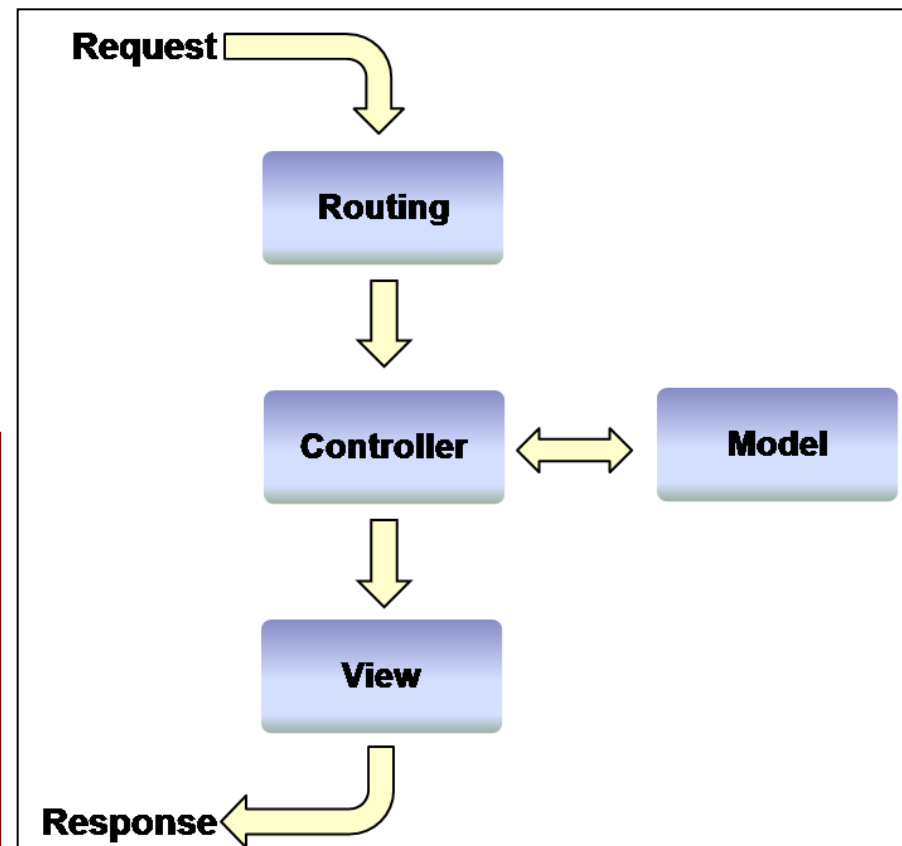
# Controllers and routing

- **MVC heavily utilizes ASP.NET URL Routing**
  - URL indicates controller and action

```
~/Product/ShowAll  
~/Product/Edit/Chai  
~/Product/Update/Chai
```



```
public class ProductController : Controller  
{  
    public ActionResult ShowAll() {  
        // ...  
    }  
    public ActionResult Edit(string id) {  
        // ...  
    }  
    public ActionResult Update(string id,  
                               string description, decimal price) {  
        // ...  
    }  
}
```



# Controllers and routing

- **URL segments used for controller class and action method**
  - segments named {controller} and {action}
- **MapRoute extension method used for simplification**
  - automatically uses **MVCRouteHandler** to map onto controllers
  - registered in global.asax

```
static void RegisterRoutes(RouteCollection routes)
{
    // Example: /Product/ViewAll
    routes.MapRoute("Default",
                    "{controller}/{action}/{id}"
                    new { id="" });

    // Example: /Cart/View
    routes.MapRoute("ShoppingCart",
                    "Cart/{action}",
                    new { controller="ShoppingCart",
                          action="View" });
}
```

# Controller base class

- **Controller** class provides access to runtime information
  - **HttpContext** properties
  - routing data
  - ASP.NET features

```
public abstract class Controller : ControllerBase
{
    public HttpContextBase HttpContext { get; }
    public ModelStateDictionary ModelState { get; }
    public HttpRequestBase Request { get; }
    public HttpResponseBase Response { get; }
    public RouteData RouteData { get; }
    public HttpServerUtilityBase Server { get; }
    public HttpSessionStateBase Session { get; }
    public UrlHelper Url { get; set; }
    public IPrincipal User { get; }
}
```



# Controller input

- **Action methods can accept input data as parameters**
  1. input from POST data
  2. input from routing URL segments
  3. input from query string
- **Input mapped as parameters to action method**

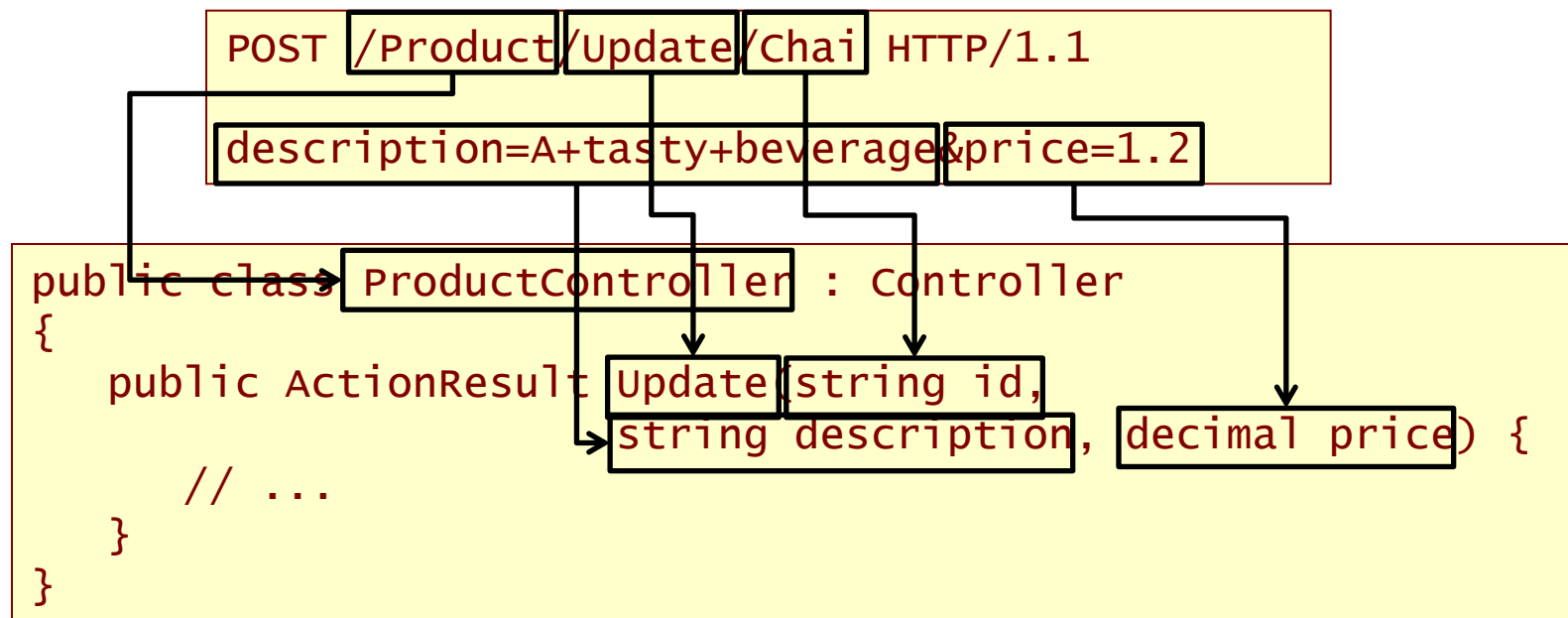
```
POST /Product/Update/Chai HTTP/1.1  
description=A+tasty+beverage&price=1.2
```

```
public class ProductController : Controller  
{  
    public ActionResult Update(string id,  
                               string description, decimal price) {  
        // ...  
    }  
}
```



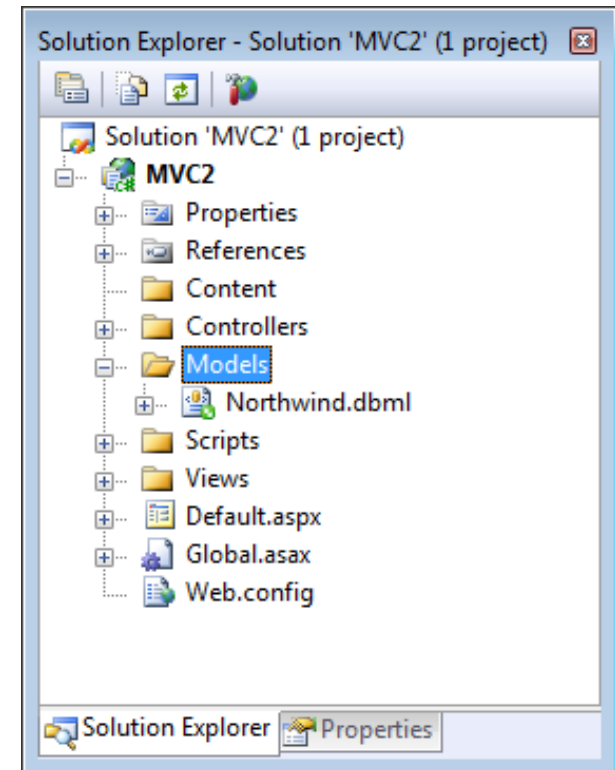
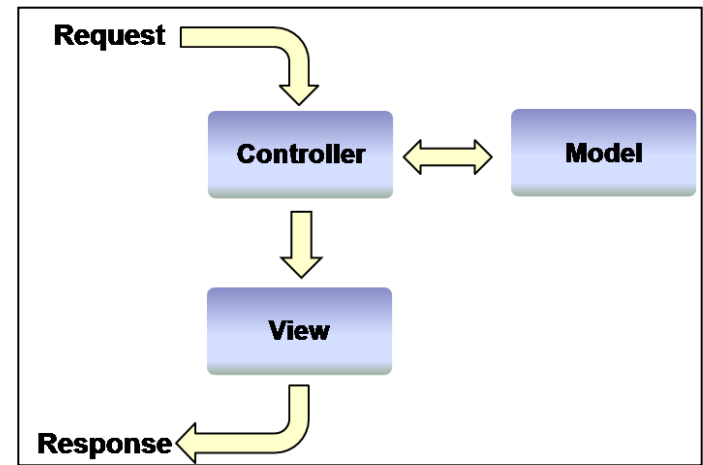
# Controller input

- **Action methods can accept input data as parameters**
  1. input from POST data
  2. input from routing URL segments
  3. input from query string
- **Input mapped as parameters to action method**



# Models

- **Model is data access layer**
  - code resides in "Models" directory
  - abstracts data access from controller and view
  - provides data passed to view for display
- **Proper design of model facilitates testing (TDD)**
  - repository pattern
  - dependency injection/inversion of control



# Action results

- **Controller logic determines result sent to client**
  - action methods must return **ActionResult**
  - **ActionResult** is base class to indicate response to client
    - derived **ContentResult** or **ViewResult** common
    - other derived action results possible
  - **Controller** provides helper APIs to create action results

```
public class ProductController : Controller
{
    public ActionResult showAll() {
        // ...
    }
}
```



# ActionResult

- **ActionResult and Controller.Content** return string
  - string is passed to **Response.Write**

```
public class ProductController : Controller
{
    public ActionResult ShowAll() {
        return Content("<h1>This is all!</h1>");
    }
}
```

- action method can return string literal instead

```
public class ProductController : Controller
{
    public string ShowAll() {
        return "<h1>This is all!</h1>";
    }
}
```



# ViewResult

- **ViewResult** and **Controller.View** indicate view to render
  - pass name of view to render

```
public class ProductController : Controller
{
    public ActionResult ShowAll() {
        return View("ShowAll");
    }
}
```

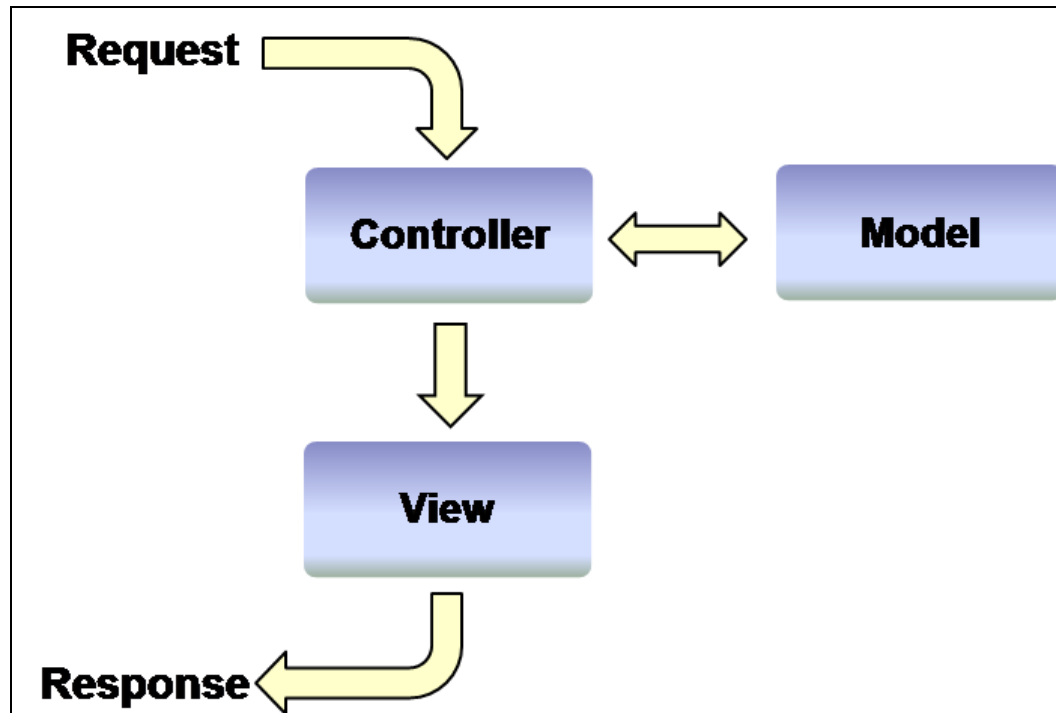
- if no name chosen then view with same name as action used

```
public class ProductController : Controller
{
    public ActionResult ShowAll() {
        return View();
    }
}
```



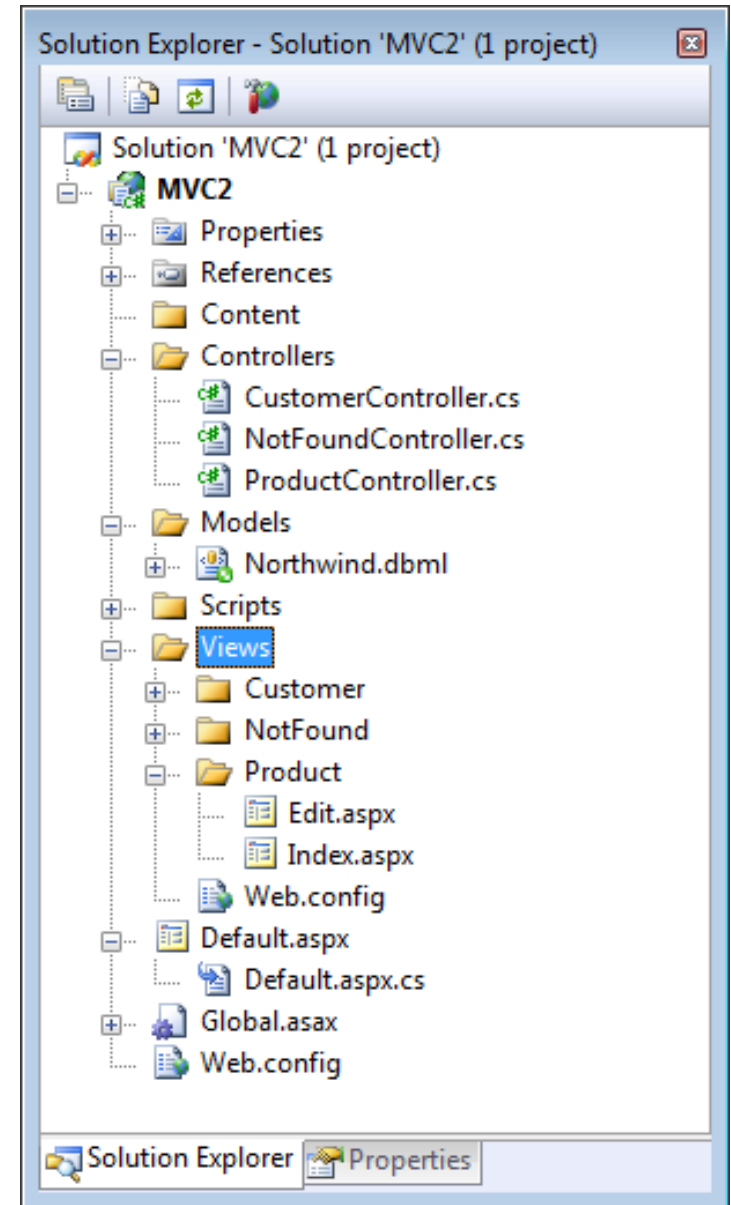
# Views

- **Templates which generate content to return to client**
  - only contains rendering logic
    - no data access
  - typically accepts data from controller



# WebForms view engine

- **ASP.NET MVC uses ASP.NET WebForms as view engine**
  - same view engine from traditional ASP.NET
    - ASPX (and ASCX) files
- **Views reside in "Views" directory**
  - subdirectories for each controller
  - view files named for corresponding controller action
- **Other view engines possible**



# View pages

- **View pages are dynamic templates for markup**
  - `<%@Page%>` directive used to control aspects of processing
    - **Inherits** directive is **ViewPage** or **ViewPage<T>**

```
<%@ Page Language="C#" Inherits="ViewPage" %>
<html>
<body>
    <h1>Hello MVC</h1>
</body>
</html>
```



# Dynamic execution

- Use code blocks used to execute logic
  - `<% code goes here %>`
- Use inline code to emit dynamic values
  - `<%= expression goes here %>`

```
<%@ Page Language="C#" Inherits="ViewPage" %>
<html>
<body>
  <% for(int i = 0; i < 5; i++) { %>
    <h1>Hello MVC, the time is <%= DateTime.Now %></h1>
  <% } %>
</body>
</html>
```



# Passing data to view

- **Controller.ViewData** property holds data to pass to view
  - supports indexer for dictionary-like access (key/value)
  - supports **Model** property for strong typing data
    - **Controller.View** accepts model as parameter

```
public class ProductController : Controller
{
    public ActionResult ShowAll()
    {
        ProductData[] list = Products.GetAll();
        ViewData["Products"] = list; // dictionary
        ViewData.Model = list; // explicit Model
        return View(list); // implicit Model
    }
}
```



# Accessing data in view

- **ViewData** provides access to data from controller
  - **ViewData** implements indexer for dictionary data
- **Model** property provides strongly typed data from controller
  - controlled by using generic parameter to **ViewPage<T>**

```
<%@ Page Inherits="System.Web.Mvc.ViewPage<Product[]>" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Product Listing</title>
</head>
<body>
  <ul>
    <% foreach (var prod in Model) { %>
      <li><%= prod.ProductName %></li>
    <% } %>
  </ul>
</body>
</html>
```



# View and HtmlHelpers

- **HtmlHelpers ease generation of HTML**
  - set of extension methods on **Html** property
  - encapsulate rendering logic
  - abstract route information
    - good to avoid hard coding route URLs into view

```
<%@ Page Inherits="System.Web.Mvc.ViewPage<Product[]>"%>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Product Listing</title>
</head>
<body>
  <ul>
    <% foreach (var prod in Model) { %>
      <li><%= Html.ActionLink(prod.ProductName,
        "Edit", new { id=prod.ProductName })%></li>
    <% } %>
  </ul>
</body>
</html>
```

# View and HtmlHelpers [example]

```
<%@ Page Inherits="System.Web.Mvc.ViewPage<Product>" %>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Edit <%= Model.ProductName %></title>
</head>
<body>
    <% using (Html.BeginForm("Update", "Product")) { %>
        <fieldset>
            <legend>Edit <%= Model.ProductName %></legend>
            <p>
                <label for="description">Description</label>
                <%= Html.TextBox("description", Model.Description) %>
            </p>
            <p>
                <label for="price">Price</label>
                <%= Html.TextBox("price", Model.Price) %>
            </p>
            <p><input type="submit" value="Submit" /></p>
        </fieldset>
    <% } %>
</body>
</html>
```

# Summary

- **ASP.NET MVC is an alternative approach to WebForms**
- **MVC promotes deliberate separation of concerns**
- **MVC promotes maintainable software**
- **MVC allows high degree of control over markup**
- **MVC has comparable features to WebForms**



# Upcoming classes

- **Essential ASP.NET MVC**
  - January 25<sup>th</sup>, Boston MA, USA
  - February 8<sup>th</sup>, London, UK
- **Contact:**
  - <http://www.develop.com/contact>
  - US (LA): 310-988-7700
  - US (Boston): 781-935-9895
  - UK: +44 (0) 1242 525108

