ASP.NET MVC

MVC 5

MVC Core

Eng. Basma Hussien

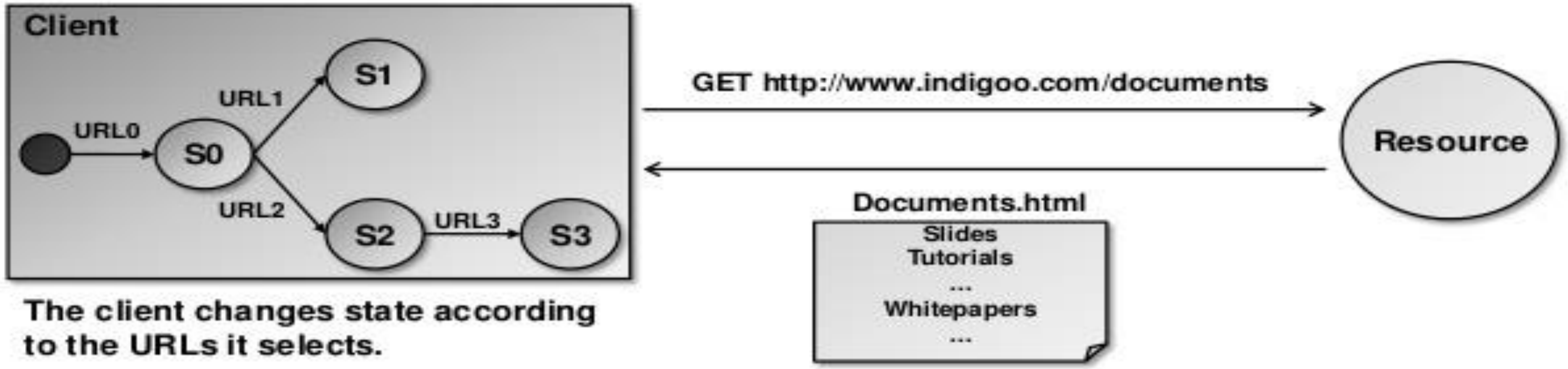
REST

REST – Representational State Transfer

indigoo.com

1. What is „Representational State Transfer“ ? (2/3)

To understand the REST principle, look at what happens in a web access of a browser:



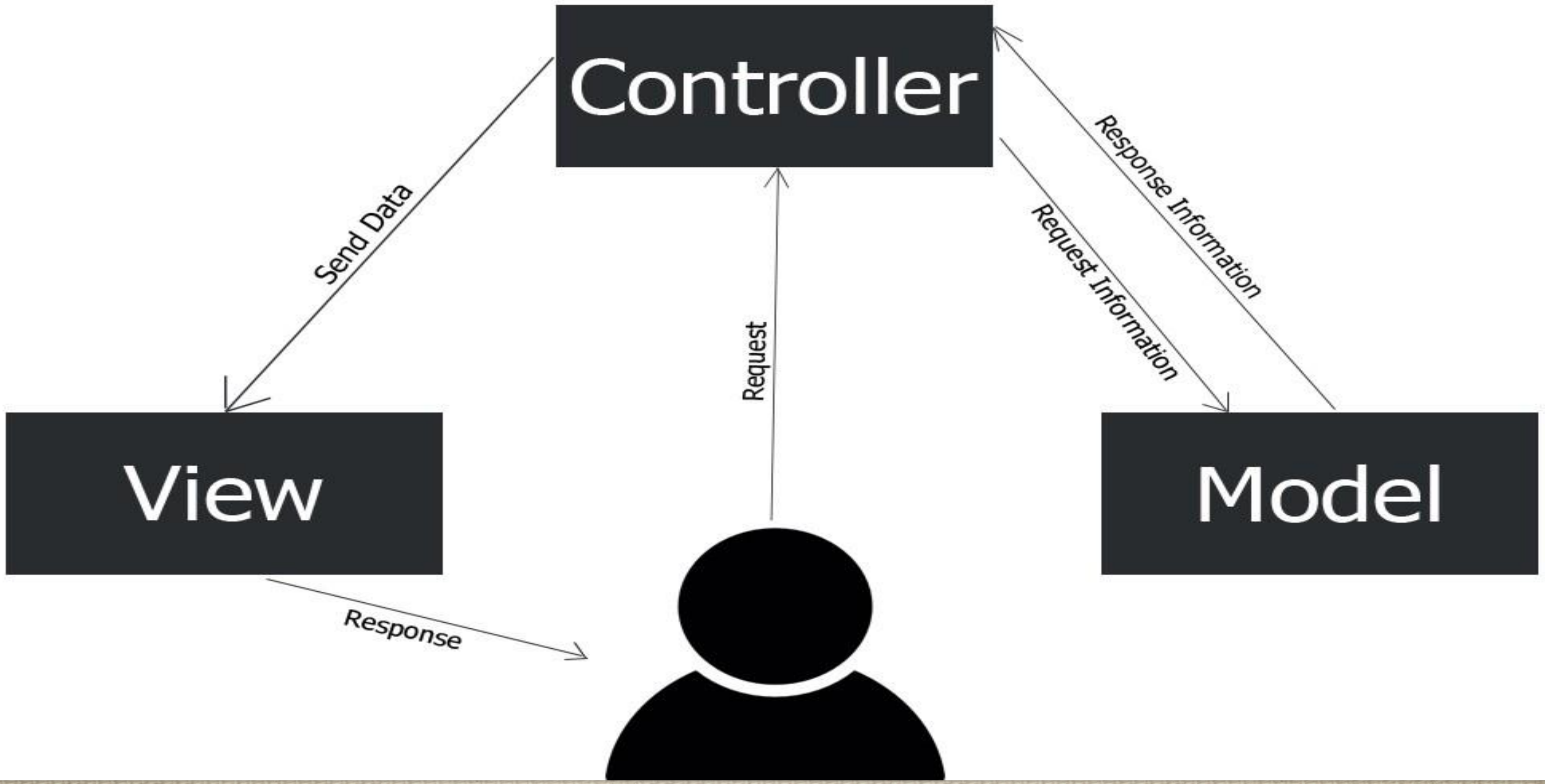
The client changes state according to the URLs it selects.

1. The client references a web resource using a URL.
2. The web server returns a *representation* of the resource in the form of an HTML document.
3. This resource places the client into a new *state*.
4. The user clicks on a link in the resource (e.g. Documents.html) which results in another resource access.
5. The new resource places the client in a new state.

➔ The client application changes (=*transfers*) *state* with each resource *representation*.

MVC Pattern

Model-View-Controller



Features of ASP.NET MVC Framework

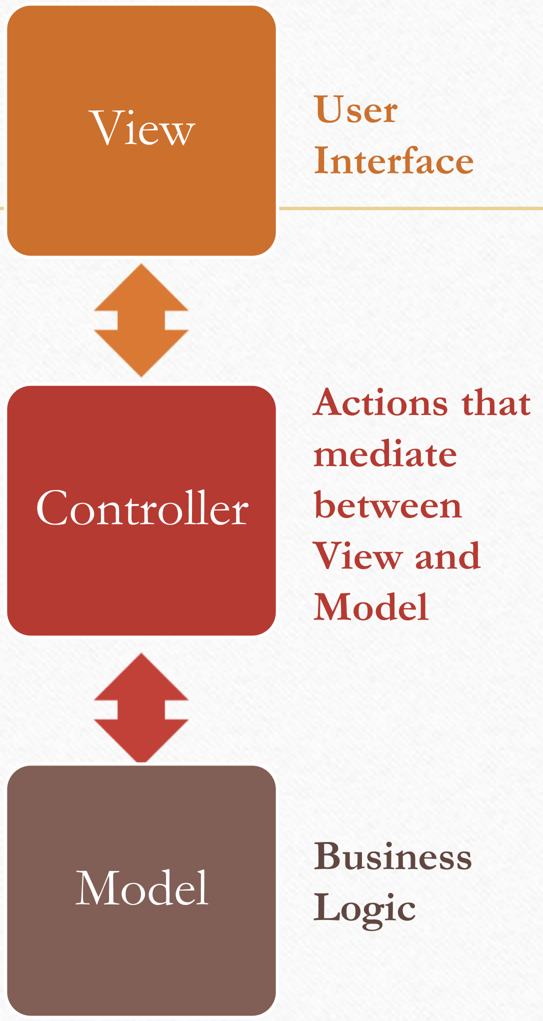
- **Separation** of application logic which makes testing easier
- **Powerful URL-mapping** component that lets you build applications that have comprehensible and searchable URLs. URLs do not have to include file-name extensions.
- Using Razor as a view Engine

MVC Components

Views. Template files that your application uses to dynamically generate HTML responses.

Controllers. Classes that handle incoming browser requests, retrieve model data, and then specify view templates that return a response to the browser..

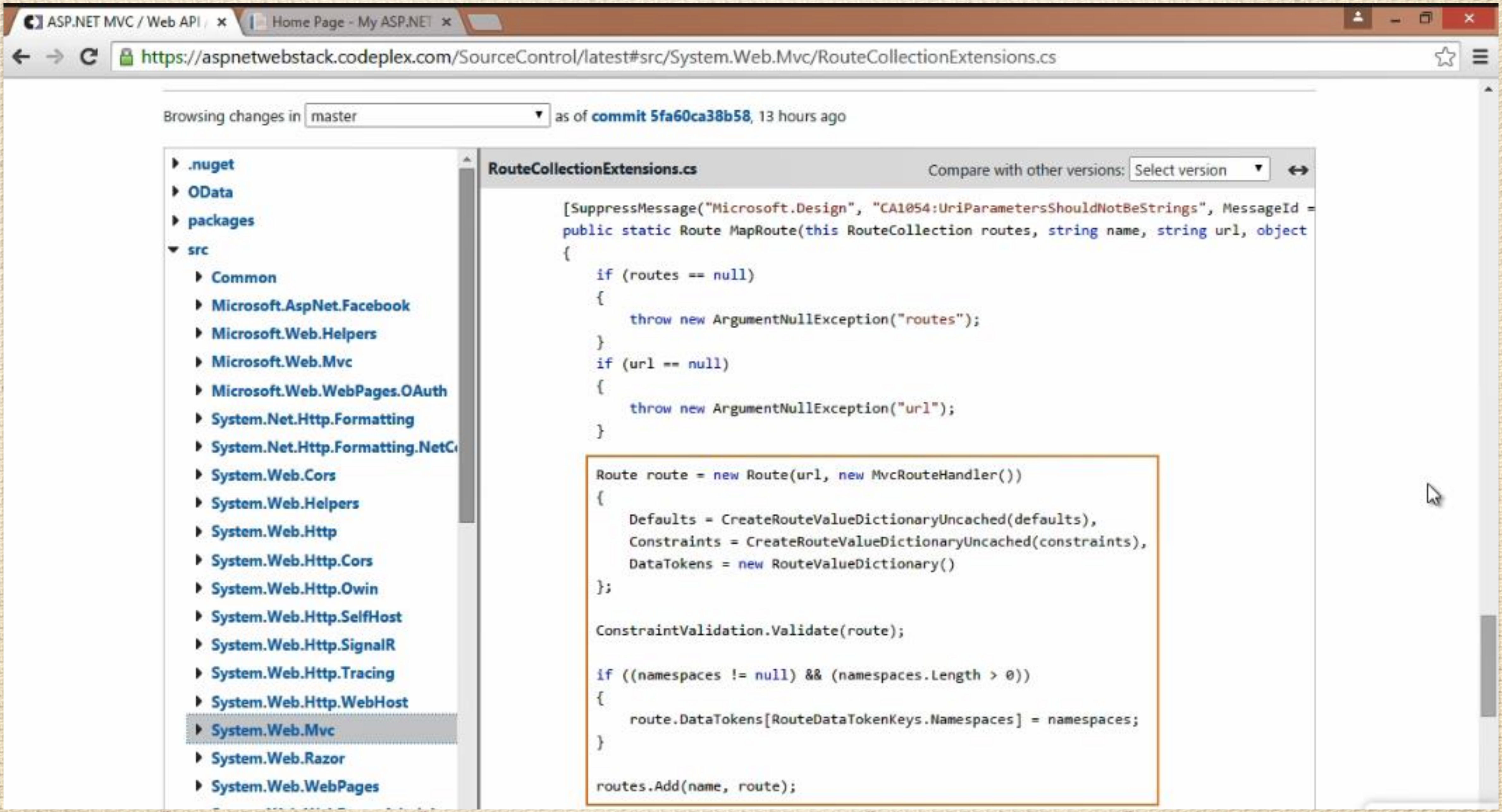
Models. Classes that represent the data of the application and that use validation logic to enforce business rules for that data.



ASP.NET Routing “*Routing*”

- ASP.NET MVC generally take a different approach by mapping the URL to a method (Action) call on a class (Controller) , rather than some physical file.

- MVC 5 adds a second option using declarative attributes on your controller classes and action methods, which is called **attribute routing**.

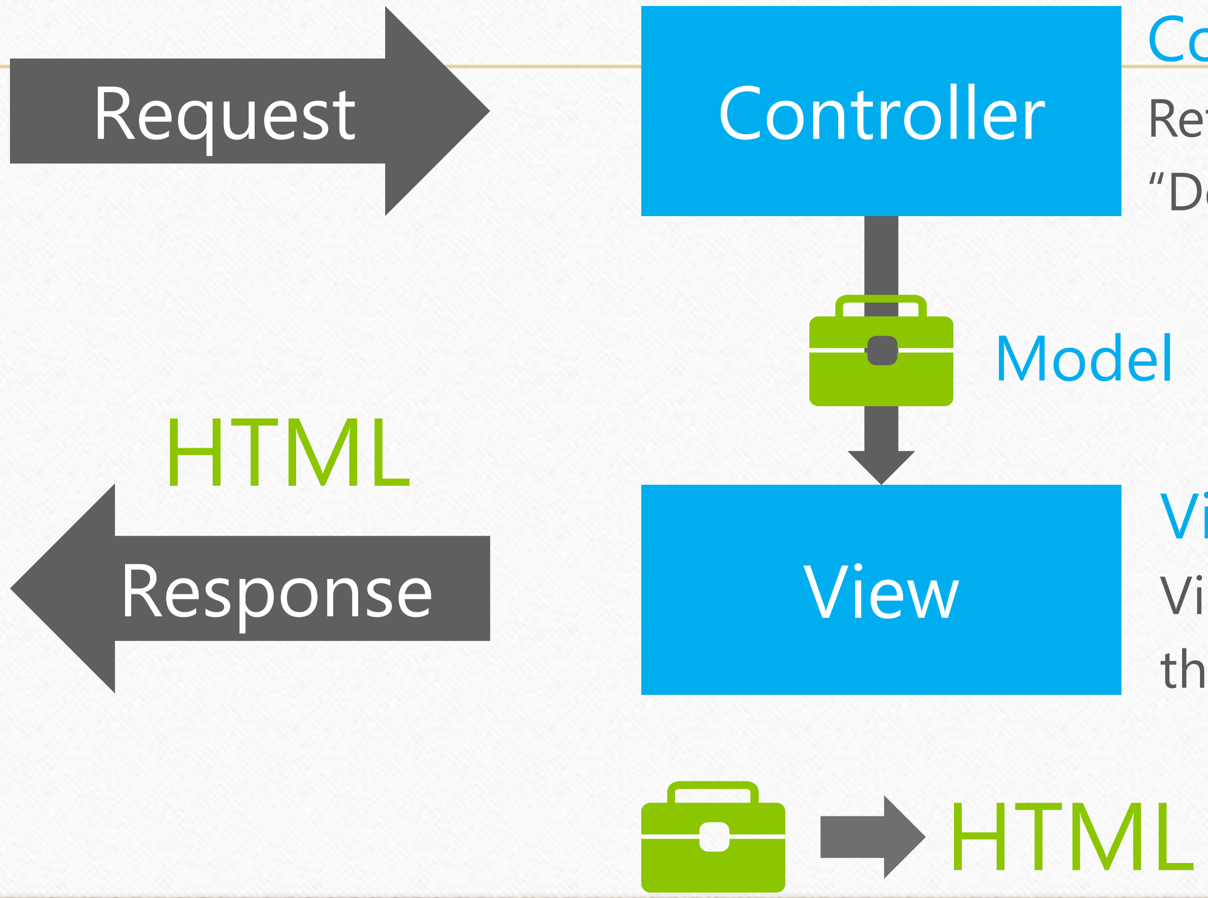


ASP.NET MVC / Web API
Home Page - My ASP.NET
https://aspnetwebstack.codeplex.com/SourceControl/latest#src/System.Web.Mvc/RouteCollectionExtensions.cs
Browsing changes in master as of commit 5fa60ca38b58, 13 hours ago
.nuget
OData
packages
src
Common
Microsoft.AspNet.Facebook
Microsoft.Web.Helpers
Microsoft.Web.Mvc
Microsoft.Web.WebPages.OAuth
System.Net.Http.Formatting
System.Net.Http.Formatting.NetC
System.Web.Cors
System.Web.Helpers
System.Web.Http
System.Web.Http.Cors
System.Web.Http.Owin
System.Web.Http.SelfHost
System.Web.Http.SignalR
System.Web.Http.Tracing
System.Web.Http.WebHost
System.Web.Mvc
System.Web.Razor
System.Web.WebPages
RouteCollectionExtensions.cs
Compare with other versions: Select version

```
[SuppressMessage("Microsoft.Design", "CA1054:UriParametersShouldNotBeStrings", MessageId =
public static Route MapRoute(this RouteCollection routes, string name, string url, object
{
    if (routes == null)
    {
        throw new ArgumentNullException("routes");
    }
    if (url == null)
    {
        throw new ArgumentNullException("url");
    }

    Route route = new Route(url, new MvcRouteHandler())
    {
        Defaults = CreateRouteValueDictionaryUncached(defaults),
        Constraints = CreateRouteValueDictionaryUncached(constraints),
        DataTokens = new RouteValueDictionary()
    };

    ConstraintValidation.Validate(route);

    if ((namespaces != null) && (namespaces.Length > 0))
    {
        route.DataTokens[RouteDataTokenKeys.Namespaces] = namespaces;
    }

    routes.Add(name, route);
```

Models, Views, and Controllers

What does MVC look like?



Controller

Retrieves Model

"Does Stuff"

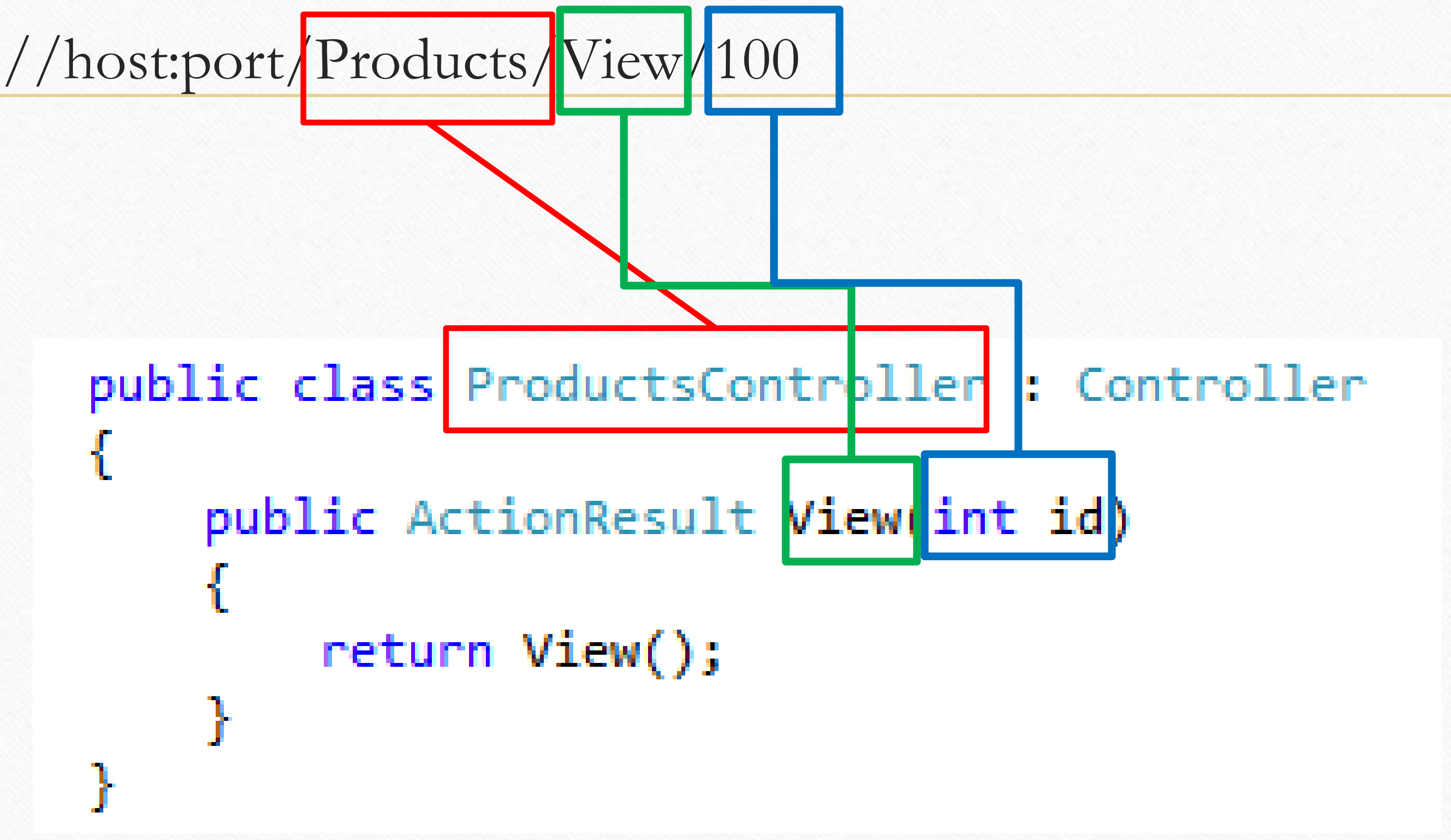
View

Visually represents the model

Demo

URL Mapping

- http://host:port/Products/View/100



```
public class ProductsController : Controller
{
    public ActionResult View(int id)
    {
        return View();
    }
}
```

Controller

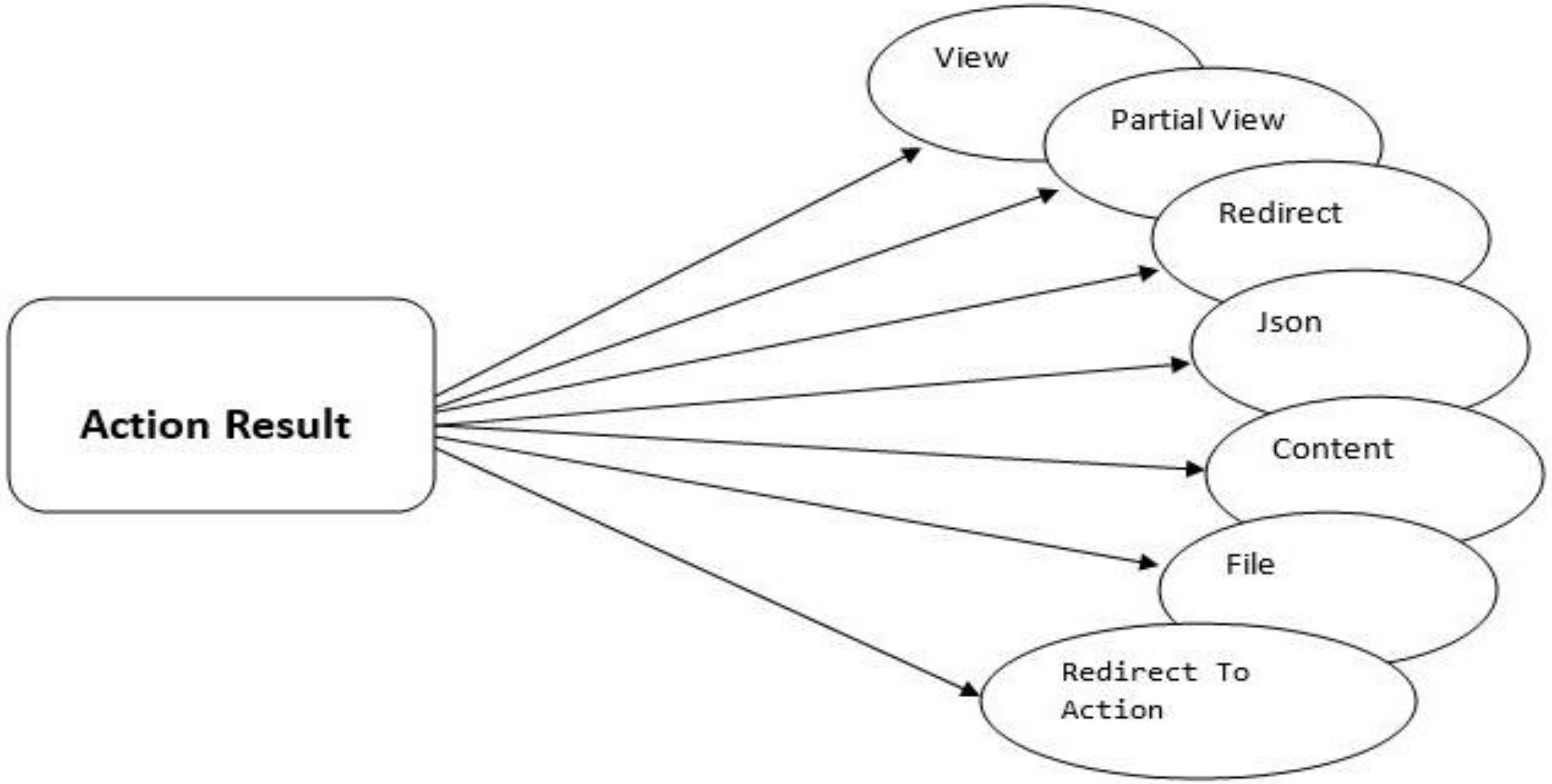
- The Controller in MVC architecture handles any incoming URL request.
- The Controller is a class, derived from the base class ***System.Web.Mvc.Controller***.
- Controller class contains public methods called ***Action*** methods.
- Controller and its action method handles incoming browser requests, retrieves necessary model data and returns appropriate responses.
- Every controller class name must end with a word "Controller“, Ex: ***ProductController***.

Controller & ActionResult

- A controller action returns something called an ActionResult.
- An action result is what a controller action returns in response to a browser request.
- Action result is an **abstract class**. It is a base class for all type of action results.

Action Result

Action Result is a result of action methods or return types of action methods. Action result is an abstract class. It is a base class for all type of action results.



The ASP.NET MVC framework supports several types of action results including:

- ViewResult – Represents HTML and markup.
- EmptyResult – Represents no result.
- RedirectResult – Represents a redirection to a new URL.
- JavaScriptResult – Represents a JavaScript script.
- ContentResult – Represents a text result.

Normally, you do not return an action result directly. Instead, you call one of the following methods of the Controller base class:

- View – Returns a ViewResult action result.
- Redirect – Returns a RedirectResult action result.
- RedirectToAction – Returns a RedirectToRouteResult action result.
- JavaScriptResult – Returns a JavaScriptResult.
- Content – Returns a ContentResult action result.

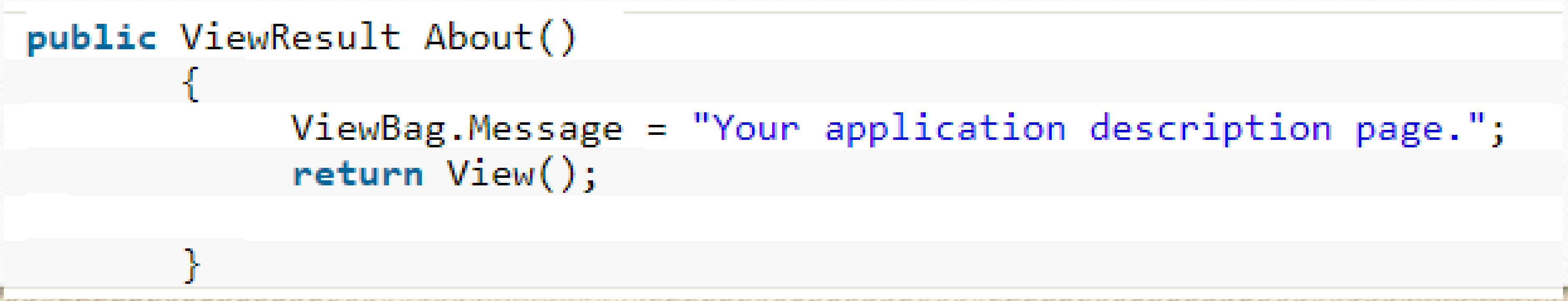
Results

- **Actions typically return an ActionResult**

Name	Framework Behavior	Producing Method
ContentResult	Returns a string literal	Content
EmptyResult	No response	
FileContentResult / FilePathResult / FileStreamResult	Return the contents of a file	File
HttpUnauthorizedResult	Returns an HTTP 403 status	
JavaScriptResult	Returns a script to execute	JavaScript
JsonResult	Returns data in JSON format	Json
RedirectResult	Redirects the client to a new URL.	Redirect
RedirectToRouteResult	Redirect to another action, or another controller's action	RedirectToRoute / RedirectToAction
ViewResult PartialViewResult	Response is the responsibility of a view engine	View / PartialView

ViewResult

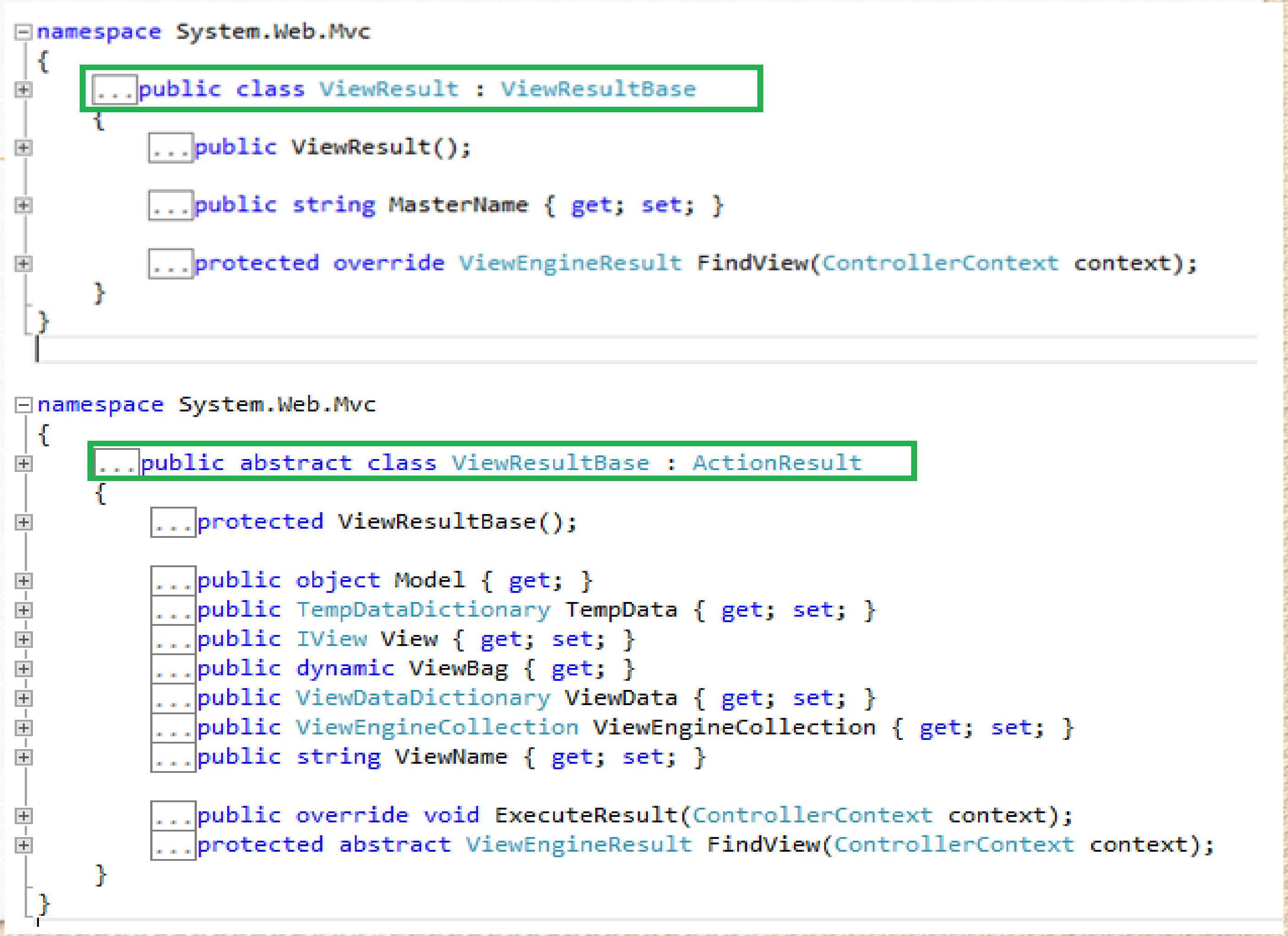
- View result is a basic view result.
- It returns basic results to view page.
- View result can return data to view page through which class is defined in the model. View page is a simple HTML page.



```
public ViewResult About()
        {
                ViewBag.Message = "Your application description page.";
                return View();

        }
```

ViewResult (Cont.)

- "View Result" is a class and is derived from "ViewResultBase" class.
- "ViewResultBase" is derived from "Action Result".
- "Action Result" is a base class of different action result.



```
namespace System.Web.Mvc
{
    ...public class ViewResult : ViewResultBase
    {
        ...public ViewResult();

        ...public string MasterName { get; set; }

        ...protected override ViewEngineResult FindView(ControllerContext context);
    }
}
```

```
namespace System.Web.Mvc
{
    ...public abstract class ViewResultBase : ActionResult
    {
        ...protected ViewResultBase();

        ...public object Model { get; }
        ...public TempDataDictionary TempData { get; set; }
        ...public IView View { get; set; }
        ...public dynamic ViewBag { get; }
        ...public ViewDataDictionary ViewData { get; set; }
        ...public ViewEngineCollection ViewEngineCollection { get; set; }
        ...public string ViewName { get; set; }

        ...public override void ExecuteResult(ControllerContext context);
        ...protected abstract ViewEngineResult FindView(ControllerContext context);
    }
}
```

Sending Data
From Controller To View

- **ViewData** and **ViewBag** are used for the same purpose “***to transfer data from controller to view***.”

- **ViewData** is a ***dictionary of objects*** and it is accessible by ***string as key***.
- ViewData is a ***property of controller*** that exposes an instance of the ViewDataDictionary class.

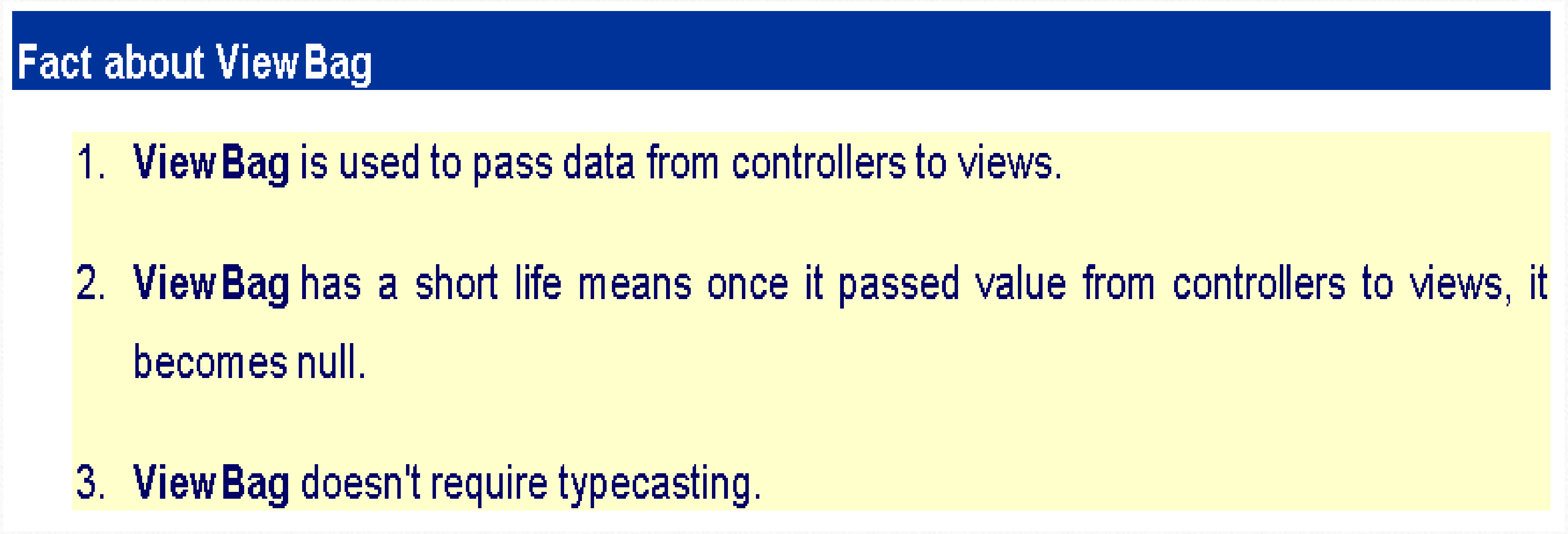
- **ViewBag** is very similar to ViewData.
- ViewBag is a ***dynamic*** property.
- ViewBag is able to set and get value dynamically and able to add any number of additional fields without converting it to strongly typed.
- ViewBag is just a ***wrapper around the ViewData***.

ViewData

Some fact about View Data

1. It is also used for sending information from controllers to views.
2. Once it sends information, it becomes null.
3. ViewData is a Dictionary Object that is derived from ViewDataDictionary.
4. ViewData uses Key-Value pair for storing and retrieving information.
5. It requires typecasting for complex data type.

ViewBag



Fact about ViewBag

1. **ViewBag** is used to pass data from controllers to views.
2. **ViewBag** has a short life means once it passed value from controllers to views, it becomes null.
3. **ViewBag** doesn't require typecasting.

Demo