C#
core

Code Focused Training

ASP.NET Core MVC

ASP.NET Web API 2

GROUND UP SERIES

Microsoft .NET MVC

DESIGN PATTERN

ASP.NET MVC

SEPERATION OF CONCERNS

Syed Awase Khirni

RESEARCHER | ENTREPRENEUR | TECHNOLOGY COACH

@sak008 | sak@sycliq.com/sak@territorialprescience.com | +91. 9035433124

Syed Awase earned his PhD from University of Zurich in GIS, supported by EU V Framework Scholarship from SPIRIT Project (www.geo-spirit.org). He currently provides consulting services through his startup www.territorialprescience.com and www.sycliq.com . He empowers the ecosystem by sharing his technical skills worldwide, since 2008.

Original Series

TERRITORIAL™ PRESCIENCE
provoke thought, invoke action

Terms of Use

- You shall not circulate these slides without written permission from Territorial Prescience Research I Pvt ltd.
- If you use any material, graphics or code or notes from these slides, you shall seek written permission from TPRI and acknowledge the author Dr. Syed Awase Khirni
- If you have not received this material, post-training session, you shall destroy it immediately and not use it for unauthorized usage of the material. If any of the material, that has been shared is further used for any unauthorized training by the recipient, he shall be liable to be prosecuted for the damages. Any supporting material that has been provided by the author, shall not be used directly or indirectly without permission.
- If this material, has been shared to any organization prior to the training and the organization does not award the contract to TPRI, it should not use the training material internally. If by any chance, the organization is using this training material without written permission, the organization is liable to pay for the damages to TPRI and is subjected to legal action, jurisdiction being Bangalore.
- Without unauthorized usage, TPRI has right to claim damages ranging from USD 50000 to USD 10,0000 dollars as damages.
- Any organization, which does not intend to go ahead with training or does not agree with the terms and conditions, should destroy the material from its network immediately. The burden of proof lies on the client, with whom this material has been shared.
- Recovery of the damages and legal fees will be born by the client organization/candidate/party, which has violated the terms and conditions.
- Only candidates who have attended the training session in person from Dr. Syed Awase Khirni, TPRI are entitled to hold this training material. They cannot further circulate it, or use it or morph it, or change it to provide trainings.
- TPRI reserves all the rights to this material and code plays and right to modify them as and when it deems fit.
- If you agree with the terms and conditions, please go ahead with using the training material. Else please close and destroy the slide and inform TPRI immediately

Slide Version Updates

Last Updated	Version	Release Date	Updated by	Code Plays Done @

Program Agenda

DAY 1	DAY 2	DAY 3	DAY 4
DAY 5	DAY 6	DAY 7	DAY 8

SECTION -O

INSTALLATION AND DEV ENVIRONMENT CHECK

Installing ASP.NET MVC

- We can have multiple versions of ASP.NET MVC applications running side by side in your server/computer.

- ASP.NET MVC 4
  - http://www.asp.net/mvc/mvc3 for version 3
    - Does not come with bootstrap view scaffolds
    - Uses modernizr views
  - http://www.asp.net/mvc/mvc4 for version 4
    - Comes with bootstrap view scaffolds
    - Uses razorview engine for rendering
    - Requires .Net Framework 4 and above.

- ASP .NET MVC 5
  - Comes preinstalled in visual studio 2015.
  - Visual studio 2015 support for c#6.0
  - Requires .Net Framework 4.5 and above

- ASP .NET MVC 6
  - Comes preinstalled in visual studio 2017
  - Visual studio 2017 support for c# 7.0

Alternative View Styles

Databases

- MSSQL Server
- MongoDB

Original Series

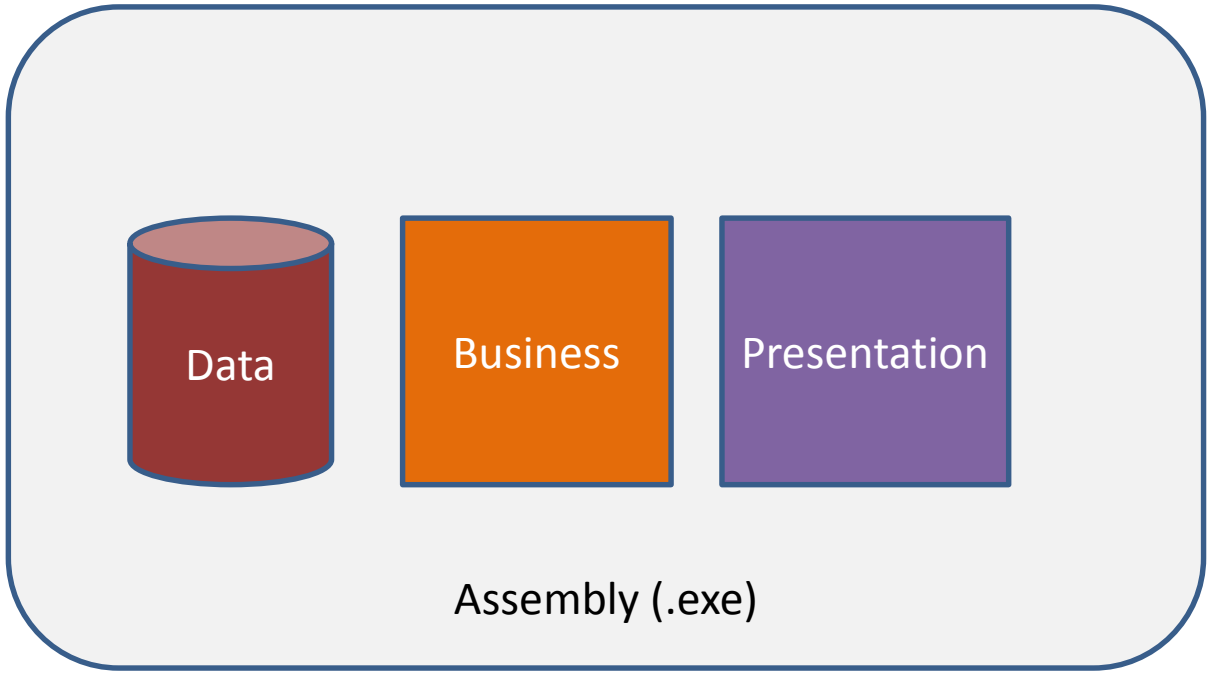
SECTION -1

ARCHITECTURES

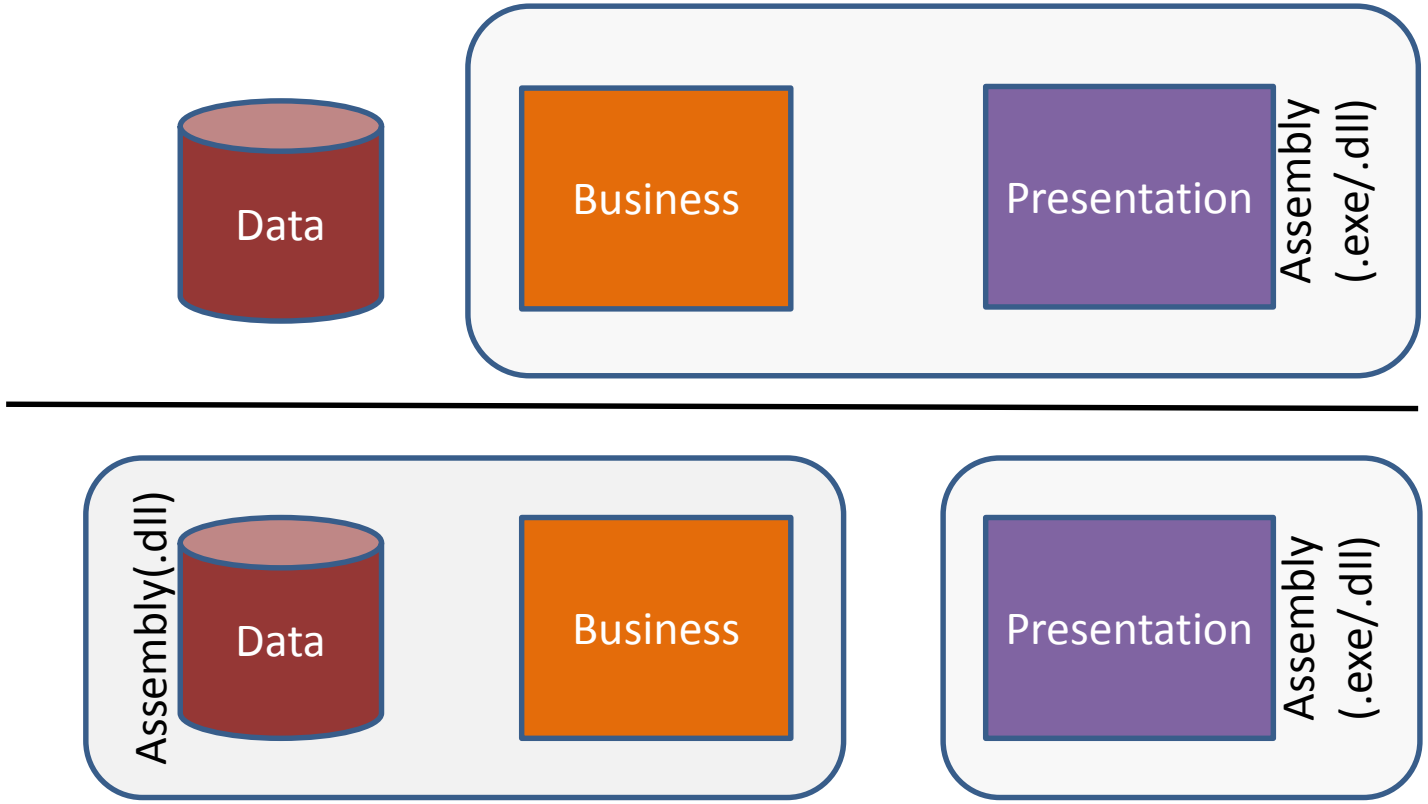
Architectures

- Three major concerns
  - Data / Storage (Files, Databases, External Storage)
  - Business Logic (Logic revolving adding/updating/deleting business objects)
  - Presentation
- One Tier
- Two Tier
- Three Tier
- n-Tier

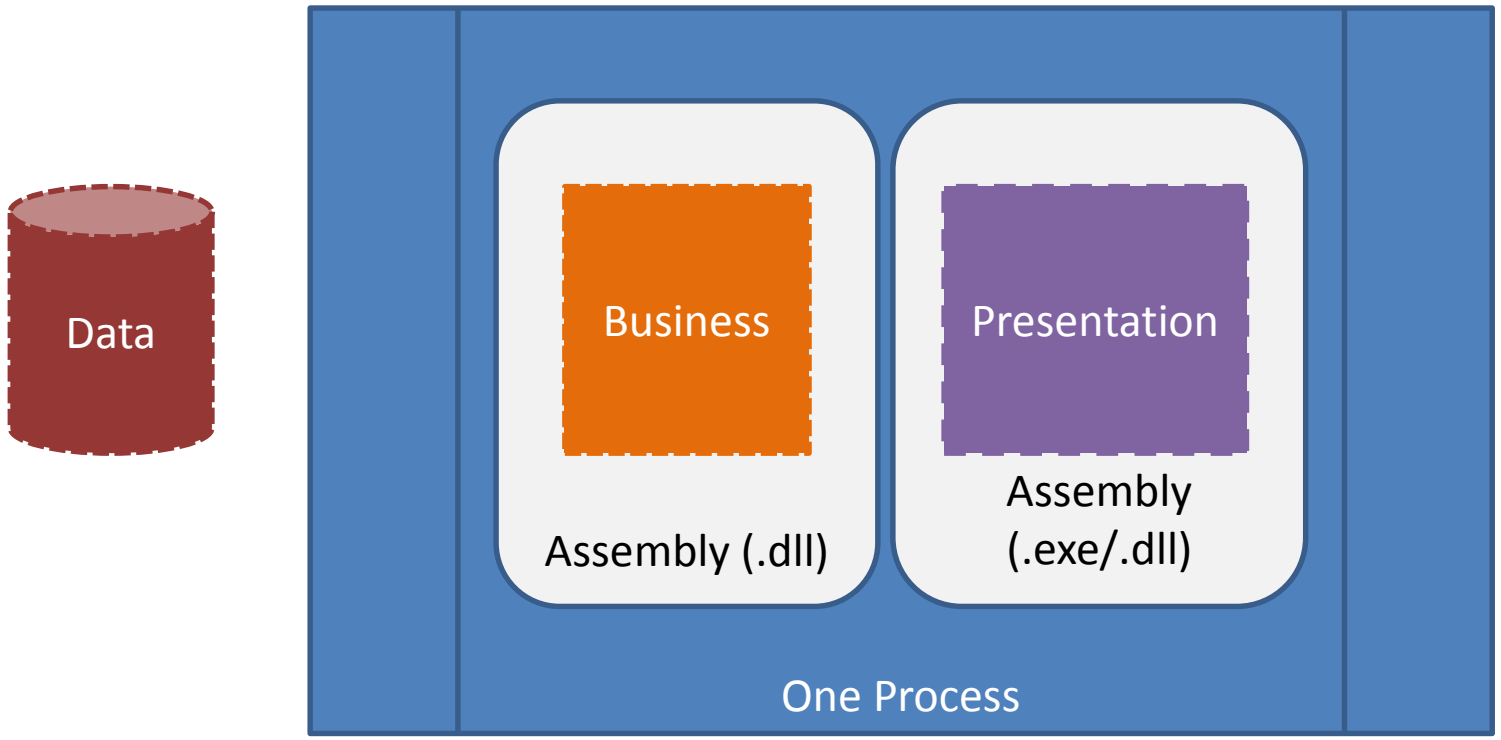
One Tier Architecture



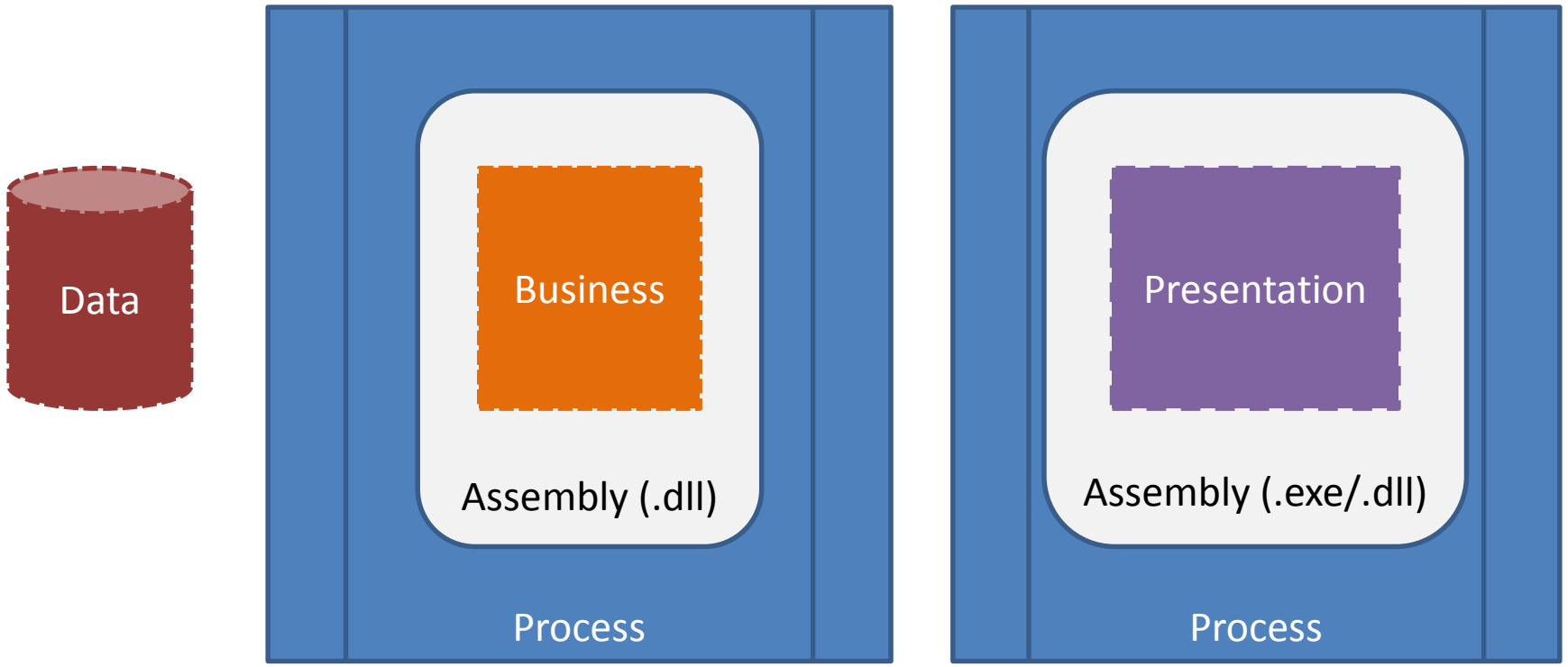
Two Tier Architecture



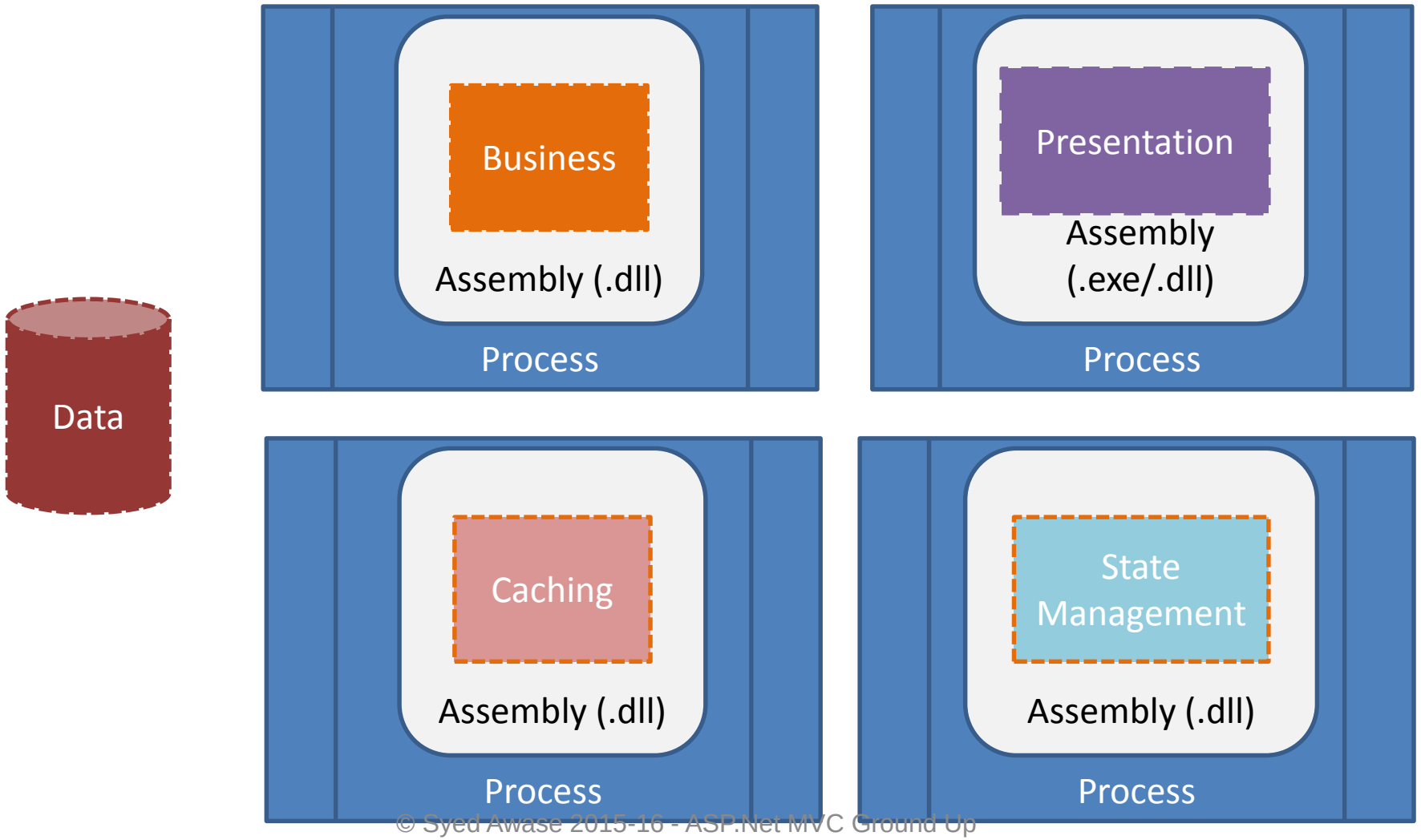
Three Tier Architecture (Logical)



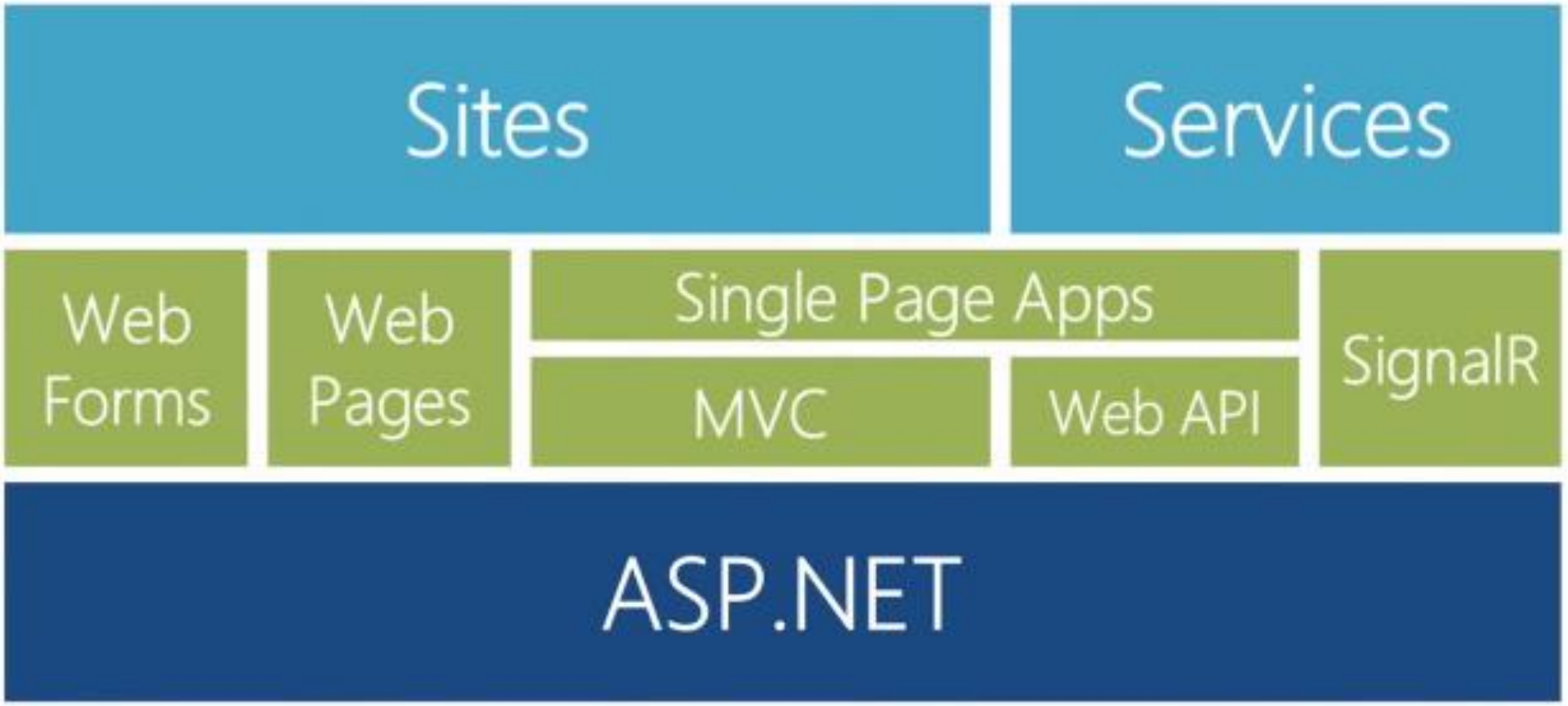
Three Tier Architecture (Physical)



n-Tier Architecture (Physical)

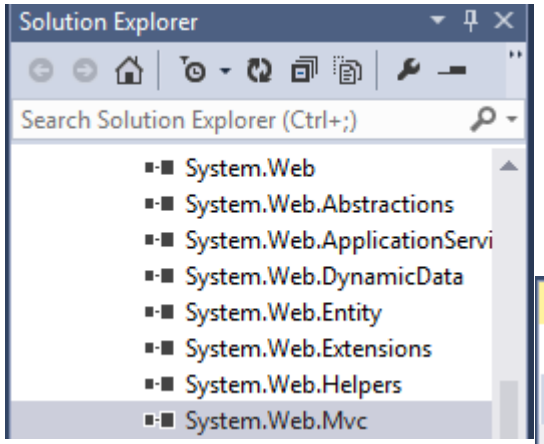


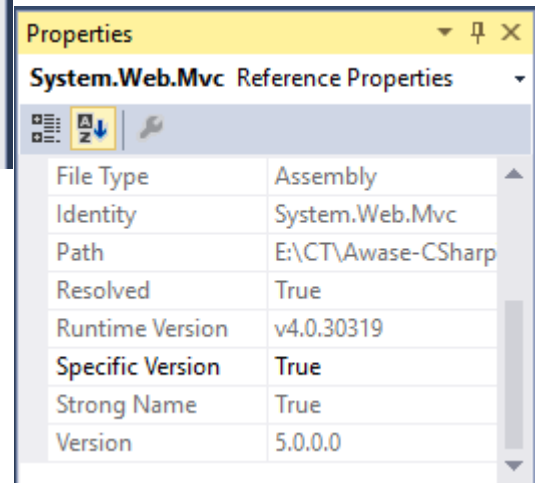
ASP.NET OVERVIEW



Installing ASP.NET MVC

- ASP.NET MVC
  - http://www.asp.net/mvc





What MVC version

- 2 ways to identify
  - At design time – Go to solution explorer -> expand "References" folder. Right click on "System.Web.MVC" assembly and select properties.
  - At run-time using the following code
    - **Typeof(Controller).Assembly.GetName().Version.ToString();**

Revisiting ASP.NET Web Forms

- First released in ASP .NET 1.0
- Replaced classic ASP (Active Server Pages)
  - Strongly typed code replace script
  - Abstract away the web
  - Click events replaced "POST" operations
- Original design from the late 90s
  - Web standards have strengthened
  - Client-side programming on the rise
- Web forms compete against other MVC frameworks
  - STURTS
  - RUBY ON RAILS (ROR)
  - DJANGO - PYTHON
  - ANGULARJS (VERY RECENTLY)

- Productive way to build web applications
- Control and event-based programming model
- Controls that abstract HTML, JS and CSS
- Rich UI Controls- datagrid, charts, AJAX
- Browser differences are handled for you

What's wrong with ASP.NET WEB FORM

- ViewState
- Page Life Cycle
- Limited Control over the rendered HTML
- Lack of **Separation of Concerns (SoC)**
- Untestable
- In a webform URL's are mapped to the **Physical Files**
  - **http://localhost/WebformsExampleOne/Default.aspx**

- Web Forms Supports rich server side controls and drag and drop option for designing View pages.
- Uses Event driven programming language.
- Easy to get started and less learning effort required.
- Logic is embedded in code behind files and page is heavy.
- Rapid Application Development and Support for ViewState.

DISADVANTAGES ASP.NET WEB FORMS

- No Application Structure guidance
- No standards on separation of concerns.
- Very complex page life cycle
- Unit testing and end to end testing becomes very difficult to manage.
- View states makes pages heavy and may reduce performance.
- Less support for parallel development.
- Tightly coupled with the view and code behind

When to USE ASP.NET MVC

- ASP.NET MVC is **NOT** a replacement of ASP.NET web forms based applications
- The approach of application development must be decided based on the application requirements and features provided by ASP.NET MVC to suite them

- Application development with ASP.NET MVC is more complex as compared to web forms based applications as they lack readily available rich controls and less knowledge of the pattern in ASP.NET Web Developers
- Application maintainability will be higher with separation of application tasks

MVC DESIGN PATTERN

- First proposed in 1970's
- Gives us a clean interaction model for web based development.
- View does not use Controller to update Model. Controller handles the events from View to manage user's interaction and data (via interaction with Model)
- Controller can be combined with View. Logical Separation of Model from the View
- The Controller does not contain the rendering logic.
- Controller uses view asking it to render new data.

SECTION -XIII

MVC 2.0

MVC 2.0

- UI Helpers with Scaffolding templates
- Model validation on Client and Server Side
- Strongly typed HTML helpers
- Enhanced Visual Studio tooling.

MVC 2.0

- Customer side validation
- Template helpers
- Regions
- Non-concurrent controllers
- Html.ValidationSummary Helper Method
- Restricting Binary data with Model Binders
- DataAnnotations Attributes
- Validator Providers
- New RequireHttpsAttribute Action Filter
- Templated Helpers
- Model-level Error diagnostics
- Displaying Model Level Error.

SECTION -XIII

MVC 3.0

MVC 3.0

- Released in 2011.
- Inclusion of Razor View Engine
- .NET 4 Data Annotations
- Robust model binding and validations
- Inclusion of Global Action Filters
- Jquery Support and JavaScript Validations
- JSON Binding
- NuGet Integration to resolve the software dependency on the fly.

MVC 3.0

- Razor view engine introduced
- HTML5 format support
- Support for Multiple View Engines
- JavaScript and Ajax Support
- Validation Enhancements.
- Templates for HTML5 and CSS3
- Improved model Validation
- Improved Controller
- Improved Dependency Injection using Idependency Resolver interface
- Partial Page output caching.
- Introduced a new feature called Bundling of resources.
- ViewBag dynamic property introduced.
- ActionResult Types feature introduced.

SECTION -XIII

MVC 4.0

MVC 4.0

- ASP.NET Web API
- Improved Project Templates, Added new ones
- Inclusion of Mobile Projects using Jquery Mobile
- Various Display Modes
- Asynchronous Controllers
- Bundling and Minification.

MVC 4.0

- ASP.NET WEB API 1.x
- Improved support for rendering various venture formats.
- Structured Application development best practices.
- Modernizer, JavaScript and Ajax Support. Look and feel improvements.
- Empty Project Template.
- Mobile Project Template
- Support for adding Controller
- Task Support for Async Controller
- Bundling and Minification Support
- Support for Oauth and OpenID.
- Support for Windows Azure SDK1.6

SECTION -XIV

MVC 5.0

MVC 5.0

- Improved Scaffolding
- ASP.NET Identity Management
- One ASP.NET
- Support for Bootstrap
- Attribute Routing
- Filter Overrides

MVC 5.0

- ASP.NET WEB API 2.x
- ASP.NET IDENTITY MANAGEMENT SUPPORT AND SOCIAL LOGIN SUPPORT.
- BOOTSTRAP 3.x Support
- CONFIRMATION FILTERS
- CHANNEL SUPERSEDES
- IMPROVED ROUTING MECHANISM
  - Attribute based routing.
- Authentication Filters and Filter Overrides.
- Filter overrides
- Internet Application Template and Intranet Application Template
- ASP.NET WEB API Template
- Mobile Project Template

SECTION -XIV

MVC 6.0

MVC 6.0

- Common framework for MVC, Web API and Web Pages
- Smooth Transiting from Web Pages to MVC
- Built DI First
- Runs on IIS or self host
- Based on the new Request Pipeline in ASP.NET vNext
- Runs cloud optimized
- No build dependency
- Enhanced developer experience
- Open source
- Cross-platform support.

MVC 6.0

- ASP.NET MVC AND WEB API converged into one.
- Reliance infusion is inbuilt and part of MVC.
- Extended support for bundling and minification using NuGet, including the .NET runtime
- New JSON based venture structure
- New cloud computing optimization system of MVC, WebAPI, SignalR and Entity Framework
- IMPROVED ROUTING PERFORMANCE.
- IMPROVED COMPILATIONS AND HOT RELOADING
- NEW ROSYLN ONGOING COMPILER
- vNext is Opensource supported by .NET Foundation
- vNext and Rosyln support for Mono to run on Mac and Linux
- Removed the dependency of System.web.dll from MVC6.
- Added a Start-up class that replaces the global.asax file.

MVC

- A framework for building
  - Scalable
  - Flexible
  - Extensible
  - Standard based web application
- Complex Applications can be easily managed.
- ASP.NET MVC is light weight as they do not maintain view state in client page.
- MVC works on Conventions over Configurations.
- MVC uses the power of ASP.net and .NET framework
- MVC uses separation of Concerns
- MVC framework is defined in "System.Web.MVC.Assembly" namespace in .NET/ASP.NET MVC
- Support for HTML5 enabled Razor view Engine.

MVC

- Search Engine Optimization (SEO) – Clean URL's and no extension methods used for locating the files.
- Rich Javascript support with unobstrusive javascript validation, Jquery validation and JSON binding.
- Requirements should be identified much in depth at design phase.

MVC flow

BROWSER

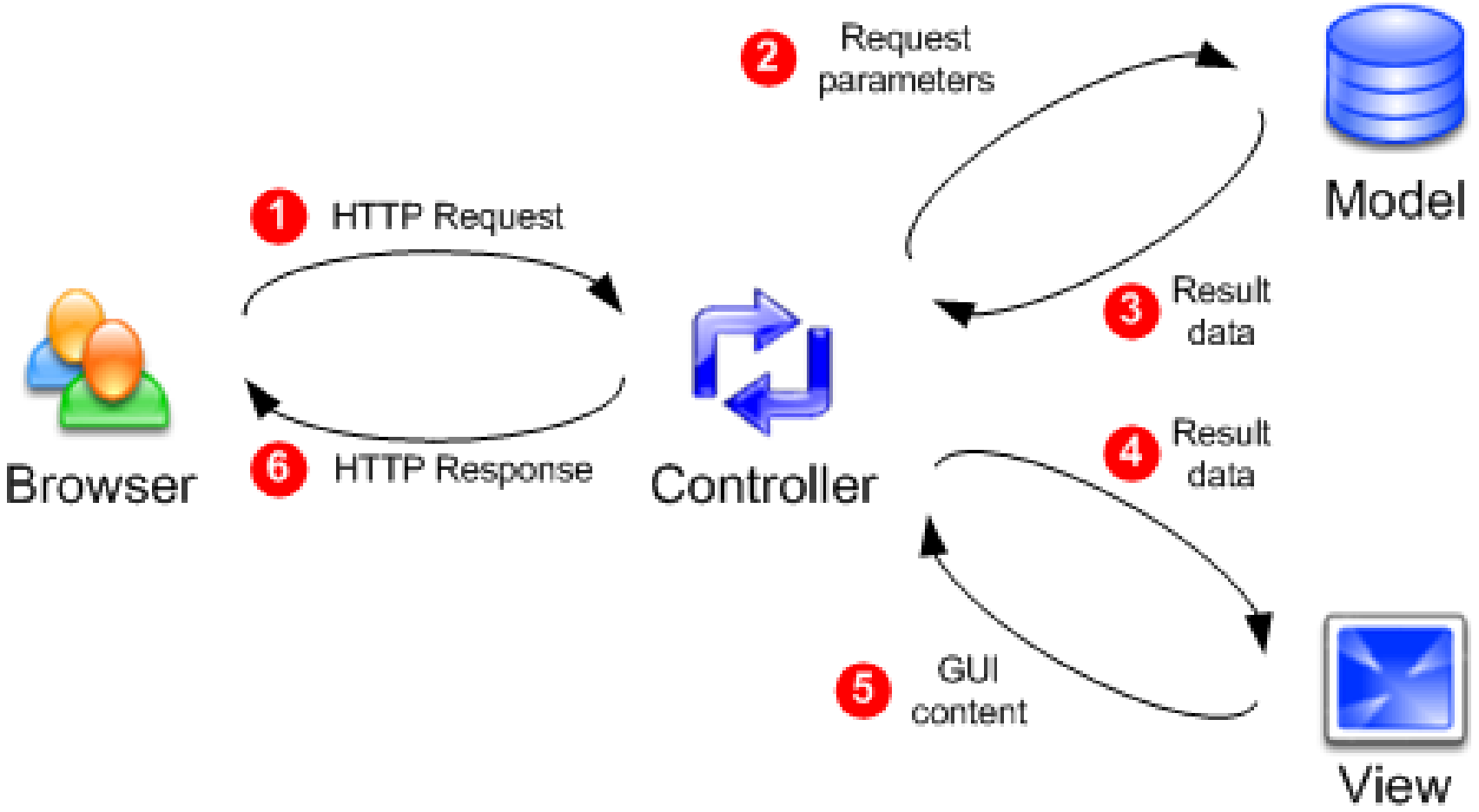
CONTROLLER

VIEW

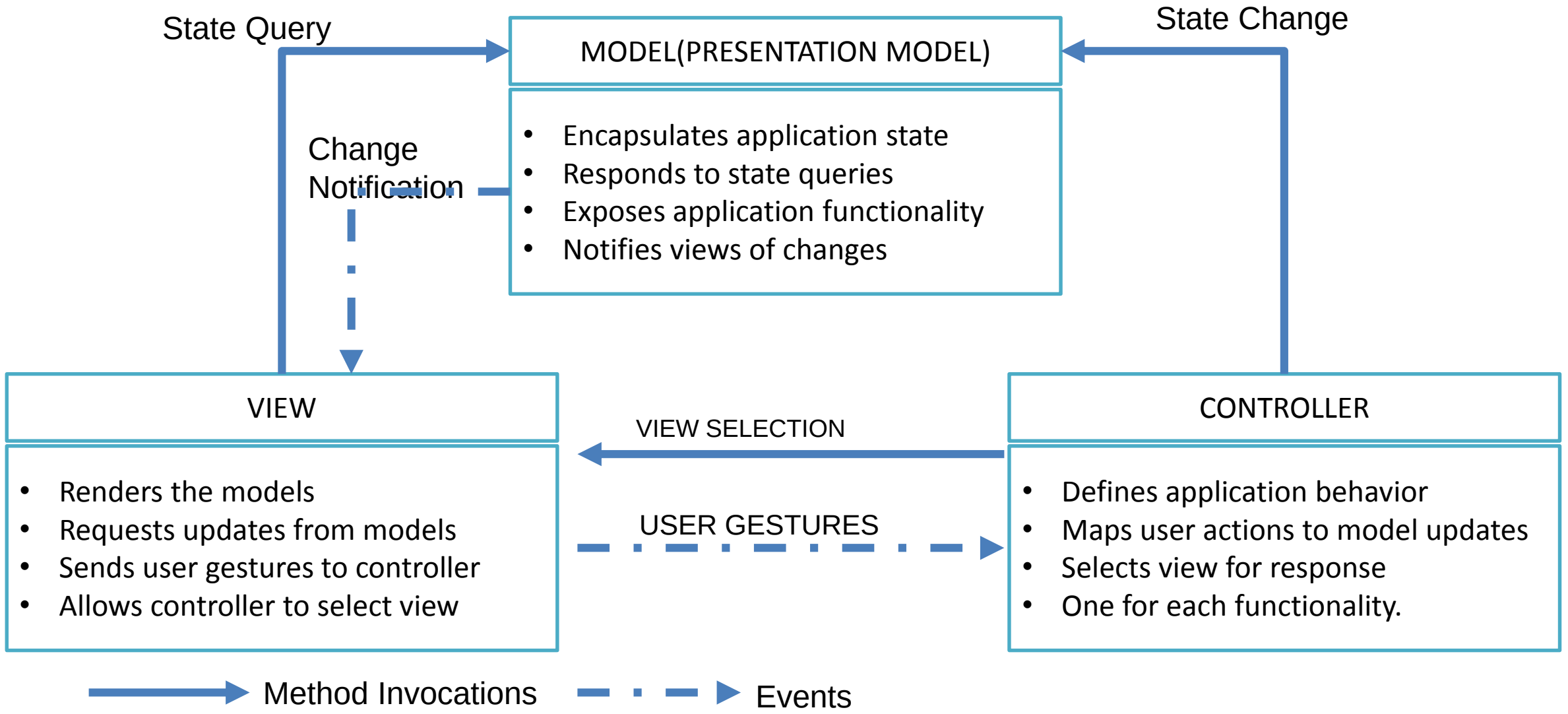
MODEL

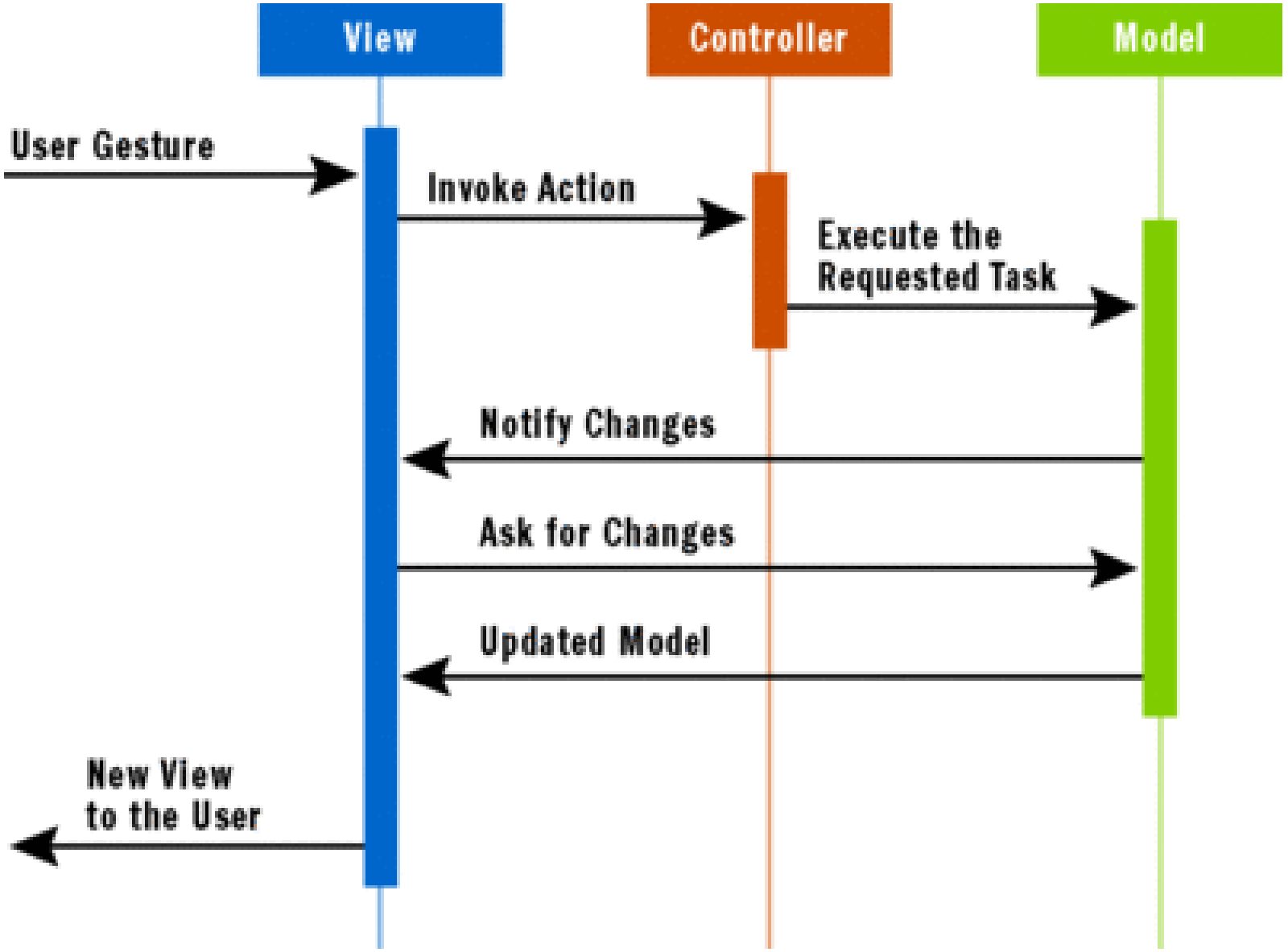
DB

MVC IN ACTION



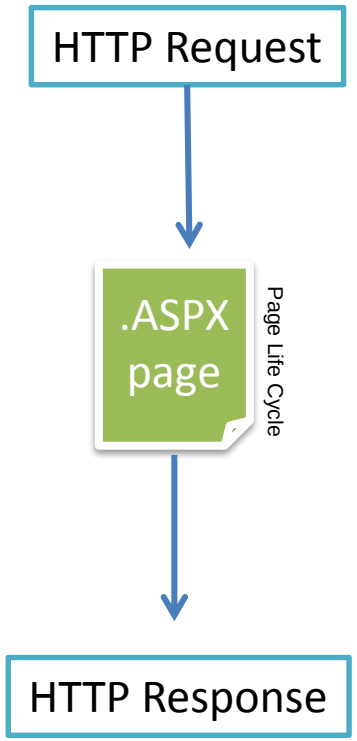
MODEL CONTROLLER VIEW





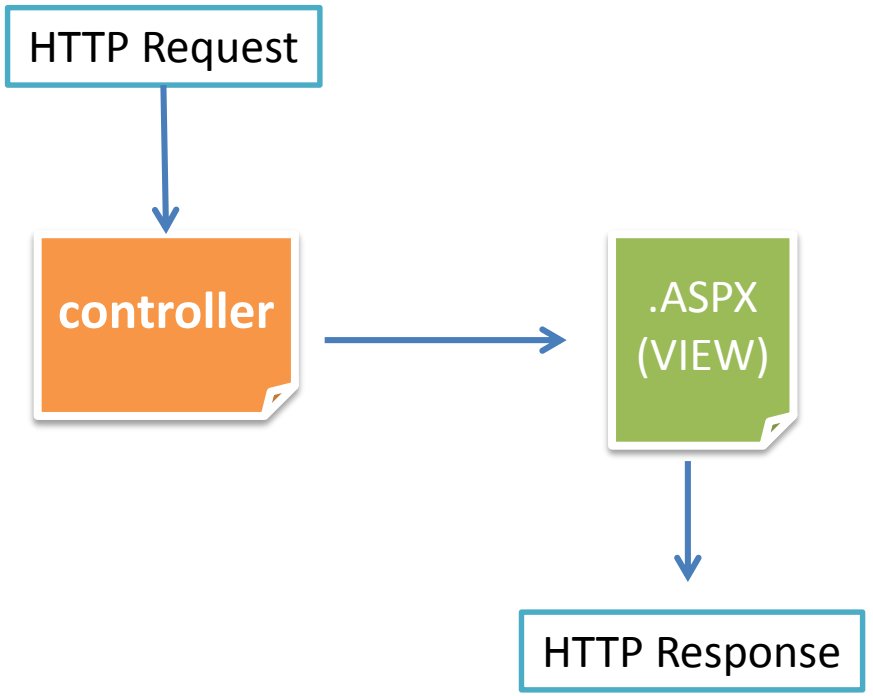
WEB FORMS vs MVC

ASP.NET WEB FORMS



url's are mapped to Physical files

ASP.NET MVC

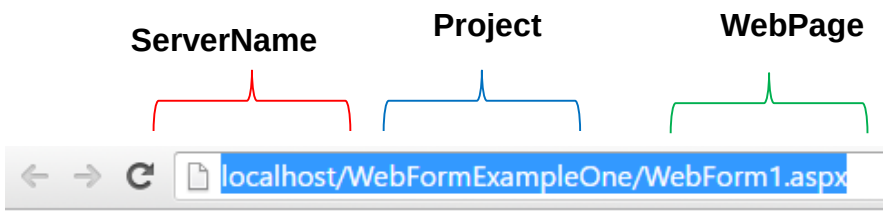


url's are mapped to controller action methods

WEB FORMS vs MVC

Web Forms

- In a **webForms URL's** are mapped to **Physical Files**

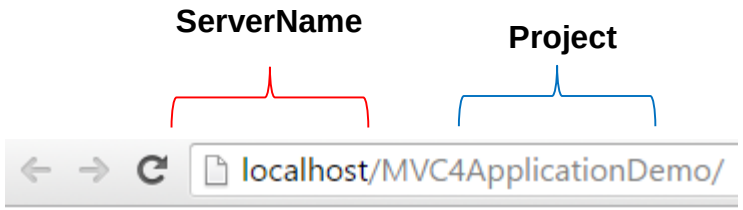


Hello from Webforms application

```
protected void Page_Load(object sender, EventArgs e)
{

    Response.Write("Hello from Webforms application");

}
```

MVC

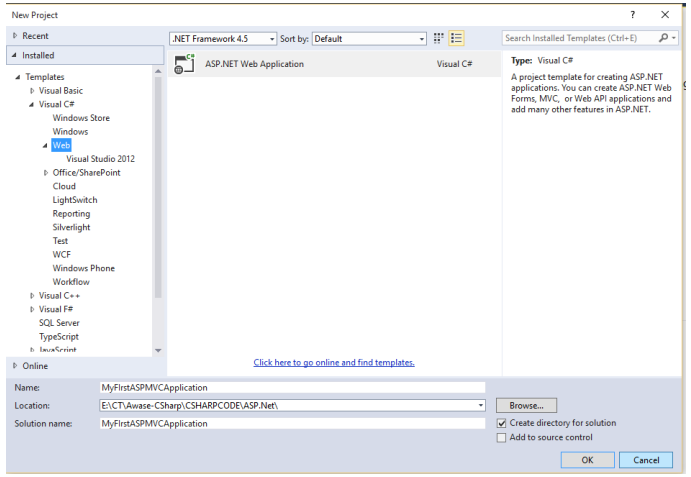
- MVC URL's are mapped to **controller Action Methods**
- Functions in a controller are generally called as **Controller Action Methods**

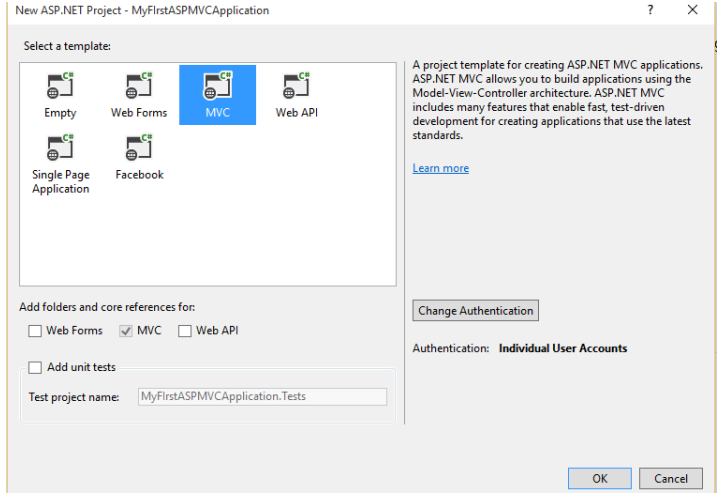


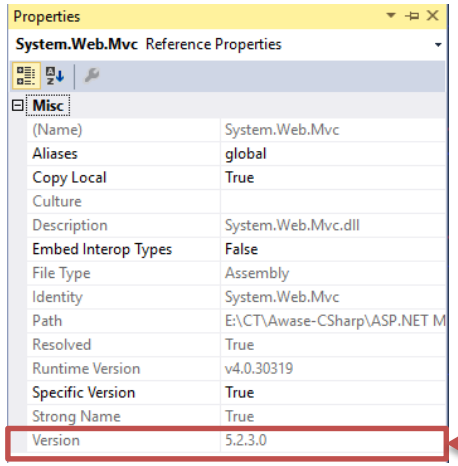
Hello from MVC4 Application using razor Engine

```
0 references
public String Index()
{
    //ViewBag.Message = "Modify this template to jump-start your ASP.NET MVC
      application.";

    //return View();
    return "Hello from MVC4 Application using razor Engine";
}
```

MVC Application with VS







@design time

Identifying MVC Version @ runtime using reflection

```
public ActionResult Index()
{
    //1. Getting the version of the MVC libraries at runtime using reflection.
    //2. Using ViewBag.Dictionary
    //3. Basic Controllers defined in the scaffolding template provided by Visual
      Studio
    ViewBag.MVCVersionQuery = typeof(Controller).Assembly.GetName
      ().Version.ToString();
    //4.Conventional view called by the Controller action method - Index()
      (without any parameters passed).
    //5.Using ViewData to pass on MVC Version Property
    ViewData["MVCversion"] = typeof(Controller).Assembly.GetName
      ().Version.ToString();
    //6. using model to pass on MVC Version Property
```

ASP.NET MVC DESIGN GOALS

- Does not replace web forms
  - An alternative project type
- Still runs on ASP.NET
  - Caching
  - Modules
  - Master pages
  - Providers
  - Handlers
  - Session state
- Embrace the web
  - No illusion of state – no page life cycle
  - Clean URLS and clean HTML
- Extensible
  - Pluggable view engines
  - Controller factories
- Testable
  - Maintains a strict separation of concerns

Features of MVC

- Separation of application tasks viz.business logic, UI logic and input logic
- Supports Test Driven Development
- Highly Testable framework
- Powerful URL-mapping component for comprehensible and searchable URLs
- Support existing ASP.net features viz.authentication, authorization, memberships and roles, caching, state management, configuration, health monitoring etc..

Model

- It represents the application domain.
- Applications business logic can also be contained in model.
- Model contains pieces of C# classes with set of entity properties and data annotation validations defined on top of it and data access logic.
- Model can be entities or business objects.

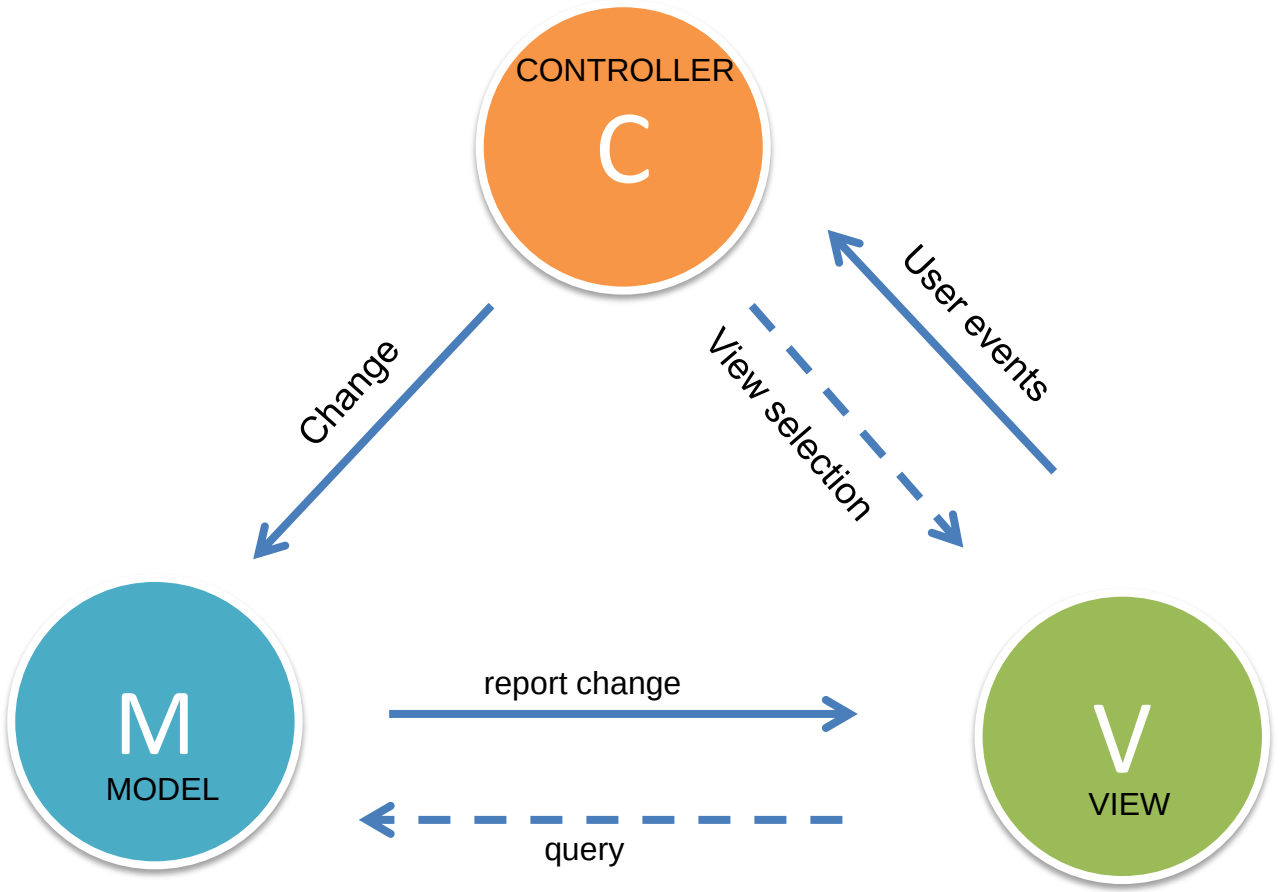
Controller

- Controller contains the control flow logic
- Controller handles the user interaction with the web application.
- Controller contains a set of action methods.
- User requests comes through controller to model and manipulate the records from it and then render the required data using view to UI.

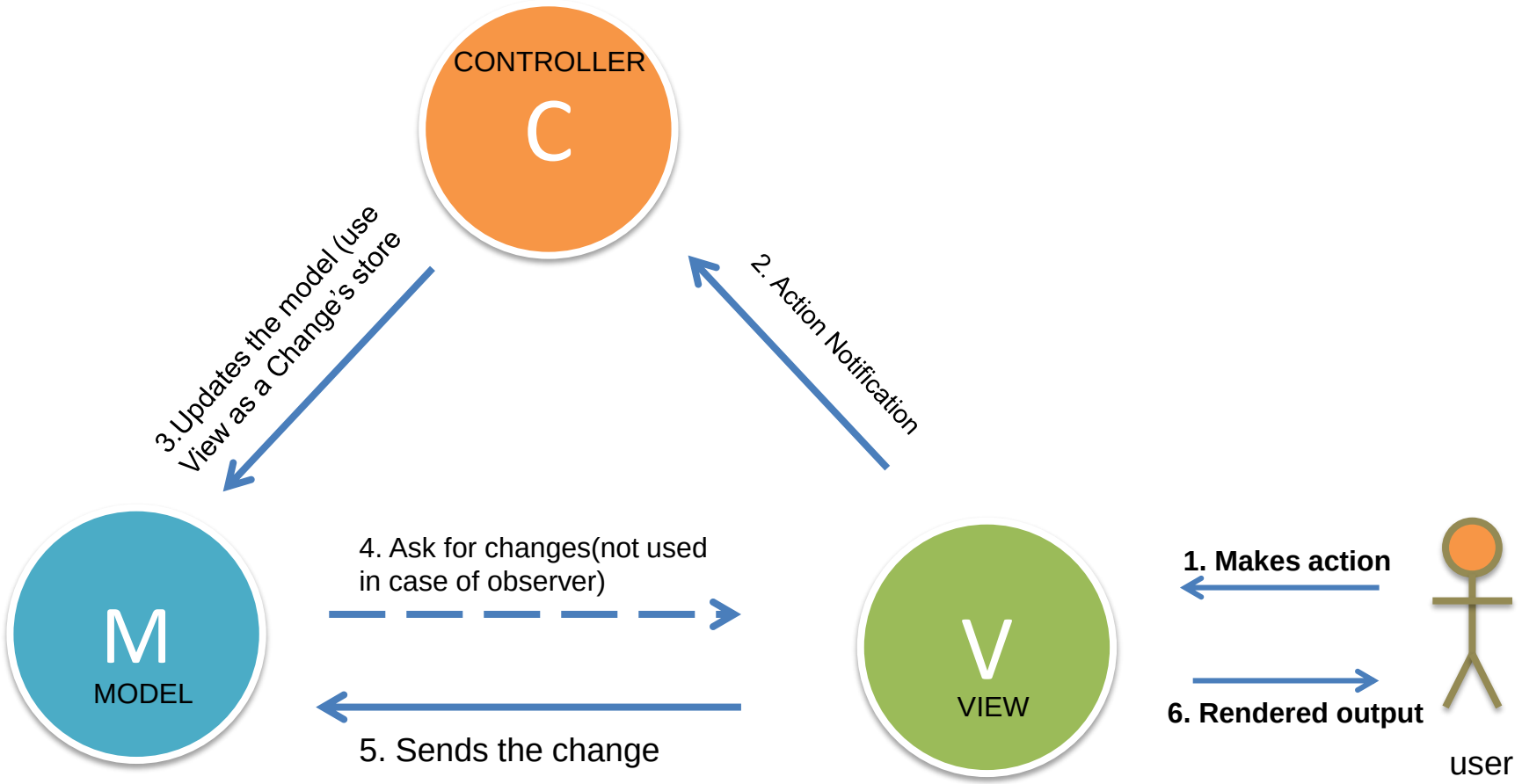
Views

- Views represent the presentation layer of the web application
- User interface (UI) logic will be contained in the view.
- Views should be dumb(shouldn't contain any application logic, only display logic.
- No code behind and No business logic.
- Action methods by default call view with the same name (Conventions over Configurations)
- Views are not tied to specific controller or action method.

MVC DESIGN PATTERN



MVC DESIGN PATTERN

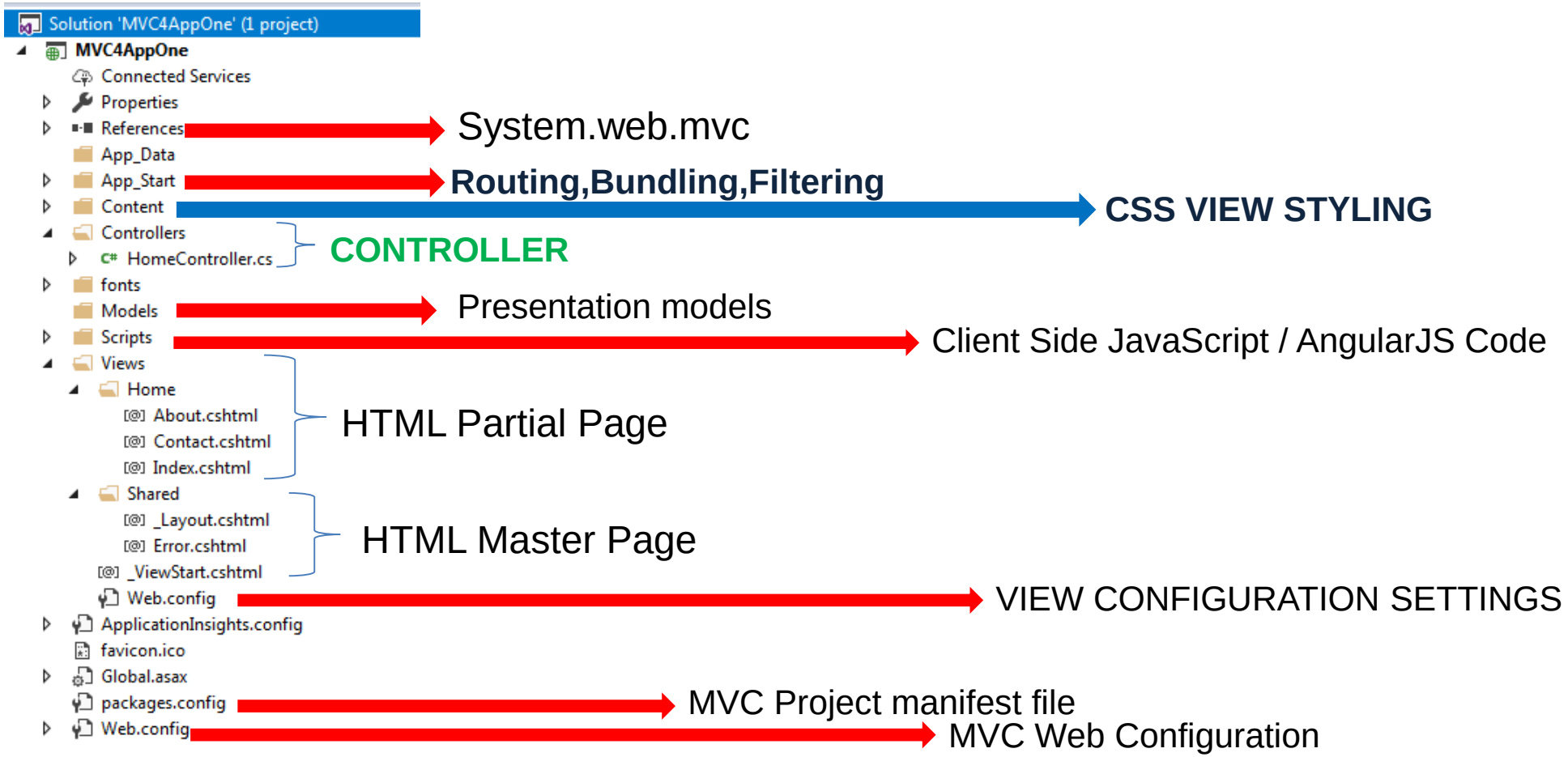


MVC DESIGN PATTERN

- Controller initializes the events of View interface to interact with model and controller.
- The user interacts with the View (UI)
- Controller handles user's events (can be the "observer" pattern) and asks Model to update.
- Model raises events, informing subscribers (View) about changes
- View (UI) (subscribes to model events) handles Model's events and shows new Model's data.
- The UI user interface waits for further user actions

MVC Life Cycle

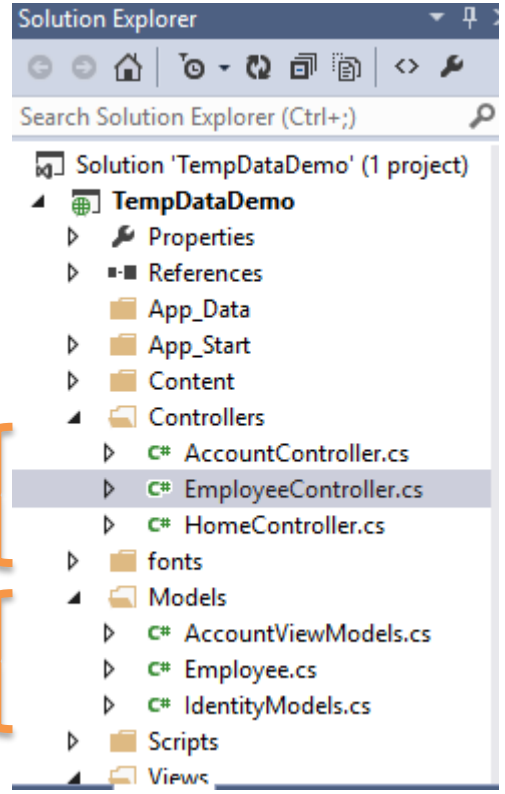
MVC Application Structure



MODELS in ASP.NET MVC


```
2 references
public class Employee
{
    1 reference
    public int EmployeeId { get; set; }
    1 reference
    public string Name { get; set; }
    1 reference
    public string Gender { get; set; }
    1 reference
    public string city { get; set; }
}
```

```
0 references
public class EmployeeController : Controller
{
    //
    // GET: /Employee/
    0 references
    public ActionResult Index()
    {
        Employee employee = new Employee
        {
            EmployeeId = 101,
            Name = "John",
            Gender = "male",
            city = "London"
        };
        return View(employee);
    }
}
```

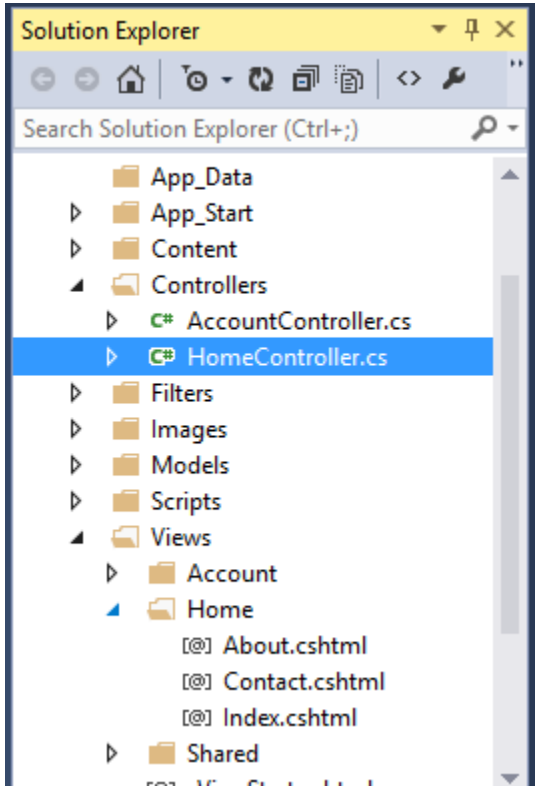


Controllers in an MVC Application

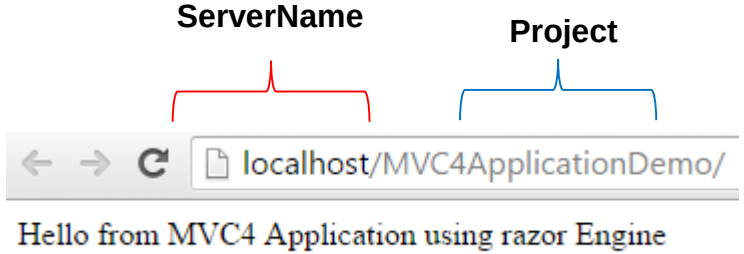
Controller action method

```
namespace MVC4ApplicationDemoTwo.Controllers
{
    0 references
    public class HomeController : Controller
    {
        0 references
        public ActionResult Index()
        {
            ViewBag.Message = "Modify this template to jump-start your ASP.NET MVC
              application.";

            return View();
        }
```

Controller

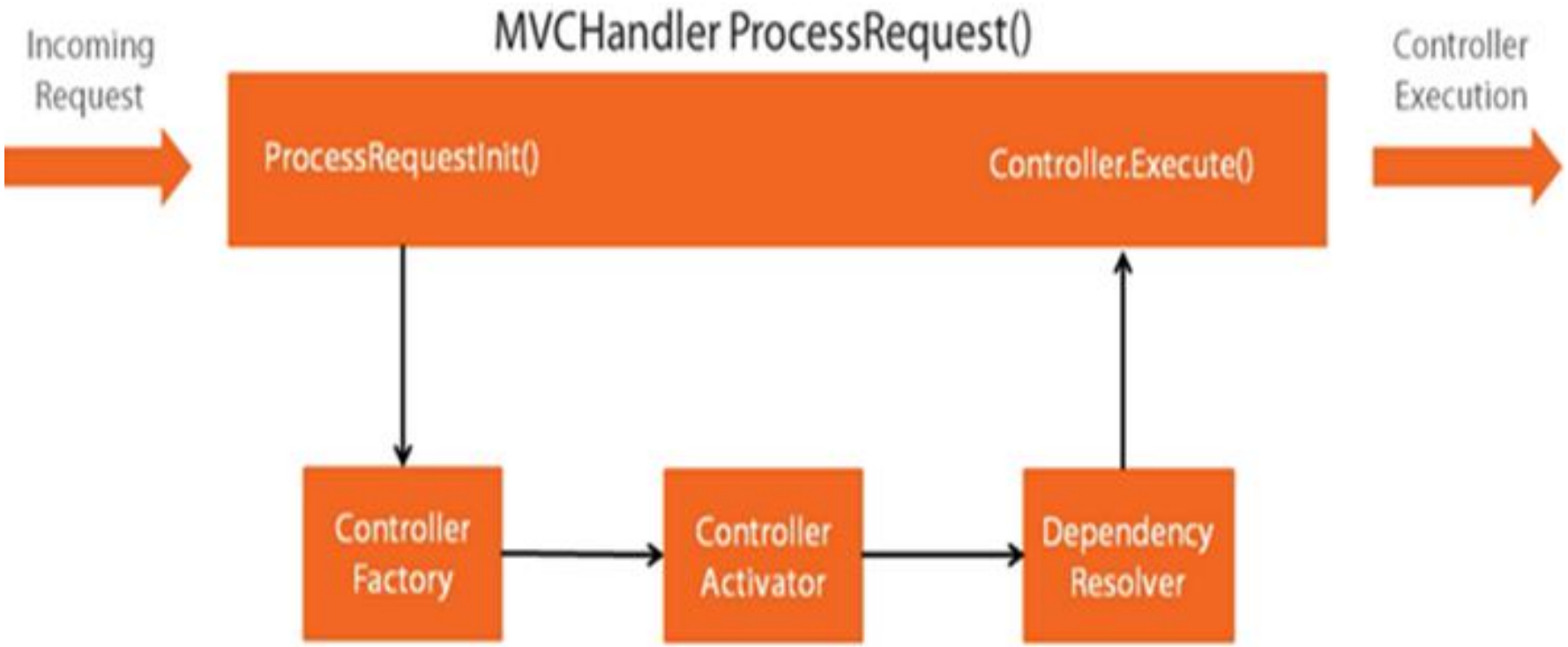


Views



How are the URL's mapped to Controller action Method? (ActionResult Index())?

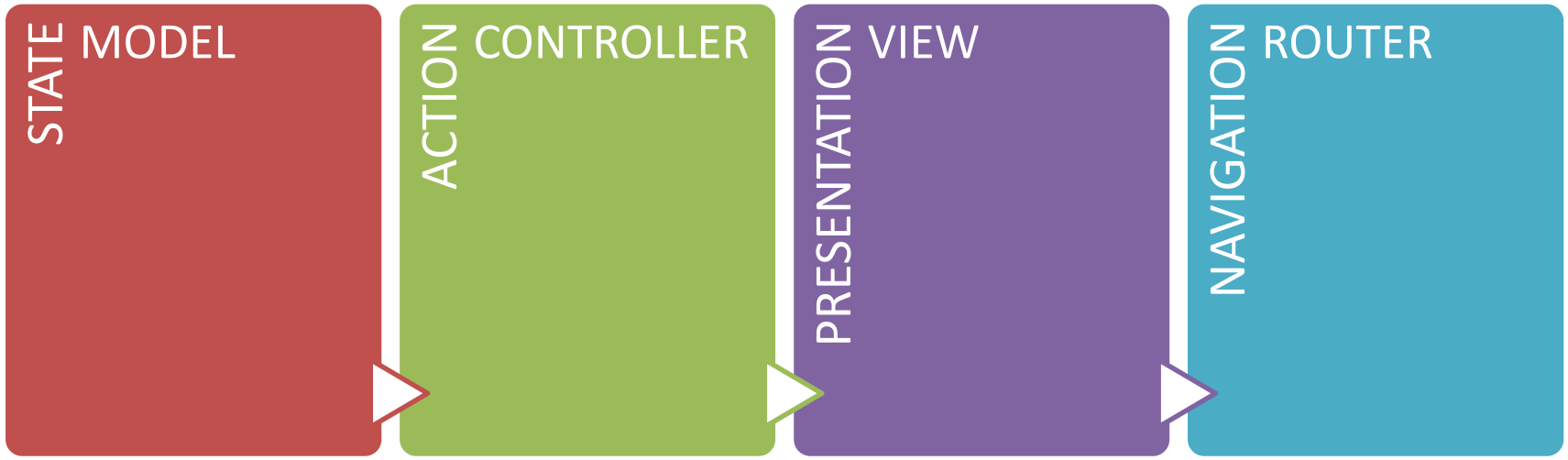
Controller Initialization



MVC Controllers

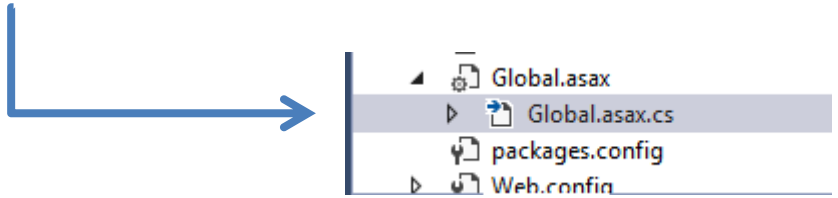
- Public methods are "**actions**"
  - Method invoked by ASP.NET once it determines the proper route
  - Controller can build the model and place in ViewData
  - Return value of ActionResult tells the framework where to go next

```
0 references
public class HomeController : Controller
{
    1 reference
    public ActionResult Index()
    {
        return View();
    }

    1 reference
    public ActionResult About()
    {
        ViewBag.Message = "Your application description page.";

        return View();
    }

    1 reference
    public ActionResult Contact()
    {
        ViewBag.Message = "Your contact page.";

        return View();
    }
}
```



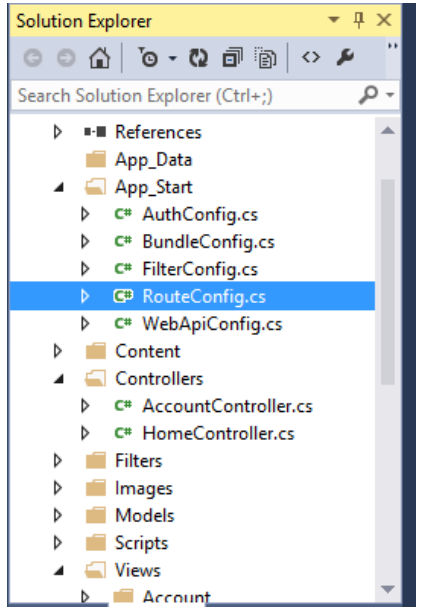
MVC Routing

- System.Web.Routing
  - Part of ASP.NET and released with .NET 3.5 SP1
- Directs incoming request to and MVC controller
  - Defines routes during application startup
  - Map URLs to controller action with parameters



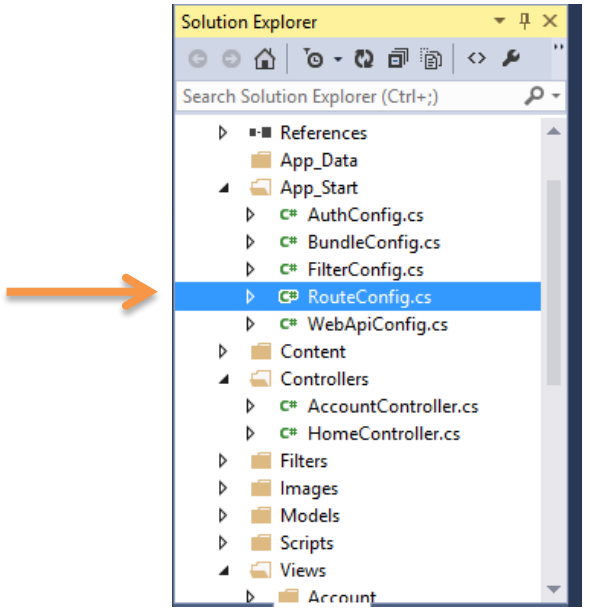

```
0 references
protected void Application_Start()
{
    AreaRegistration.RegisterAllAreas();

    WebApiConfig.Register(GlobalConfiguration.Configuration);
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
    BundleConfig.RegisterBundles(BundleTable.Bundles);
    AuthConfig.RegisterAuth();
}
```

```
routes.MapRoute(
        "Default",                          // Route name
        "{controller}/{action}/{id},        //URL with parameters
        new {
                controller = "Home",
                action = "Index",
                id = UrlParameter.Optional
        } ;             //Parameter defaults
```



MVC Routing

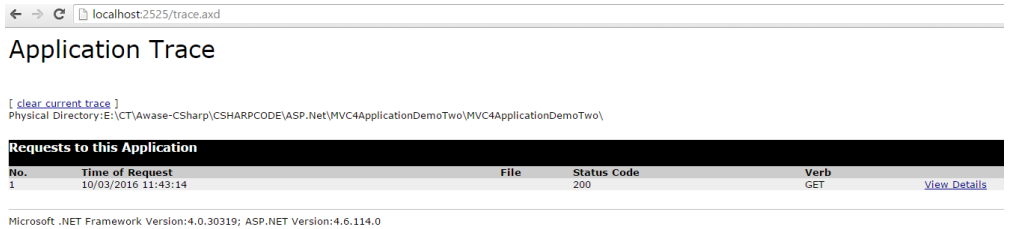

```
1 reference
public class RouteConfig
{
    1 reference
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id =
              UrlParameter.Optional }
        );
    }
```

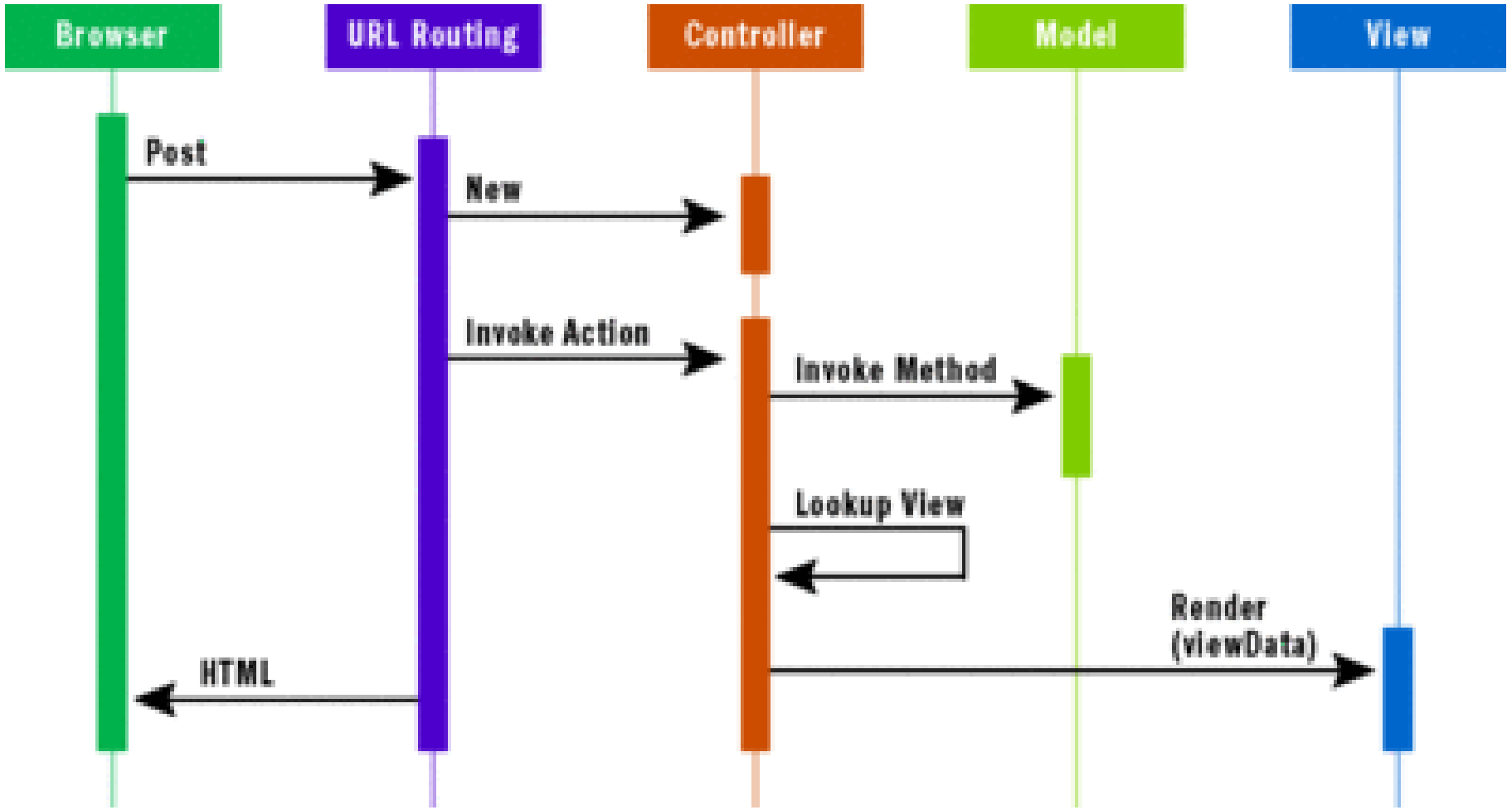


Enable trace in web.config

```
<system.web>
  <compilation debug="true" targetFramework="4.0" />
    <trace enabled="true" pageOutput="false"/>
```

http://localhost:2525/trace.axd



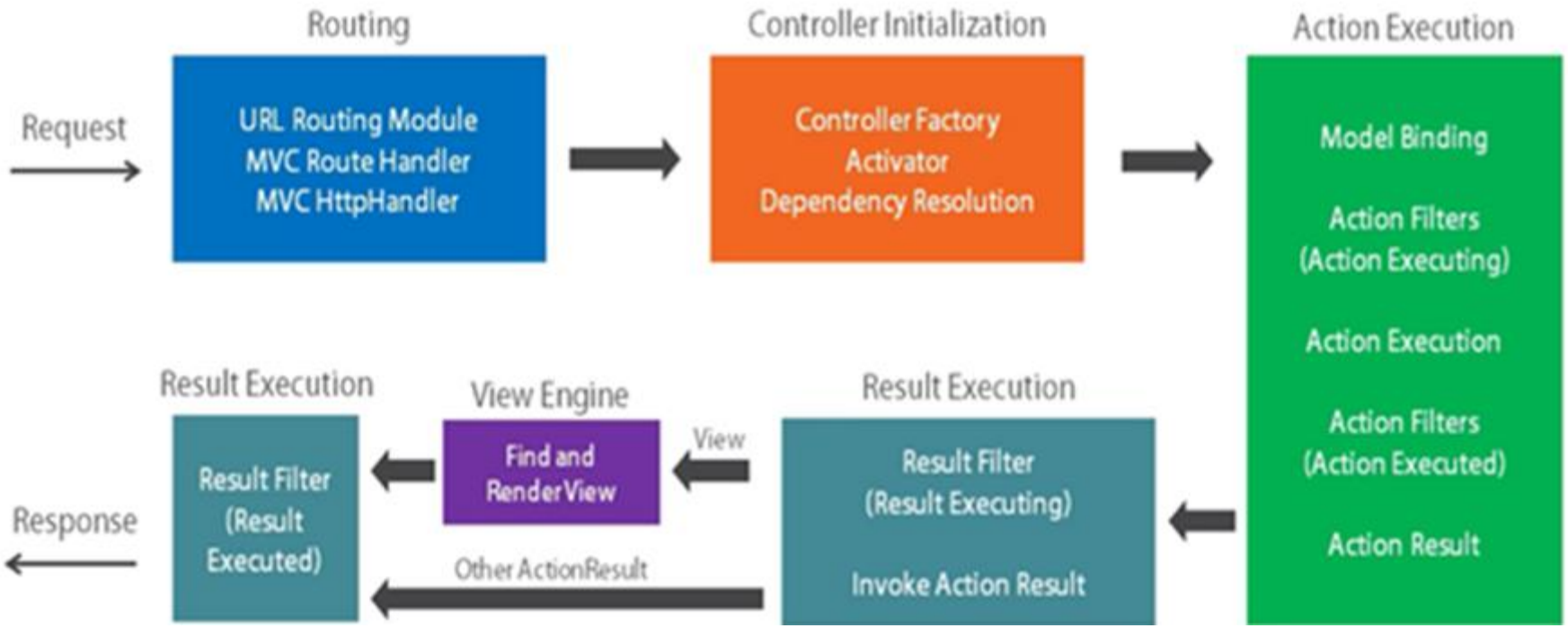


Original Series

Stages of Request Execution

Stage	Description
httpRequest to the Application	**Global.asax** file, Route Objects are added to the **Route Table** Object
Perform routing	**UrlRoutingModule** uses the first matching Route Object in the **RouteTable** collection to create the **RouteData** Object, which it then uses to create a **RequestContext** Object
Create MVC request handler	**MvcRouteHandler** object creates an instance of the MvcHandler class and passes the RequestContext instance to the handler
Create Controller	**MvcRouteHandler** object uses the RequestContext instance to identify the IControllerFactory object (typically an instance of the DefaultControllerFactory class) to create the controller instance with.
Execute Controller	MvcHandler instance calls the controller's EXECUTE method
Invoke Action	For controllers that inherit from the ControllerBase class. The ControllerActionInvoker object that is associated with the controller determines which action method of the controller calss to call and then calls that method
Execute Result	Action method receives user input, prepares appropriate response data and then executes the result by returning a result type. The built-in result type that can be executed include the following: ViewResult(which renders a view and is most often used result type). RedirectToRouteResult, RedirectResult, ContentResult, JsonReseult, FileResult and EmptyResult.

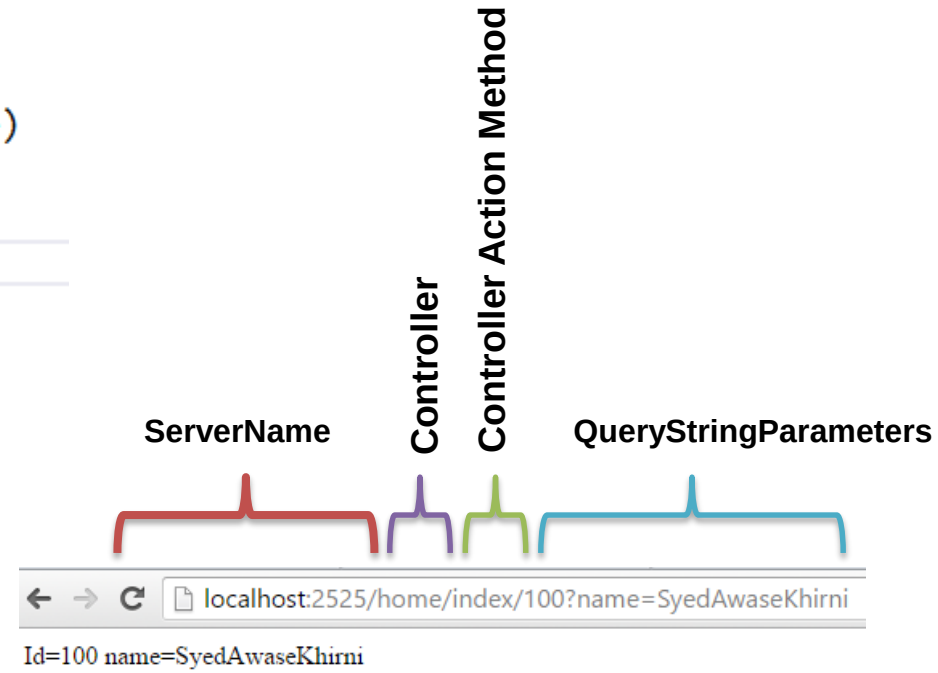
The MVC Request Life Cycle



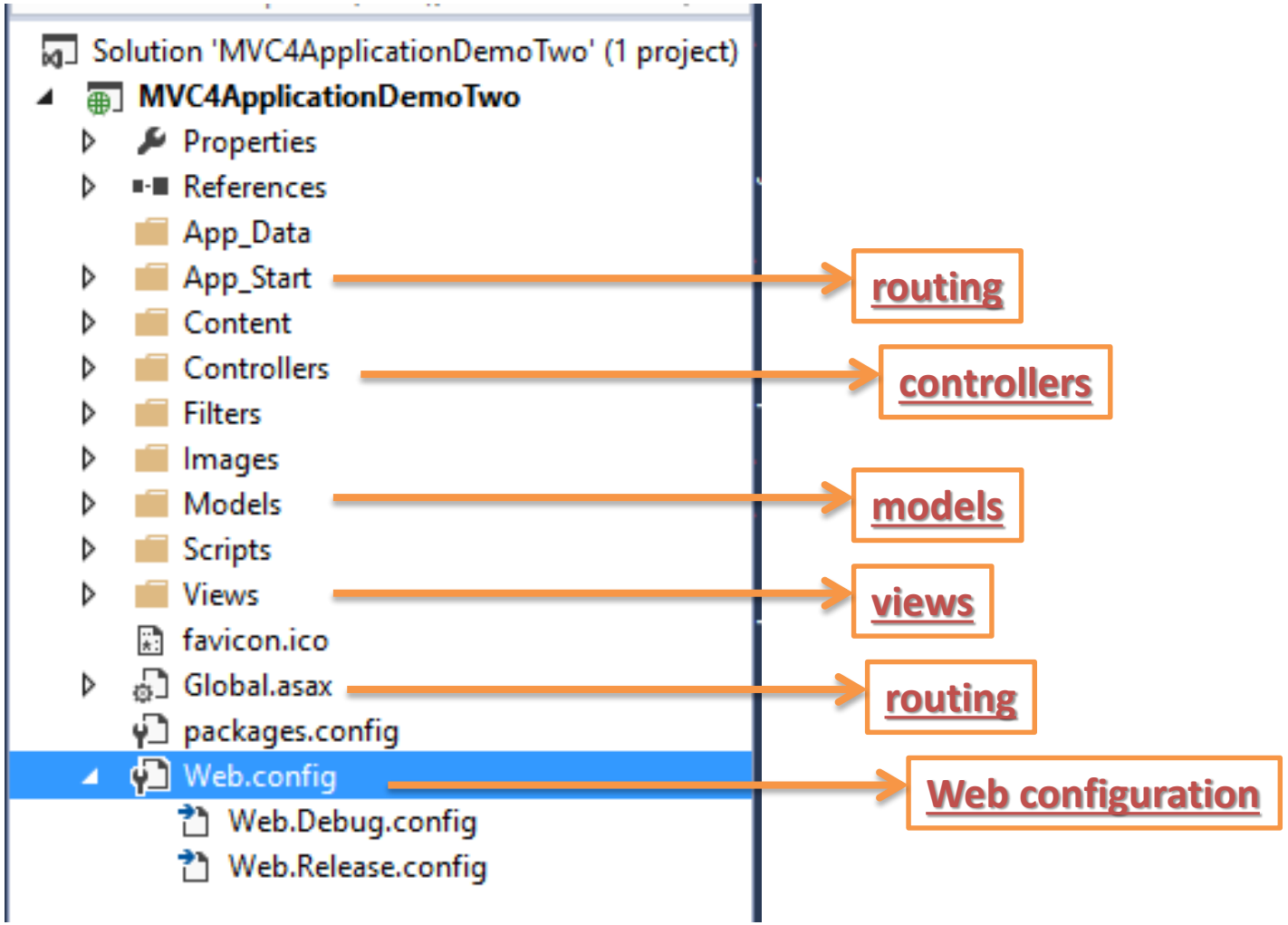
ASP.NET MVC FORM POST PARAMETERS

- ASP.NET MVC will automatically pass any query string or form post parameters named "name" to Index action method when it is invoked.

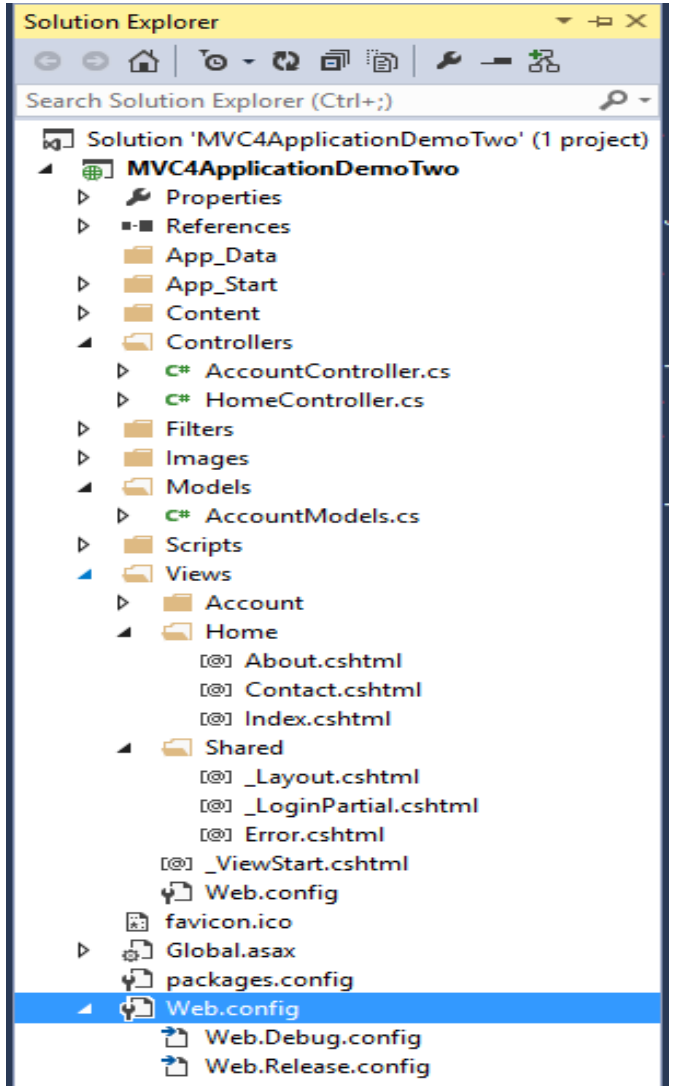
```
0 references
public class HomeController : Controller
{
    0 references
    public string Index(string id, string name)
    {
        return "Id=" + id +" name=" +name;
    }
```



Controller and View Conventions



Controller and View Conventions



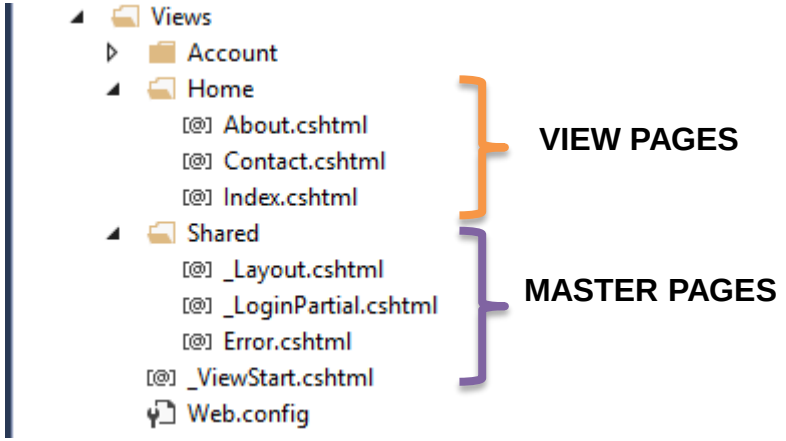
- **Controllers folder**
  - Recommended location for controllers
  - Controller type name must end with **Controller(thus omitted from the route)**
- **Views folder**
  - Recommended location for views
  - .aspx, .ascx, .master files
  - Subfolders for every controller
  - Shared folder contains views used by multiple controllers

Views/View Engine

- WebForms views
  - Use .aspx, ascx and .master files
  - No server-side form required, no viewstate used
- Razor
  - **Preferred view engine moving forward**
  - **Introduced in MVC 5.0**

Views

- **Views are .aspx files**
  - Derive from ViewPage, which derives from Page
  - Have a ViewData dictionary property populated from the controller
  - Still use markup, can still contain server-side script
  - No server-side form required
    - No _VIEWSTATE
    - No control state
- Strongly types views
  - Derive from ViewPage<T> instead of ViewPage
  - Generic type Parameter represents the type of the model



Views

- ViewData, ViewBag and TempData mechanisms to pass on data from the **Controller** to the **View.**
- **To pass data from Controller to a View. *It's always a good practice to use strongly typed view models***
- ***"@" symbol is used to switch between html and C# code***

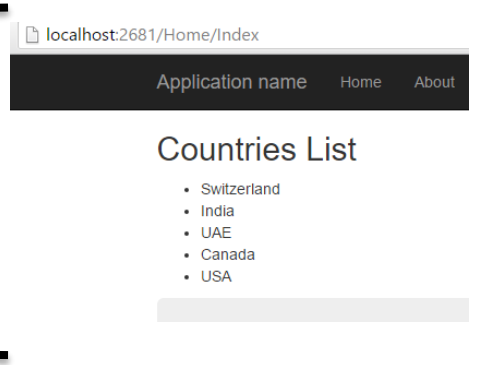
Controller Code

```
public class HomeController : Controller
{
    0 references
    public ActionResult Index( string id, string name)
    {
        ViewBag.Countries = new List<string>()
        {
            "Switzerland",
            "India",
            "UAE",
            "Canada",
            "USA"

        };

        return View();
    }
```

View Code (in html)

```
@{
    ViewBag.Title = "Countries.List";
}

<h2> Countries List</h2>
<ul>
    @foreach (string strCountry in ViewBag.Countries)
    {
        <li>@strCountry</li>
    }

</ul>
```

output



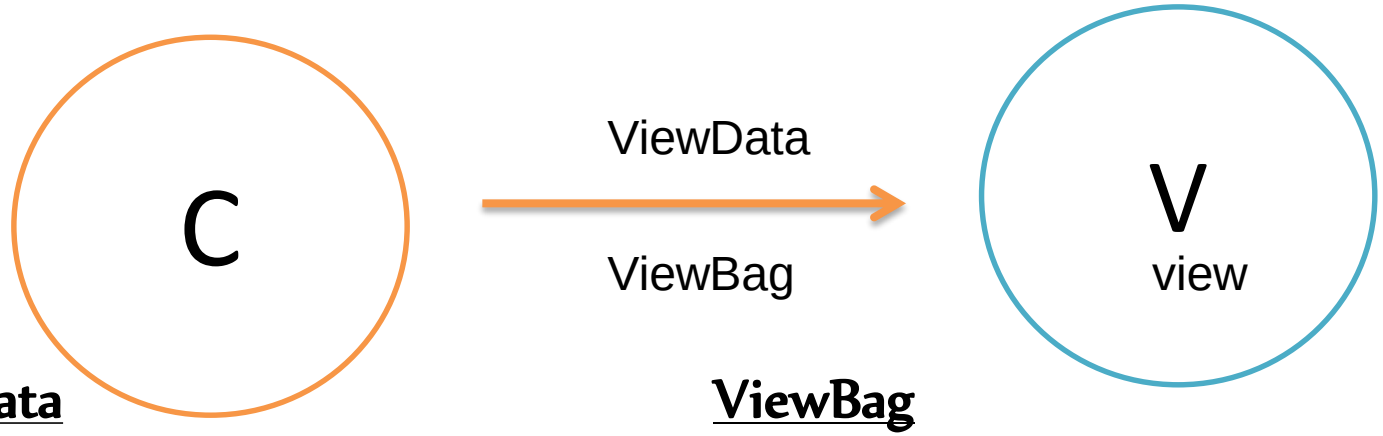
Stongly Typed View

- *It's always a good practice to use strongly typed view models*
- *ViewData and ViewBag are used to pass data from a controller to a view.*
- *Strongly types view models provide compile time error checking.*

```
0 references
public class HomeController : Controller
{
    0 references
    public ActionResult Index()
    {
        return View();
    }

    0 references
    public ActionResult About()
    {
        ViewBag.Message = "Your application description page.";

        return View();
    }
```

Both ViewData and Viewbag do not Provide compile time error checking

ViewData vs ViewBag



ViewData

- ViewData is a dictionary of objects that is derived from ViewDataDictionary class and is accessible using strings as keys.
- ViewData requires typecasting for complex data type and check for null values to avoid error.

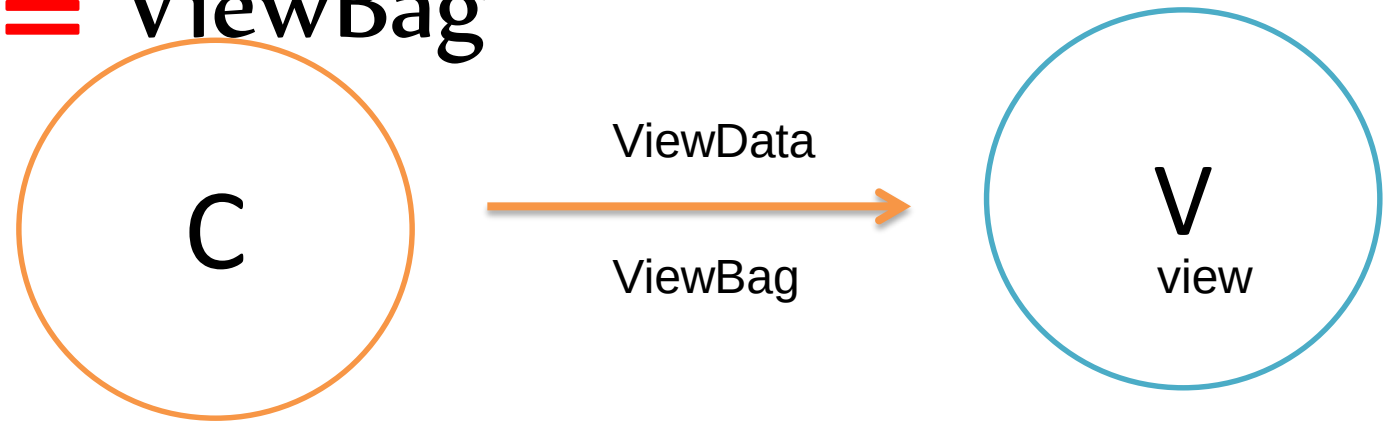
ViewBag

- A dynamic feature introduced in C# 4.0
- Allows an object to have properties dynamically added to it.
- ViewBag doesn't require typecasting for complex data type.

```
ViewData["YourKey] = "YourData";
```

```
ViewBag.YourProperty="YourData";
```

ViewData ≅ ViewBag



- Similarities between ViewBag & ViewData:
  - Helps to maintain data when you move from controller to view.
  - Used to pass data from controller to corresponding view.
  - **Short life** means value becomes null when redirection occurs. This is because their ***goal is to provide a way to communicate between controllers and views***. It's a communication mechanism within the server call.

TEMPDATA

- A dictonary derived from TempDataDictonary Class and stored in short lives Session and it is a string key and object value.
- The difference is the lifecycle of the object.
- **Works with 302/303 redirection because it's in the same HTTP Request.**
- **Used to move data from one controller to other controller or from one action to other action.**
- **On redirection, "Tempdata" helps to maintain data between those redirects.**
- **Internally uses session variables.**
- **Tempdata is used during the current and subsequent request only, means it is used when you are sure that next request will be redirecting to next view.**
- **Requires typecasting for complex data type and check for null values to avoid error.**
- **Used to store only one time messages like error messages, validation messages**

Scenarios when to use ViewBag, ViewData and TempData

- ViewBag and View Data object work well in the following scenarios:
  - Incorporating dropdown lists of lookup data into an entity
  - Components like a shopping cart
  - Widgets like a user profile widget
  - Small amounts of aggregate data

- TempData object works well in **one basic scenario**:
  - Passing data between the current and next HTTP requests

View Helpers

- Helpers available via properties of a ViewPage

Property	Class	Description
Ajax	AjaxHelper	Invoke controller actions asynchronously and update client content
Html	HtmlHelper	Create anchor tags, encode HTML
Url	UrlHelper	Create URLs to invoke controller actions

```
<div>
    @Html.ActionLink("About Us", "About", "Home");
</div>
```

→
About us

Preparing Models

- Attributes
  - Decorate properties
- Available Attributes
  - DataType Attribute
  - Display Attribute
  - Validation
    - RequiredAttribute
    - StringLength Attribute
    - RegularExpressionAttribute
    - CompareAttribute

ASP.NET WEBFORM vs MVC

Features	Web Forms	ASP.NET MVC
Separation of concerns	NO	YES
Familiar Event Driven Model	YES	NO
ViewState Issues	YES	NO
Server Controls	Yes	No
Control over HTML	No	Yes
Test Driven Development	No	Yes

ASP.NET WEBFORM vs MVC

ASP.NET Web Forms	ASP.NET MVC
ASP.NET Web Forms **uses Page controller pattern** approach for rendering layout. In this approach, every page has it's own controller i.e. code-behind file that processes the request.	ASP.NET MVC **uses Front Controller approach**. That approach means ,a common controller for all pages, processes the requests.
No separation of concerns. As we discussed that every page (.aspx) has it's own controller (code behind i.e. aspx.cs/.vb file), so both are tightly coupled.	Very clean separation of concerns. View and Controller are neatly separate.
Because of this coupled behavior, automated testing is really difficult.	Testability is key feature in ASP.NET MVC. Test driven development is quite simple using this approach.
In order to achieve stateful behavior, viewstate is used. Purpose was to give developers, the same experience of a typical WinForms application.	ASP.NET MVC approach is stateless as that of the web. So here no concept of viewstate.
Statefulness has a lots of problem for web environment in case of excessively large viewstate. Large viewstate means increase in page size.	As controller and view are not dependent and also no viewstate concept in ASP.NET MVC, so output is very clean.

ASP.NET WEBFORM vs MVC

ASP.NET WebForms model follows a Page Life cycle.	No Page Life cycle like WebForms. Request cycle is simple in ASP.NET MVC model.
Along with statefulness, microsoft tries to introduce server-side controls as in Windows applications. Purpose was to provide somehow an abstraction to the details of HTML. In ASP.NET Web Forms, minimal knowledge of HTML, JavaScript and CSS is required.	In MVC, detailed knowledge of HTML, JavaScript and CSS is required.
Above abstraction was good but provides limited control over HTML, JavaScript and CSS which is necessary in many cases.	Full control over HTML, JavaScript and CSS.
With a lots of control libraries availability and limited knowledge of other related technologies, ASP.NET WebForms is RAD(Rapid Application Development) approach.	It's a step back. For developers decrease in productivity.
It's good for small scale applications with limited team size.	It's better as well as recommended approach for large-scale applications where different teams are working together.

WebForms is an abstraction of web application programming; envisaged to ease the transition for the Visual Basic programmers who were the primary target for .NET when it was launched. Uses VB's event-based model, such as viewstate and postbacks	
Concurrency between multiple clients goes out the window.	
It is stored as a massive chunk of data in a hidden field leading to increased page load sizes.	
It is wholly dependent on POST methods, so it breaks navigation in HTML.	

Summary

- ASP.NET MVC is an alternative to WEB FORMS
  - Builds on ASP.NET, does not replace ASP.NET
- Strives for simplicity
  - Clean URLs
  - Clean HTML
- Separation of concerns
  - It's what Model, View, Controller is about.

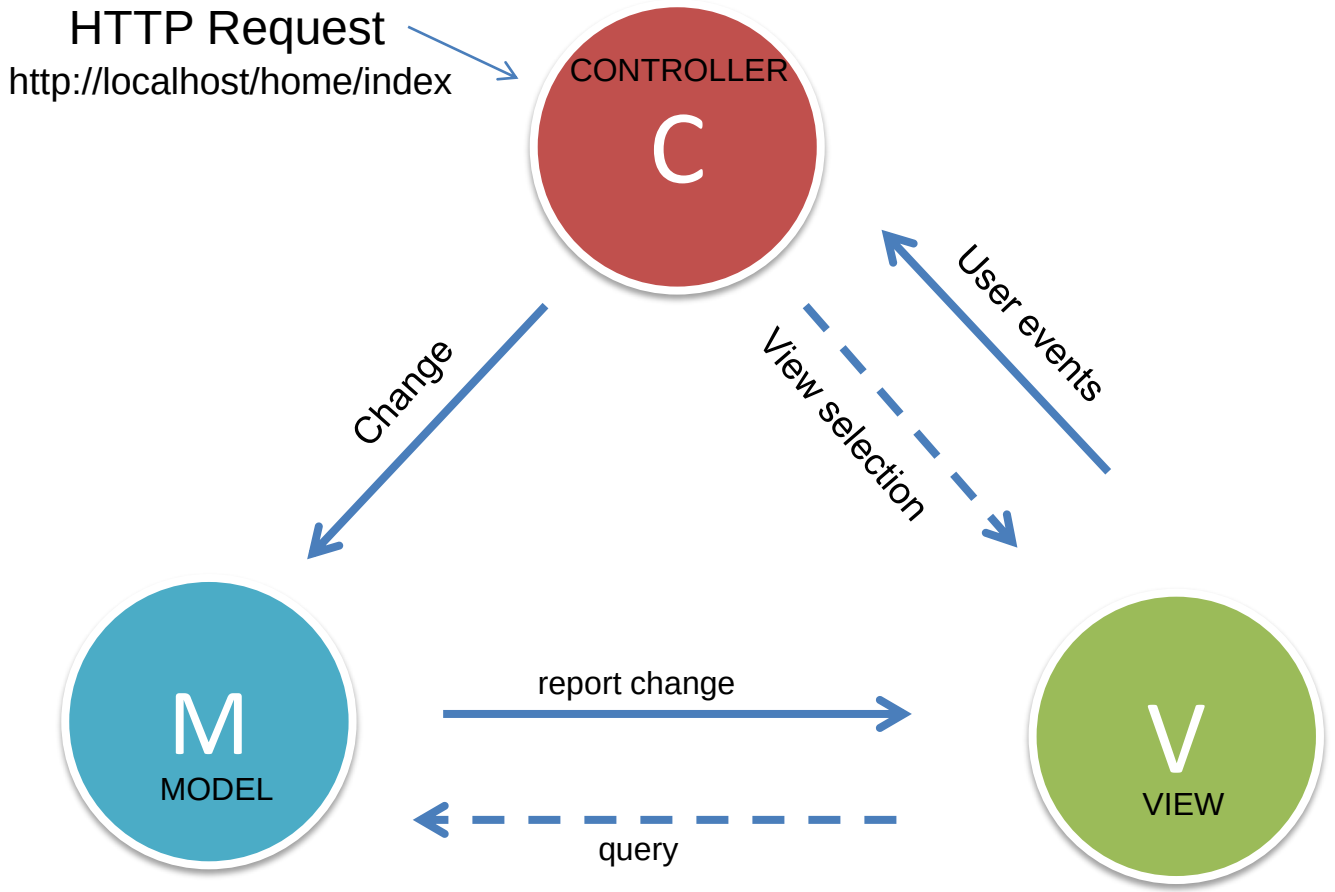
SECTION -II

CONTROLLERS

Overview

- MVC Controllers
- Routing Rules
- Controller Actions
- Action filters
- Action parameters

MVC CONTROLLER



MVC Controllers

- ASP.NET MVC Framework maps URLs to classes that are referred to as **Controllers**
- **Controllers process incoming requests, handle user inputs and interacts, execute appropriate application logic.**
- **It calls a separate view component to generate the HTML markup for the request.**
- The base class for all controllers is the ControllerBase class, which provides general MVC handling. The Controller class inherits from **ControllerBase**and is the default implement of a controller.
- The **Controller** class is responsible for the following processing stages:
  - Locating the appropriate action method to call and validating that it can be called
  - Getting the values to use as the action method's arguments
  - Handling all errors that might occur during the execution of the action method
  - Providing the default Redering Engine to render views.

Controller Action Methods


```
public class HomeController : Controller
{
    1 reference
    public ActionResult Index()
    {
        return View();
    }

    1 reference
    public ActionResult About()
    {
        ViewBag.Message = "Your application description page.";

        return View();
    }

    1 reference
    public ActionResult Contact()
    {
        ViewBag.Message = "Your contact page.";

        return View();
    }
}
```

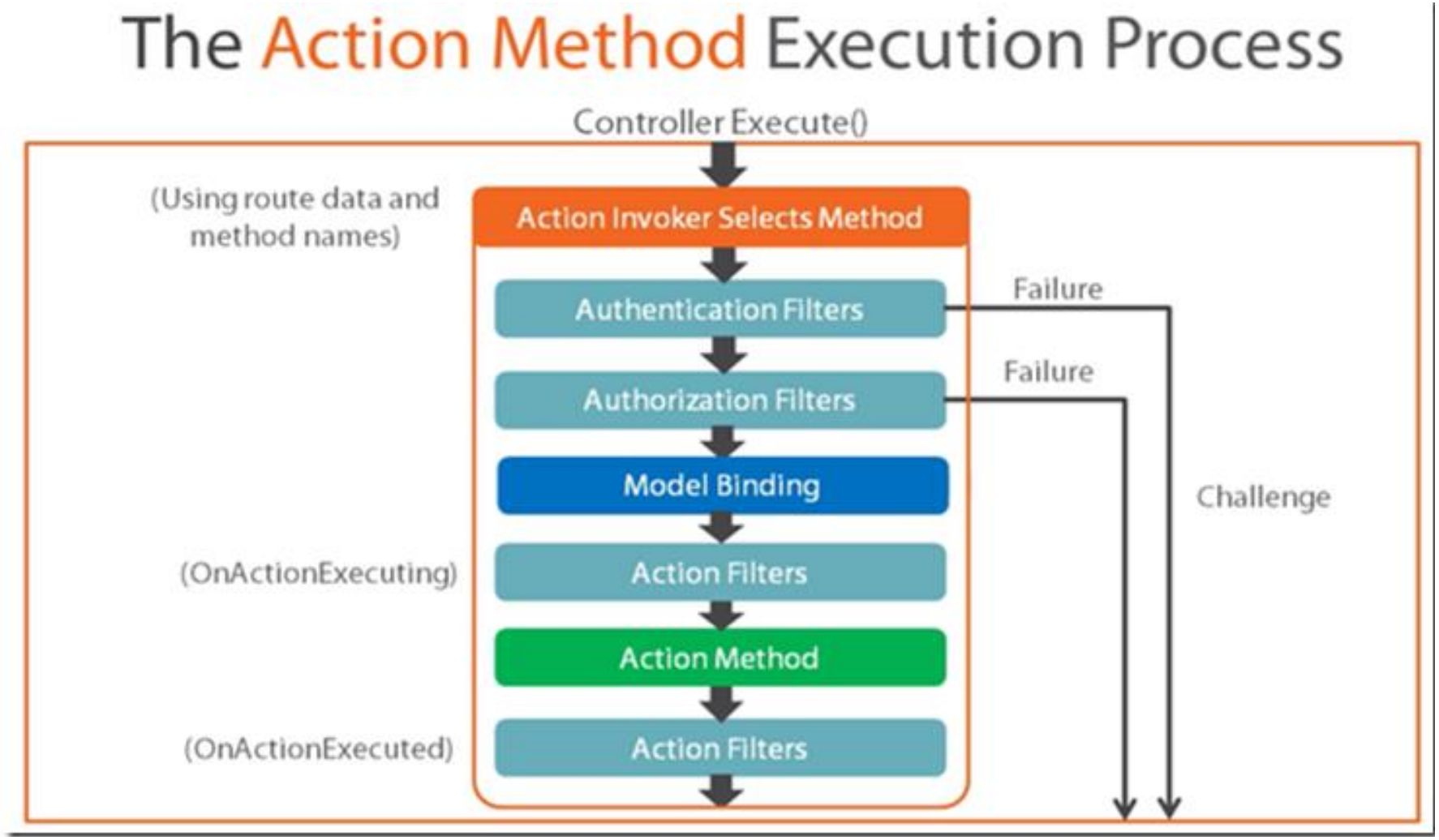
Index() Actionmethod

About() Actionmethod

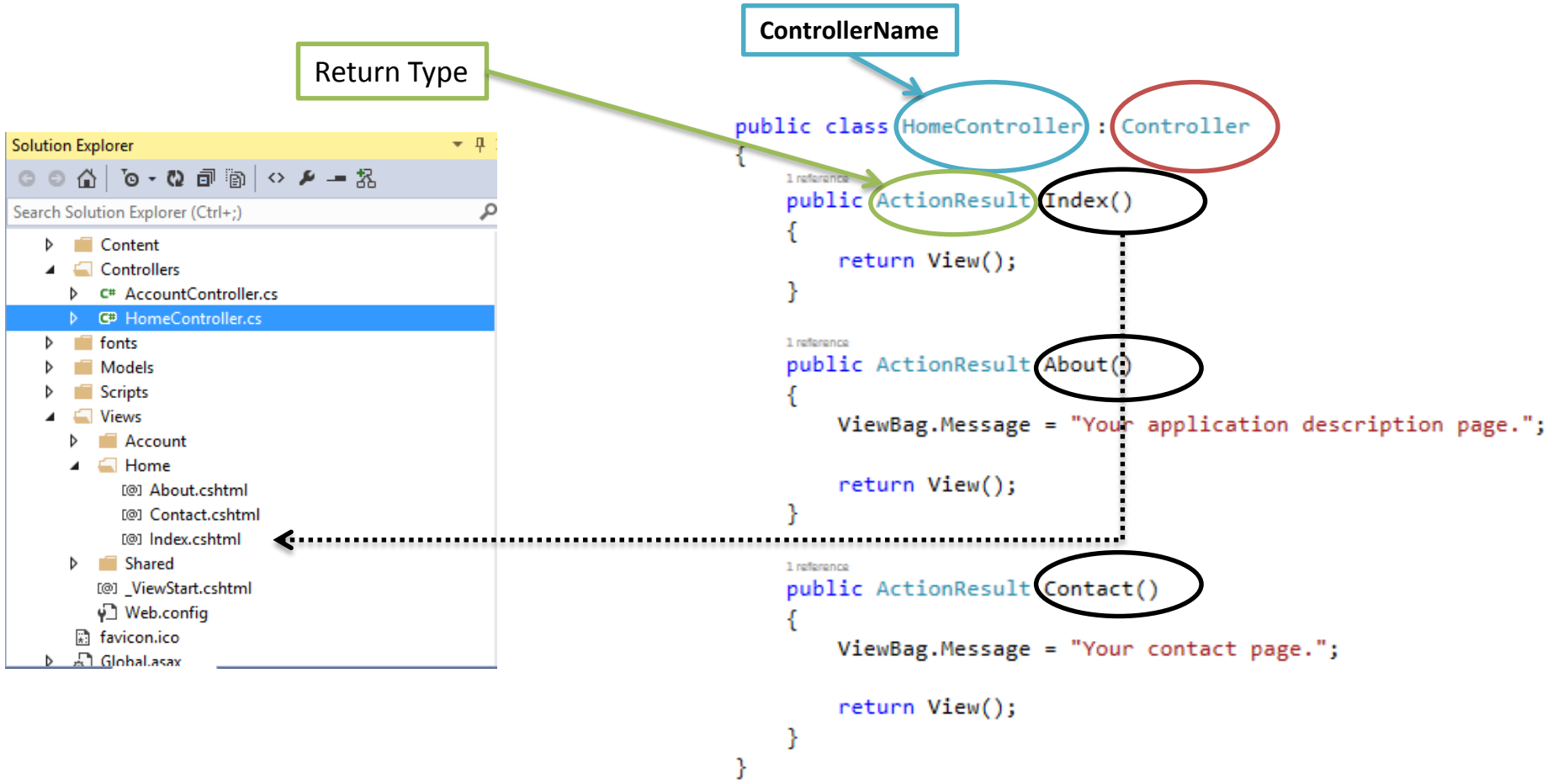
Contact() Actionmethod

- User Interaction with ASP.NET MVC Applications is organized around controllers and action methods.
- The controller defines action methods.
- Controllers can include as many action methods as needed
- **Action methods typically have a one-to-one mapping with user interactions.**
- **On http url request, MVC application uses routing rules that are defined in the GLOBAL.ASAX to parse the URL and determine the path of the Controller**
- **Controller determines appropriate action method to handle the request.**

http://localhost/home/Index -> Index()
http://localhost/home/About -> About()
http://localhost/home/Contact -> Contact()

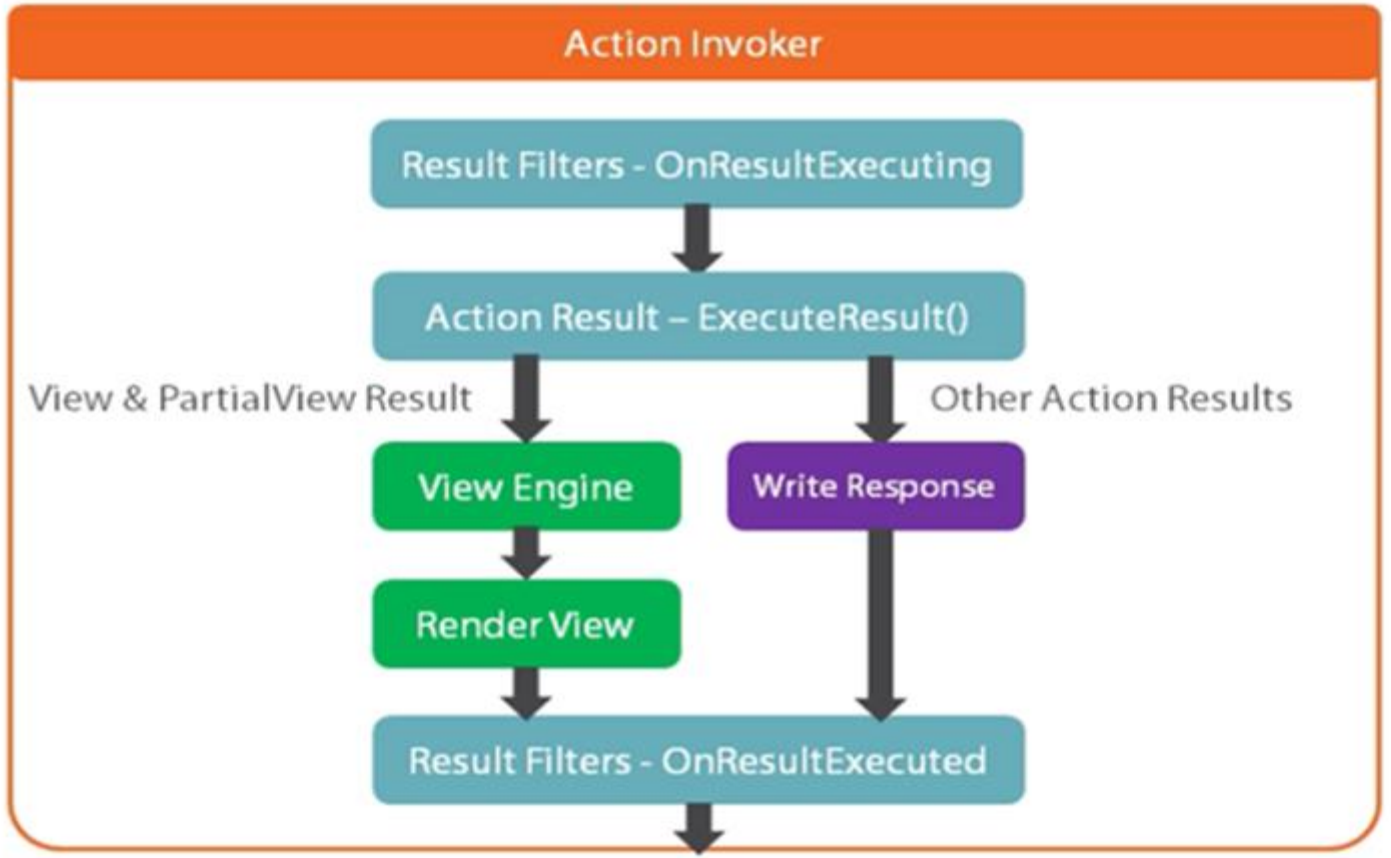


Action Results



- ActionResult Return Type
  - When creating new controllers, they will come with one or more actions by default.
  - The Empty controller includes an Index action with a return value of type **ActionResult**

The Action Result Execution Process



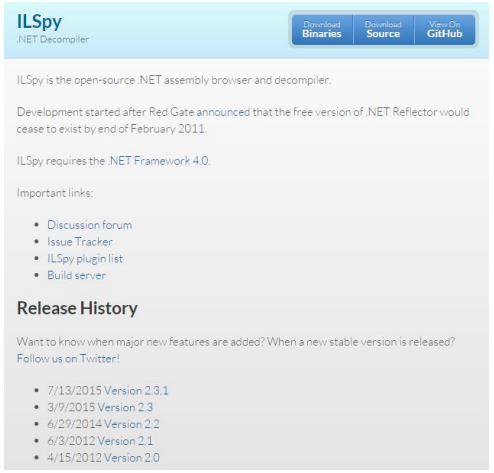
Controller Action Methods

- Actions typically return an ActionResult/VIEWRESULT

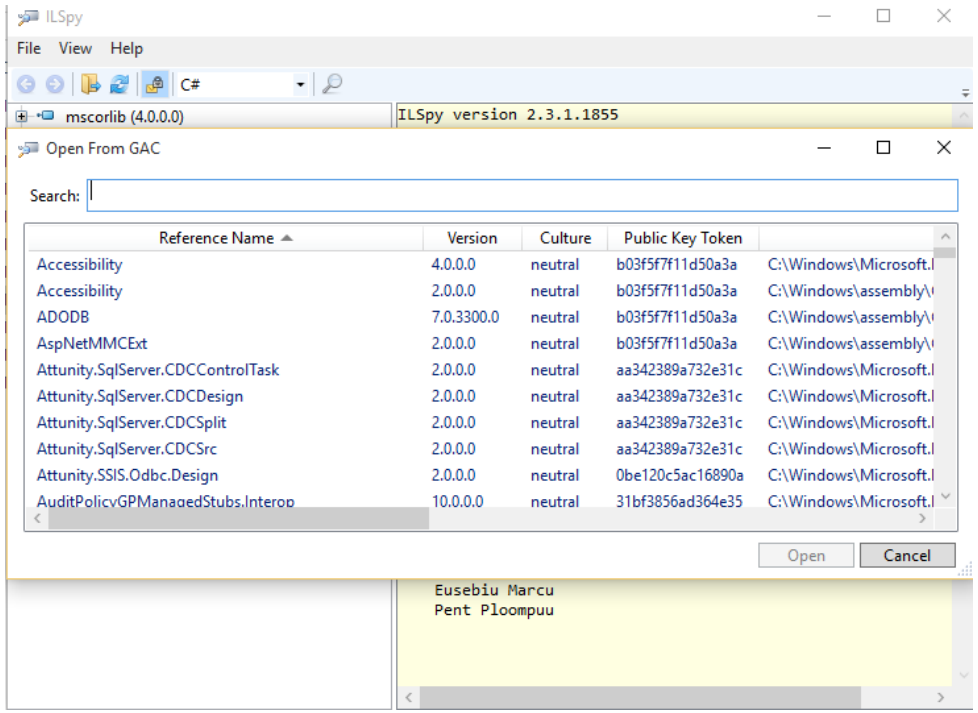
Name	FrameworkBehaviour	Producing Method
ContentResult	Returns a String literal	Content
EmptyResult	No response	
FileContentResult/ FilePathResult/FileStreamResult	Return the contents of a file	File
HttpUnauthorizedResult	Returns an HTTP 403 status	
JavaScript Result	Returns a script to execute	JavaScript
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 view engine	View/PartialView

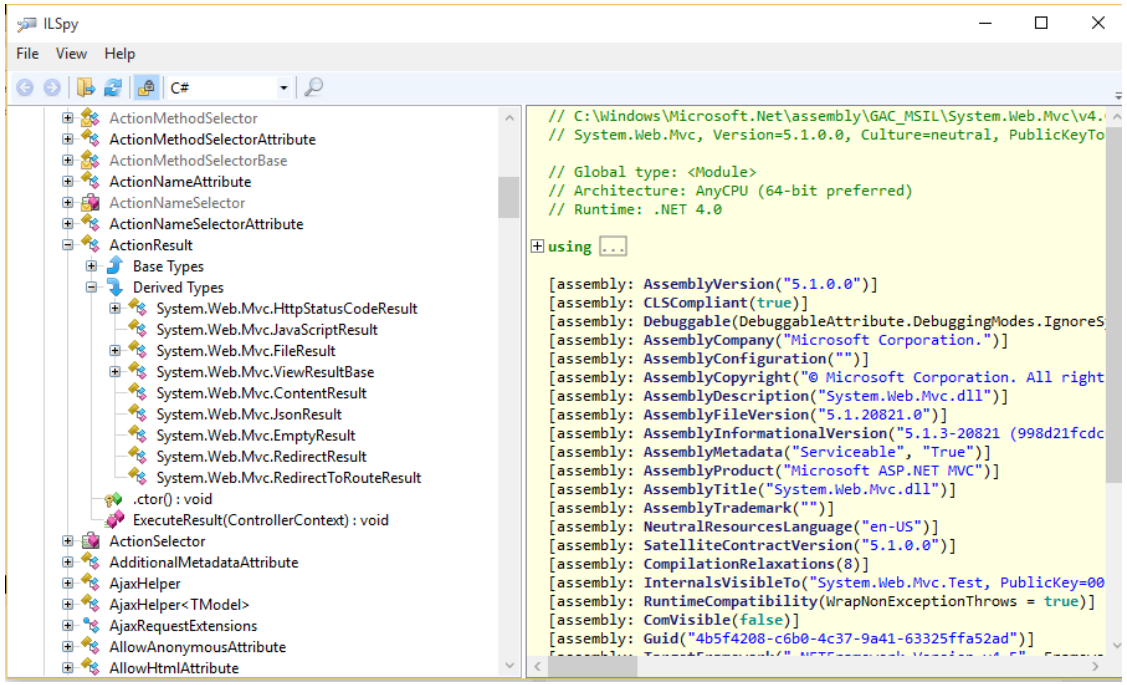
ILSPY

- An opensource .NET assembly browser and decompiler



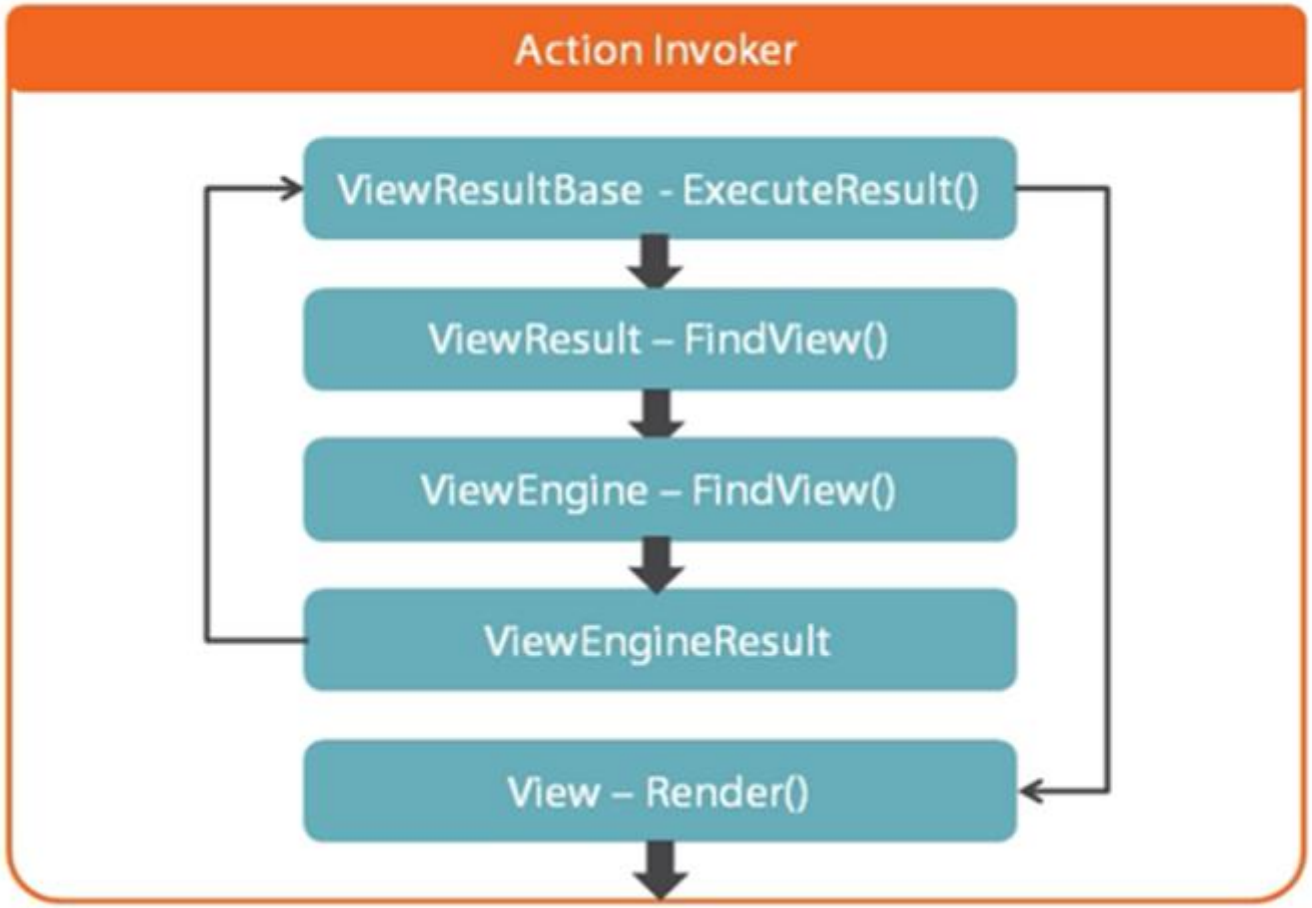
GAC – Global Assembly Cache





- Best Practice
  - Return specific sub-types, but if different paths of the action method return different subtypes, then it is better to return an ActionResult Object

The View Result Execution Process



Action Selectors

- Action selectors are **attributes** that can be applied *to action methods and are used to influence which action method gets invoked in response to a request.*
- **ACTION NAME (ALIAS THE NAME OF ACTION METHOD)**
  - Used to **change the name to invoke an action method.**
- **ACTION VERB**
  - used **when we want to control the selection of the action method based on request type.**

```
[ActionName("Location")]
1 reference
public ActionResult Contact()
{
    ViewBag.Message = "Your contact page.";

    return View();
}
```

Using action name

default

http://localhost/home/**location**

http://localhost/home/**contact**

[HttpGet]or [HttpPost] → shortcuts

```
[AcceptVerbs(HttpVerbs.Post)]
1 reference
public ActionResult About()
{
    ViewBag.Message = "Your application description
      page.";

    return View();
}
```

Why use the MVC AcceptVerbs attribute?

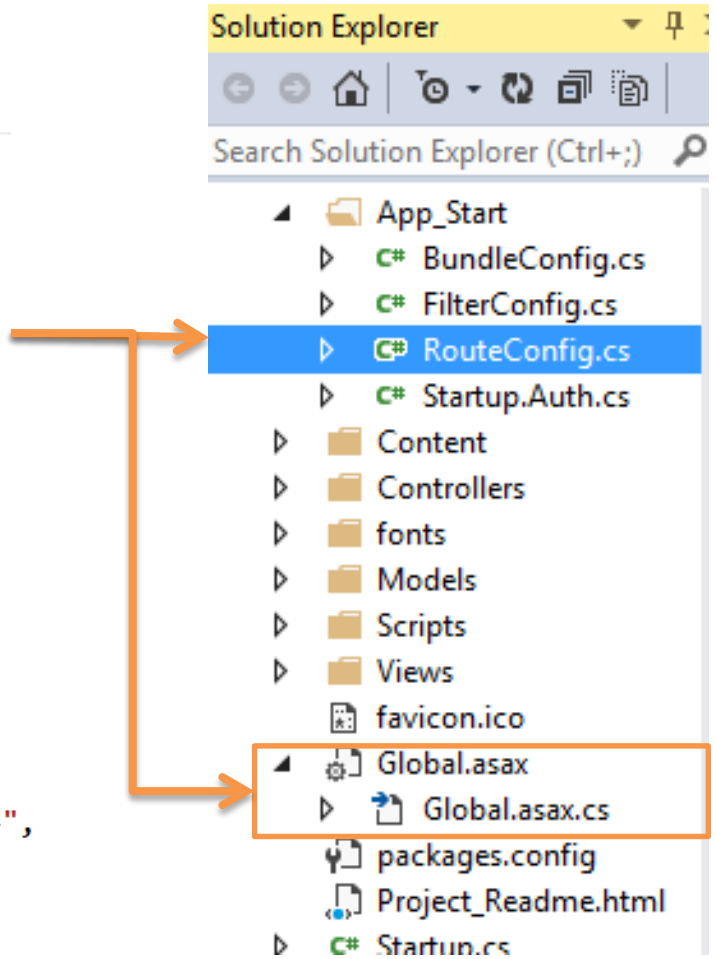
- Can be applied to action methods in a controller so that appropriate overloaded method is invoked for a given request.
- MVC will automatically **dispatch a request to appropriate action method based on the HTTP VERB.**
- Advantage of differentiating methods based on HTTP VERB is that the same URL can be used for multiple purposes (e.g. Display and edit).
  - Which is achieved with 2 separate URLs, but downside is that it makes bookmarking pages difficult.

```
[AcceptVerbs(HttpVerbs.Get)]
public ActionResult Edit(int id) {
   // code snipped
   // this is invoked when viewing the edit page
}

[AcceptVerbs(HttpVerbs.Post)]
public ActionResult Edit(int id, FormCollection formValues) {
   // code snipped
   // this is invoked when POSTing data to the edit page
}
```

Routes and Controllers

RouteData

```
public class RouteConfig
{
    1 reference
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id =
              UrlParameter.Optional }
        );
    }
}
```



Insert this in **Controlleractionmethod**

```
ViewBag.PageBreadCrumbValue = string.Format("{0}>{1}>{2}",
                    RouteData.Values["controller"],
                    RouteData.Values["action"],
                    RouteData.Values["id"]);
```

Define a New Route

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index", id =
      UrlParameter.Optional }
);
```

Action Filters

- Attributes can add pre/post processing to an action
- Or to an entire controller

```
[HandleError]
public ActionResult Index()
{

return View();

}
```

```
[Authorize]
[HandleError]
public class HomeController:Controller
{

//controller action methods go here

}
```

ACTION FILTERS

- An action filter is an **attribute** that you can **apply to individual controller action – or an entire controller – that modifies the way in which the action is executed.**
- **Users can create their own custom action filters**

Name	Description
OutputCache	Cache the output of the controller
ValidateInput	Turn Off request validation and allow dangerous input
Authorize	Restrict an action to authorized users or roles
ValidateAntiForgeryToken	Helps prevent cross site request forgeries
HandleError	Can specify a view to render in the event of an unhandled exception

```
[ActionName("Location")]
[Authorize(Roles="Admin")]
1 reference
public ActionResult Contact()
{
    ViewBag.Message = "Your contact page.";

    return View();
}
```

Action Filter

ACTION FILTER

INDIVIDUAL Controller Action Method()

```
[ActionName("Location")]
[Authorize(Roles="Admin")]
1 reference
public ActionResult Contact()
{
    ViewBag.Message = "Your contact page.";

    return View();
}
```

ENTIRE CONTROLLER

```
[Authorize(Roles="Admin")]
6 references
public class HomeController : Controller
{
```

GLOBAL

ACTION FILTER TYPES

ASP.NET MVC Framework supports four different types of filters:

If you want to control the order in **which filters of the same type are executed** then you can set a **filter's Order property**.

Authorization filters	Implements the IAuthorizationFilter Attribute	used to implement authentication and authorization for controller actions.
Action Filters	Implements the IActionFilter attribute	contains logic that is executed before and after a controller action executes. You can use an action filter, for instance, to modify the view data that a controller action returns.
Result filters	Implements the IResultFilter attribute	Result filters contain logic that is executed before and after a view result is executed. For example, you might want to modify a view result right before the view is rendered to the browser.
Exception Filters	Implements the IExceptionFilter attribute	Exception filters are the last type of filter to run. You can use an exception filter to handle errors raised by either your controller actions or controller action results. You also can use exception filters to log errors.

Filters are executed in the order listed above. For example, authorization filters are always executed before action filters and exception filters are always executed after every other type of filter.

ACTION FILTER

Name	Behaviour
Authorize	Actions are restricted to allow only authorized users.
ValidateInput	Validate all kinds of inputs
OutputCache	Caches controller's output
HandleError	User can specify custom views for handling different kind of exceptions
ValidateAntiForgeryToken	Prevent Cross Site Scripting (XSS)

Partial Views

- Similar to Web User Controls in ASP.NET Web Forms
- Reusable components
- Used to create HTML markups in separate files as a partial view
- Partial view enables you to define a view that will be rendered inside a parent view

- Generally Classified into two
  - Static
    - @Html.RenderPartial("_partialView")
    - @Html.Partial("PartialView)
  - Dynamic
    - @Html.RenderAction("PartialView)
    - @Html.Action("PartialView)

HTML.Partial

- Used to render the partial view in the view page
- Returns MVCHTMLString, which can be assigned to the variable and we are able to manipulate it if required or return it from a function
- Slower in loading when compared to the HTML.RenderPartial, since it is assigned to some variable and then used
- Used when displaying data is present in the corresponding view model.

HTML.RenderPartial

- Used to render the partial view in the view page
- Method returns void and the output will be directly written to the output stream
- Faster in loading as compared to @HTML.Partial as the output is written directly to the output stream
- Used when displaying data is present in the corresponding view model

HTML.Action

- Used to render the partial view in the view page
- Returns MVCHTMLString which can be assigned to the variable and manipulate it if required or return it from a function.
- Slower loading when compared to HTML.RenderAction.
- Child action method is required for rendering the partial view.

HTML.RenderAction

- Used to render the partial view in the view page
- Returns void and the output will be directly written to the output stream
- Faster loading when compared to HTML.Action as the output is directly written to the output stream
- Child action method is required for rendering the partial view.

Child Output Caching using Partial Views

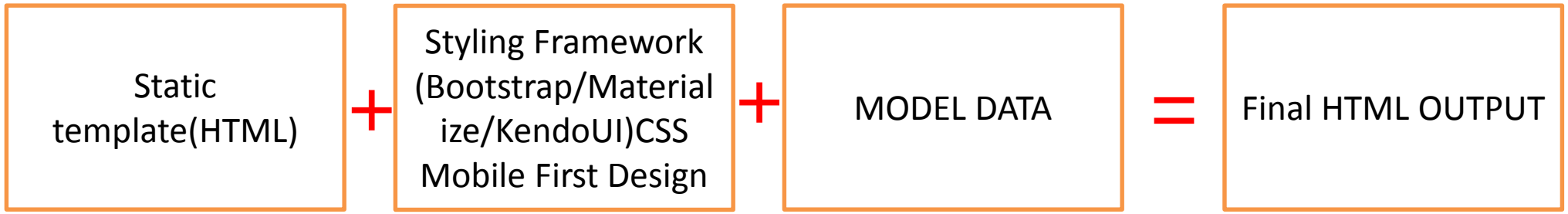
- OutputCache can now apply to child actions using partial views

```
[OutputCache(Duration=40)]
1 reference | 0/1 passing
public ActionResult Index()
{
    var currenttime = DateTime.Now;
    return View(currenttime);
}
```

```
//partial view
[ChildActionOnly]
[OutputCache(Duration = 10)]
0 references
public PartialViewResult CurrentTime()
{
    var model = DateTime.Now;
    return PartialView(model);
}
```

Summary

- Foundational knowledge about how MVC Works
  - MVC Controller
  - Action Methods and Action Parameters
  - Action Names and Action Verbs
  - Routing
  - Action Filters



SECTION -XIV

RAZOR VIEW ENGINE

Alternative to Razor View Engine are NHaml

Overview

- Razor Syntax
  - Transition between C# code and HTML code
- Layout
- HTML Helpers
- XSS (Cross Site Scripting) and CSRF (cross site request forgeries)
- Partial Views
- Introduced in ASP.NET MVC 2013.

Razor Templates

A General Purpose templating engine built upon **Microsoft's Razor parsing technology**. The RazorEngine allows you to use Razor syntax to build robust templates

- Markup syntax for adding server-based code to web pages
- It has the power of traditional ASP.NET markup, but is easier to learn and easier to use.
- Razor supports C# and Visual Basic Programming Languages

Razor Goals

- Easy to use/easy to learn
- No ties to ASP.NET runtime
  - Unlike ASP.NET – lifecycle of ASPX files have dependency on ASP.NET runtime. While Razor does not have any dependency on ASP.NET runtime to execute its code.

Razor Expression

- Razor code blocks are enclosed in @{…}
- Inline expressions (variables and functions) start with @
- Code statements end with semicolon
- Variables are declared with the var keyword
- Strings are enclosed with quotation marks
- C# code is case sensitive
- C# files have the extension **.cshtml**

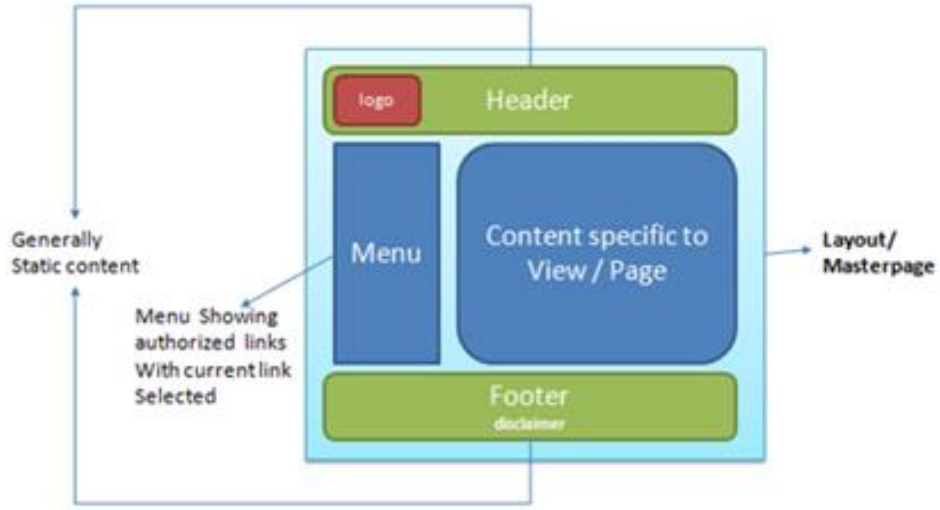
Razor code block

```
@{
    ViewBag.Title = "BestReview";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

```
<!-- Inline expression or variable -->
<p>The value of myMessage is: @myMessage</p>
```

```
<!-- Multi-statement block -->
@{
var greeting = "Welcome to our site!";
var weekDay = DateTime.Now.DayOfWeek;
var greetingMessage = greeting + " Today is: " + weekDay;
}
<p>The greeting is: @greetingMessage</p>
```

ContentBlocks




```
<html>
<body>
@RenderPage("header.cshtml")
<h1>Hello Web Pages</h1>
<p>This is a paragraph</p>
@RenderPage("footer.cshtml")
</body>
</html>
```

- With Web Pages you can use the **@RenderPage()** method to import content from separate files.
- Content block (from another file) can be imported anywhere in a web page, and can contain text, markup and code just like any regular web page
- Using common headers and footers as an example, this saves you a lot of work. You don't have to write the same content in every page, and when you change the header or footer files, the content is updated in all your pages.

Razor Syntax

Syntax/Sample	Razor	Web Forms Equivalent (or remarks)
Code Block	@{ int x = 123; string y = "because."; }	<% int x = 123; string y = "because."; %>
Expression (Html Encoded)	@model.Message	<%: model.Message %>
Expression (Unencoded)	@Html.Raw(model.Message)	<%= model.Message %>
Combining Text and markup	@foreach(var item in items) { @item.Prop }	<% foreach(var item in items) { %> <%: item.Prop %> <% } %>

Syntax/Sample	Razor	Web Forms Equivalent (or remarks)
Mixing code and Plain text	@if (foo) { <text>Plain Text</text> }	<% if (foo) { %> Plain Text <% } %>
Using block	@ using (Html.BeginForm()) { <input type="text" value="input here"> }	<% using (Html.BeginForm()) { %> <input type="text" value="input here"> <% } %>
Mixing code and plain text (alternate)	@if (foo) { @:Plain Text is @bar }	Same as above
Email Addresses	Hi philha@example.com	Razor recognizes basic email format and is smart enough not to treat the @ as a code delimiter

Syntax/Sample	Razor	Web Forms Equivalent (or remarks)
Explicit Expression	ISBN@(isbnNumber)	In this case, we need to be explicit about the expression by using parentheses.
Escaping the @ sign	In Razor, you use the @@foo to display the value of foo	@@ renders a single @ in the response.
Server side Comment	@* This is a server side multiline comment *@	<%-- This is a server side multiline comment --%>
Calling generic method	@(MyClass.MyMethod<AType>())	Use parentheses to be explicit about what the expression is.
Creating a Razor Delegate	@{ Func<dynamic, object> b = @@item; } @b("Bold this")	Generates a Func<T, HelperResult> that you can call from within Razor. See this blog post for more details.
Mixing expressions and text	Hello @title. @name.	Hello <%: title %>. <%: name %>.

NEW IN RAZOR v2.0/ASP.NET MVC 4

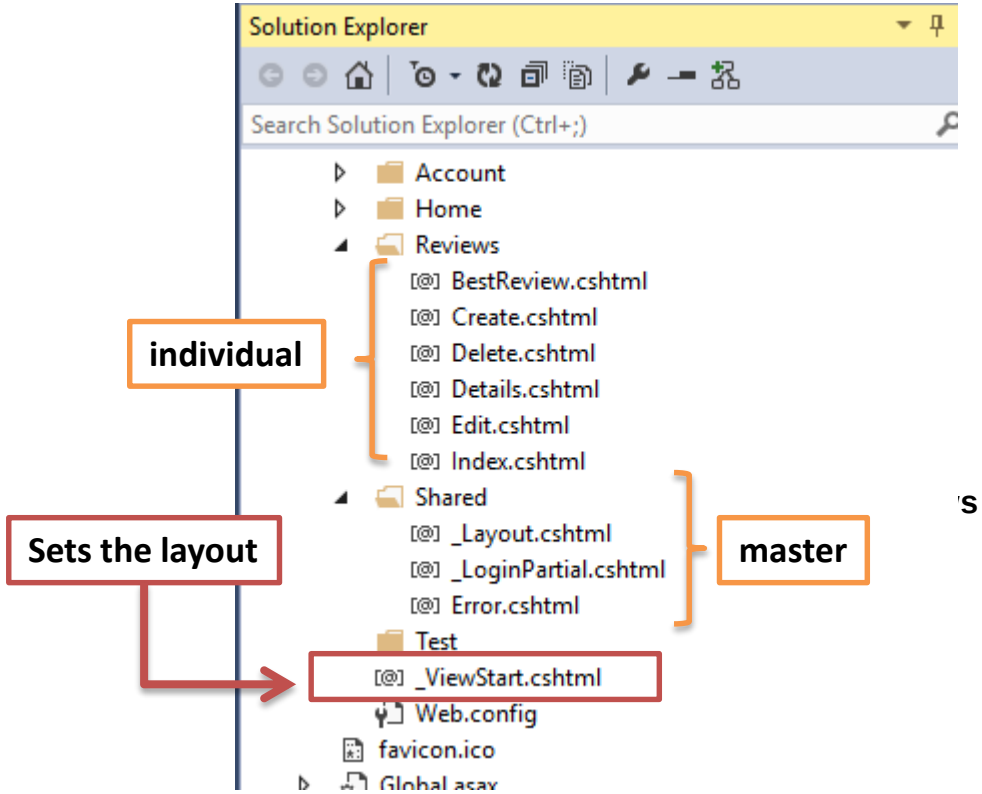
Syntax/Sample	Razor	Web Forms Equivalent (or remarks)
Conditional attributes		When className = null When className = "" When className = "my-class"
Conditional attributes with other literal values		When className = null *Notice the leading space in front of* foo *is removed.* When className = "my-class"
Conditional data-* attributes. *data-* attributes are always rendered.*		When xpos = null or "" When xpos = "42"
Boolean attributes	<input type="checkbox" checked="@isChecked" />	When isChecked = true<input type="checkbox" checked="checked" />When isChecked = false<input type="checkbox" />
URL Resolution with tilde	<script src="~/myscript.js"> </script>	When the app is at /<script src="/myscript.js"> </script>When running in a virtual application named MyApp<script src="/MyApp/myscript.js"> </script>

Razor Layout

http://razorcheatsheet.com/

- Layout views are "master pages" for razor
- Use inherited method to specify content areas
  - RenderBody
  - RenderSection
- Common UI Structure for application

```
<!DOCTYPE html>
<html>
<head>
    <title>@ViewBag.Title</title>
    <script src="@Url.Content("~/Scripts/jquery-1.4.4.min.js")"
            type="text/javascript"></script>
</head>

<body>
    @RenderBody()
</body>
</html>
```



Razor Layout

```
            <div id="logindisplay">
                @Html.Partial("_LogOnPartial")
            </div>

            <div id="menucontainer">

                <ul id="menu">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                </ul>

            </div>
        </div>

        <div id="main">
            @RenderBody()
            <div id="footer">
            @RenderSection("Footer", true);
            </div>
        </div>
    </div>
</body>
</html>
```

HTML Helper

- HTML Helper extension method generates html elements based on **model properties**
- **Html.TextBox() is loosely type method while Html.TextBoxFor() is strongly type (generic) extension method**
- **It is advisable to use "For" extension methods for compile time type checking e.g. TextBoxFor, CheckBoxFor etc..**

HTML Helpers

- HTML Helpers are used to modify HTML output (easy to create small blocks of HTML)
- HTML is a property of the ViewPage base class
  - Create input
  - Create links
  - Create forms

HTML.ActionLink

```
Html.ActionLink(article.Title,
                "Item",   // <-- ActionMethod
                "Login",  // <-- Controller Name.
                new { article.ArticleID }, // <-- Route arguments.
                null  // <-- htmlArguments .. which are none. You need this value
                      //     otherwise you call the WRONG method ...
                      //     (refer to comments, below).
                )
```

```
routes.MapRoute(
    "Default",     // Route name
    "{controller}/{action}/{id}",                           // URL with parameters
    new { controller = "Home", action = "Index", id = "" }  // Parameter defaults
);
```

HTML.ActionLink()

- The easiest way to render an HTML link in is to use the **HTML.ActionLink()** helper.
- With MVC, the **Html.ActionLink()** does not link to a view. It creates a link to a controller action.

Property	Description	
@Html.ActionLink()		
@Html.linkText		
@Html.action()		
@Html.AntiForgeryToken()		
@Html.RouteCollection()		

HTML Helper : FORM ELEMENTS

- HTML helpers are available for eve kind of form control
- HTML label elements use descripti text to form control and provide usability improvements
- Each helper method provides a shorthand way to render valid HTML for the specific control

Helper	HTML Element
Html.CheckBox	<input type="checkbox" />
Html.DropDownList	<select></select>
Html.Hidden	<input type="hidden" />
Html.Label	<label for="" />
Html.ListBox	<select></select> or <select multiple></select>
Html.Password	<input type="password" />
Html.Radio	<input type="radio" />
Html.TextArea	<textarea></textarea>
Html.TextBox	<input type="text" />

```
@using ( Html.BeginForm() )
{
    <fieldset>
        <legend>Product</legend>

        <div class="editor-label">
            @Html.LabelFor(model => model.Name)
        </div>
        <div class="editor-field">
            @Html.TextBoxFor(model => model.Name)
        </div>

        <div class="editor-label">
            <div style="float: left;">
                @Html.CheckBoxFor(model => model.Featured)
            </div>
            @Html.LabelFor(model => model.Featured)
        </div>

        <!-- Price, Featured, ... -->
    </fieldset>
}
```

Built-in ASP.NET MVC Framework: HTML HELPERS

HTML Helper	Generates HTML Controls
Html.BeginForm	Form element
Html.EndForm	
Html.extBox	Textbox
Html.TextArea	Text area
Html..Password	Password TextBox
Html..Hidden	Hidden field
Html..CheckBox	Checkbox
Html..RadioButton	Radio button
Html..DropDownList	Dropdownbox, combobox
Html..ListBox	Mult-select list box
Html.Display	HTML text
Html.Label	Label

HTML HELPERS FOR STRONGLY TYPED VIEWS

HTML Helper	Generates Html Control
Html.TextBoxFor	Textbox
Html.TextAreaFor	TextArea
Html.PasswordFor	Password textbox
Html.HiddenFor	Hidden field
Html.CheckBoxFor	Checkbox
Html.RadioButtonFor	Radiobutton
Html.DropDownListFor	Dropdown, combobox
Html.ListBoxFor	Multiselect list box
Html.DisplayFor	Html text
Html.LabelFor	Label
Html.EditorFor	Generate Html controls based on data type of specified model property

```
@using (Html.BeginForm()) {
        @Html.ValidationSummary(true)
            <div class="editor-label">
                @Html.LabelFor(model => model.FirstName)
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.FirstName)
                @Html.ValidationMessageFor(model => model.FirstName)
            </div>
```

TryUpdateModel()

- **TryUpdateModel**() allows you to bind parameters to your model inside your action.
- useful if you want to load your model from a database then update it based on user input rather than taking the entire model from user input.
- use this method to update the model that backs a particular view via the given controller.

```
public class Student {
  public string studentID { get; set; }
}

// ... in the controller
public ActionResult Save() {
  var myStudent = new Student();
  TryUpdateModel(myStudent);
}
```

Custom Helpers

- can create custom HTML Helpers that you can use within your MVC views.
- By taking advantage of HTML Helpers, you can reduce the amount of tedious typing of HTML tags that you must perform to create a standard HTML page.

We need a custom helper for image tag as shown below

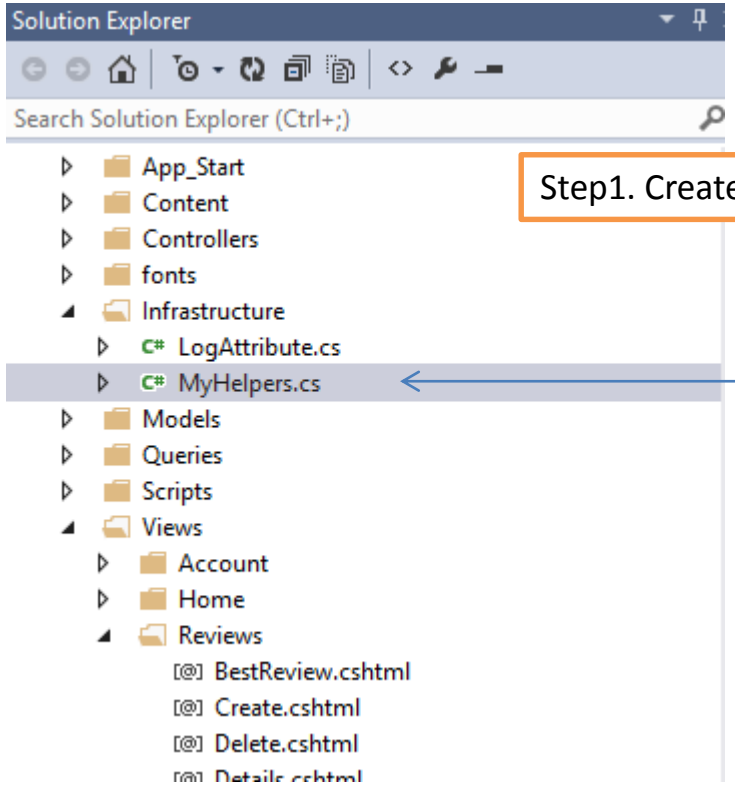
```
<img src="@item.Restaurant.ImageUrl" alt="@item.Restaurant.Name" />
```

Custom Html helper tag in the view that generates above html snippet:

```
@Html.Image(item.Restaurant.ImageUrl, item.Restaurant.Name)
```

Introduced in C# 3.0 Extension Methods to create custom helpers

Steps to Create a Custom Helper



Step1. Create a Helpers Class

Step2: Write the builder function

```
namespace ZomatoReviewApp.Infrastructure
{
    0 references
    public class MyHelpers
    {
        //html required output
        //<img src="@item.Restaurant.ImageUrl" alt="@item.Restaurant.Name"/>
        //custom helper
        //@html.Image(@item.Restaurant.ImageUrl, @item.Restaurant.Name)
        0 references
        public static MvcHtmlString Image(this
          System.Web.WebPages.Html.HtmlHelper helper,
            string src, string altText)
        {
            var builder = new TagBuilder("img");
            builder.MergeAttribute("src", src);
            builder.MergeAttribute("alt", altText);
            return MvcHtmlString.Create(builder.ToString
              (TagRenderMode.SelfClosing));
        }
    }
}
```

Step3: include the namespace in the views

@ using zomatoreviewapp.Infrastructure **(locally)**

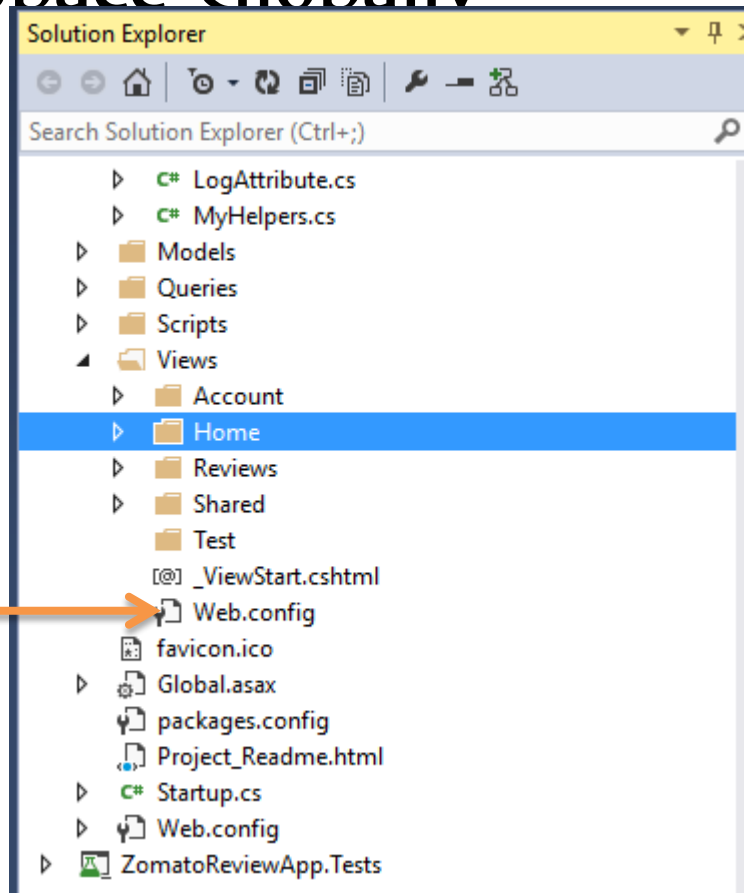
Step4: Use Html.Image() in View

@Html.Image(item.Restaurant.ImageUrl, item.Restaurant.Name)

Registering Custom Helper in NameSpace Globally

Now you can use the
@html.Image()
Custom helper globally in all the views

1.Open Web.config



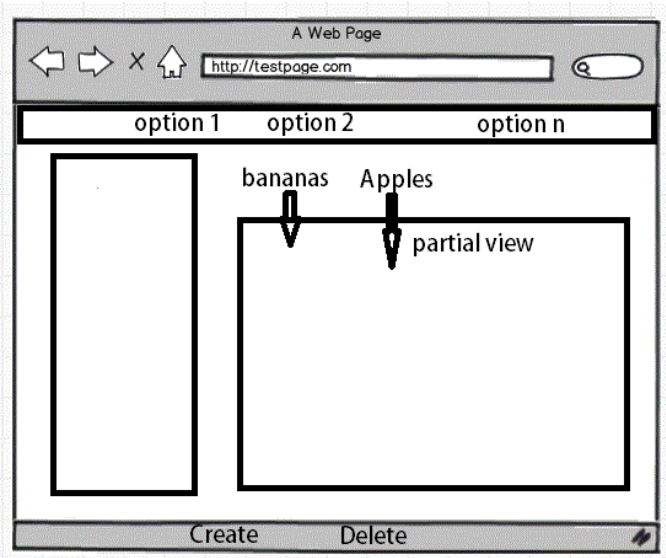
2.add

```
<system.web.webPages.razor>
  <host factoryType="System.Web.Mvc.MvcWebRazorHostFactory, System.Web.Mvc,
  <pages pageBaseType="System.Web.Mvc.WebViewPage">
    <namespaces>
      <add namespace="System.Web.Mvc" />
      <add namespace="System.Web.Mvc.Ajax" />
      <add namespace="System.Web.Mvc.Html" />
      <add namespace="System.Web.Optimization"/>
      <add namespace="System.Web.Routing" />
      <add namespace="ZomatoReviewApp" />
      <add namespace="ZomatoReviewApp.Infrastructure" />
    </namespaces>
  </pages>
</system.web.webPages.razor>
```

http://www.asp.net/mvc/overview/older-versions-1/views/creating-custom-html-helpers-cs

Partial Views

- Allows users to put html and C# code into file for code reusability.
- A partial view enables you to define a view that will be rendered inside a parent view.
- By using partial view we can render a view inside a parental view and to create reusable content in the project




```
<div id="main">
    @RenderBody()
    <div id="footer">
    @Html.Action("BestReview", "Reviews")
    @RenderSection("Footer", false)
    </div>
</div>
```

```
<h2>The Latest Reviews</h2>

@foreach (var item in Model)
{
    @Html.Partial("_Review", item)
}
<p>
    @Html.ActionLink("Create New", "Create")
</p>

@section Footer {

    <p>This is the footer</p>

}
```

Security

- Encoding
  - Helps to avoid XSS(cross site scripting) attacks
  - Not encoding user input makes you particularly vulnerable

- **Html.AntiForgeryToken**
  - Helps to avoid CSRF attacks
  - Requires a ValidateAntiForegeryToken attribute on controller action
  - Valid only for POST operators
  - Cross Site Request forgery is a type of a hack where the hacker exploits the trust of a website on the user.
  - the site trusts the user (because they have authenticated themselves) and accepts data that turns out to be malicious.

Summary

- Razor Syntax – Implicit and Explicit expressions
- HTML Helpers and Creating Custom HTML Helpers
- Razor Layout
- Partial Views
- Introduced the concepts of XSS and CSRF

SECTION -XIV

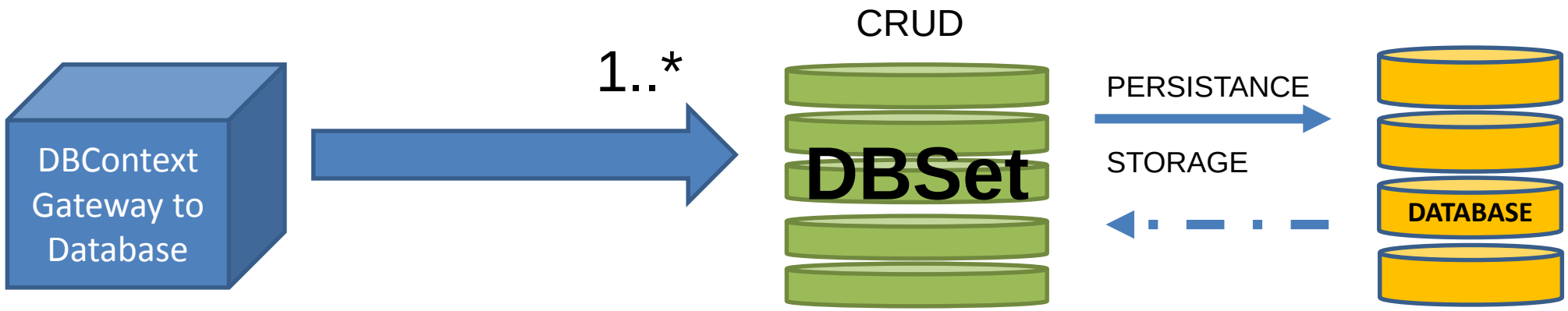
WORKING WITH DATA IN ASP.NET MVC USING ENTITY FRAME WORK DB FIRST

Object Relational Mapping Landscape

- ADO.NET
- LINQ TO SQL
- Entity Framework v1.0
- nHibernate
- Entity Framework 4
- Entity Framework 4.1 : DbContext
- Entity Framework 5
- Entity Framework 6
- Entity Framework 7 (Complete Rewrite)
- Dapper
- ORMLite
- PetaPocos

Entity Framework

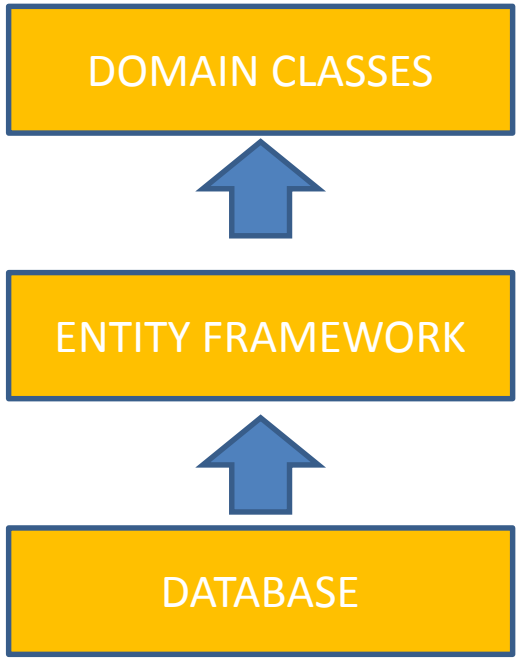
VERSIONS
- EF 4.0
- EF 5.0
- EF 6.0
- EF CORE



Approaches

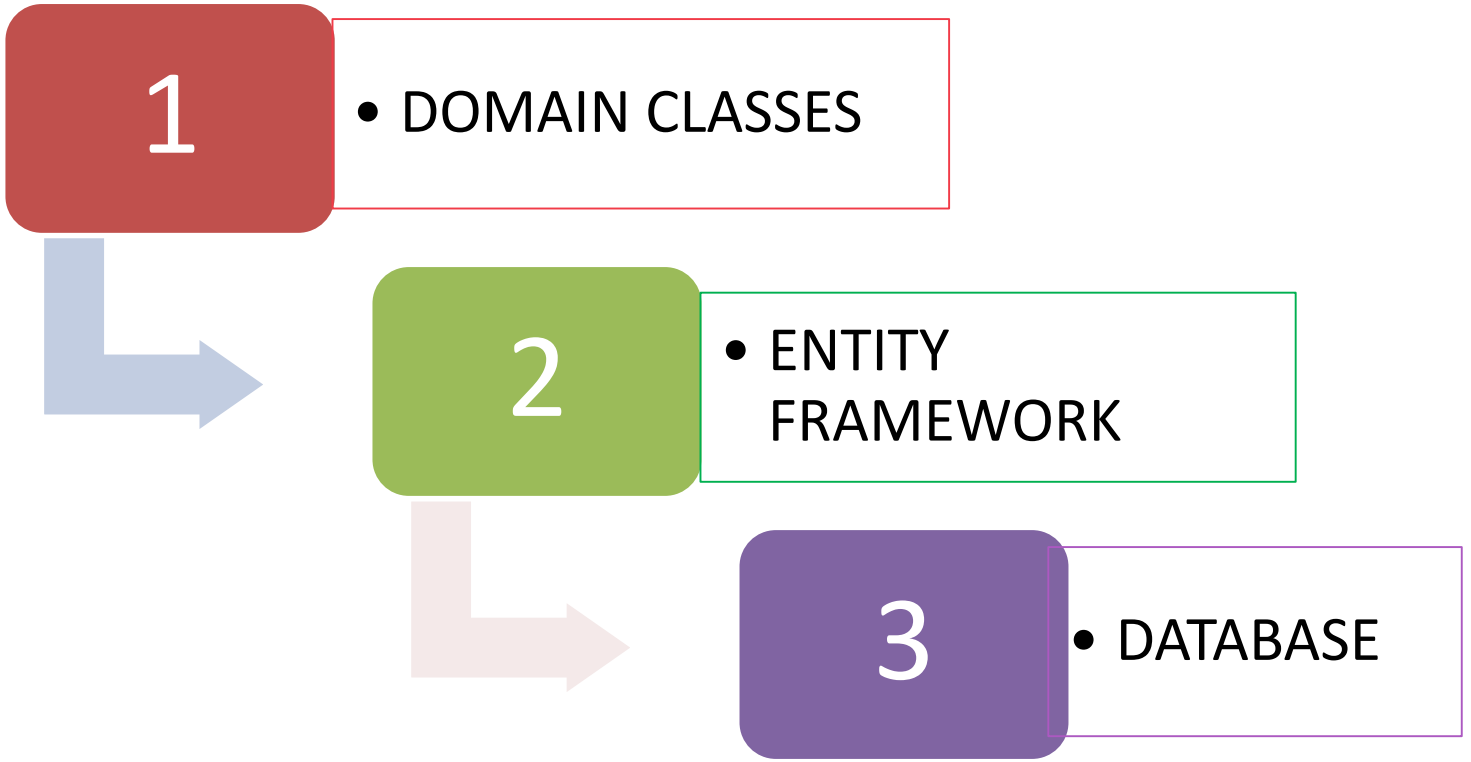
DBFIRST APPROACH

DBA FRIENDLY

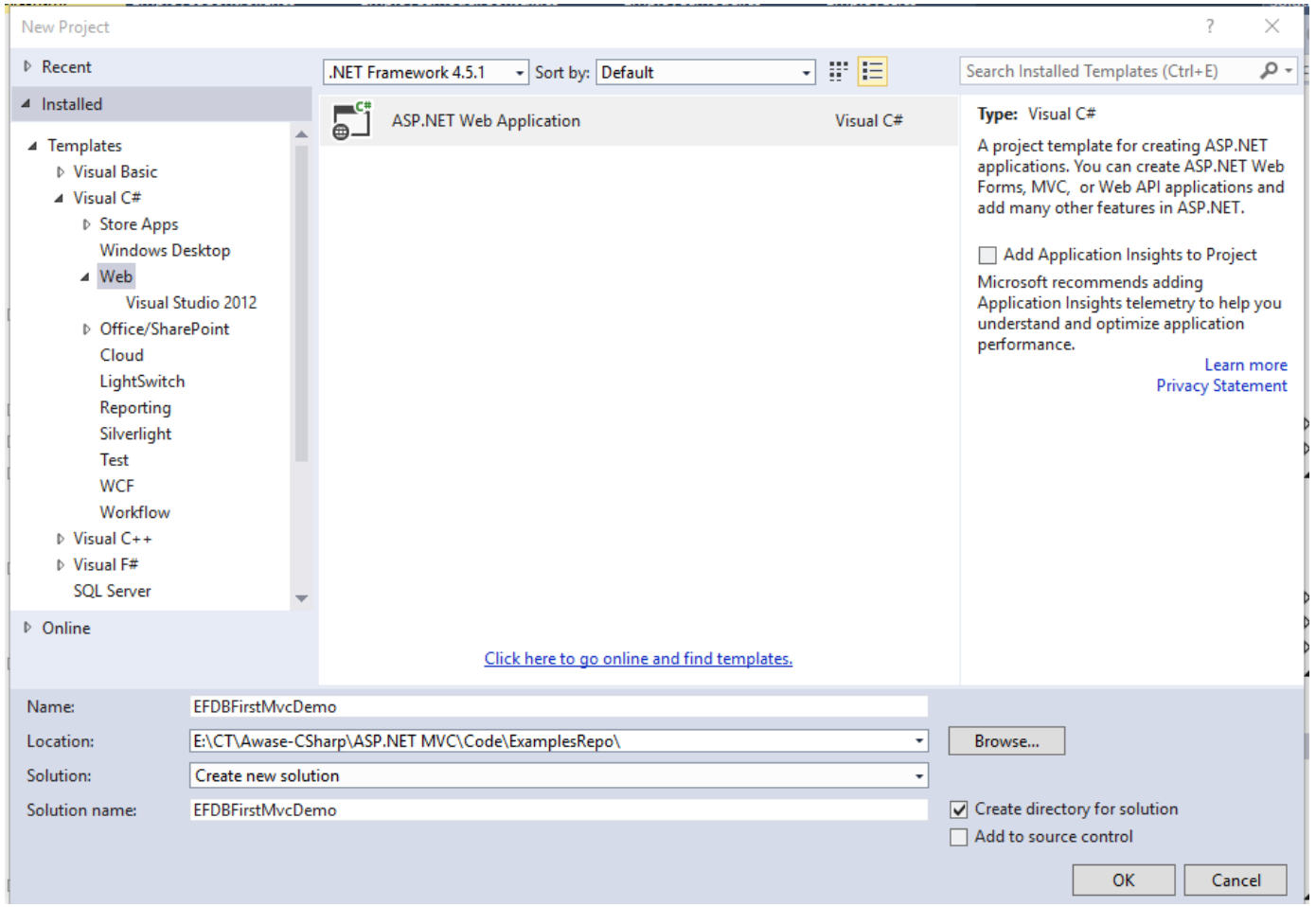


CODE FIRST APPROACH

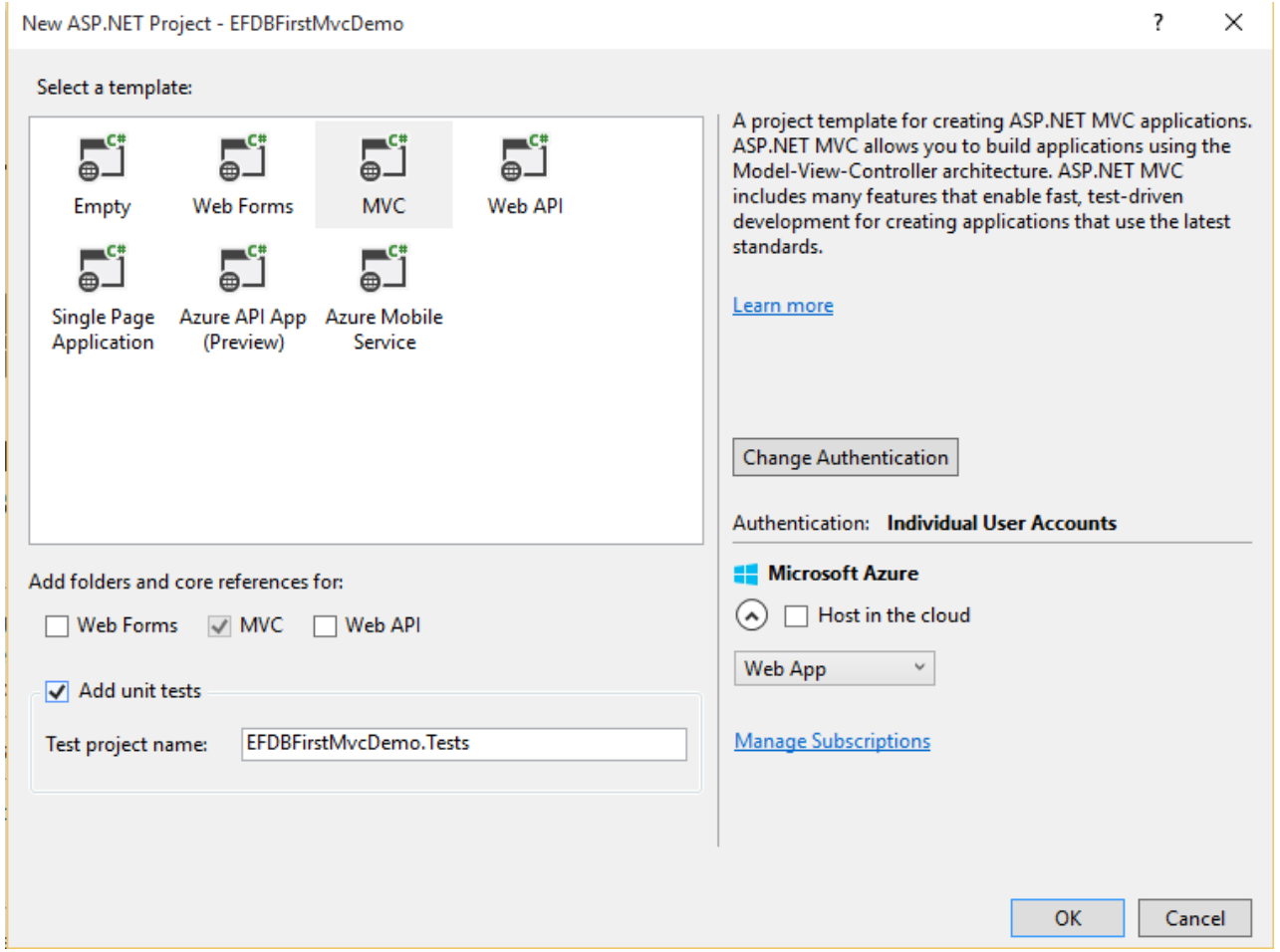
DEVELOPER FRIENDLY



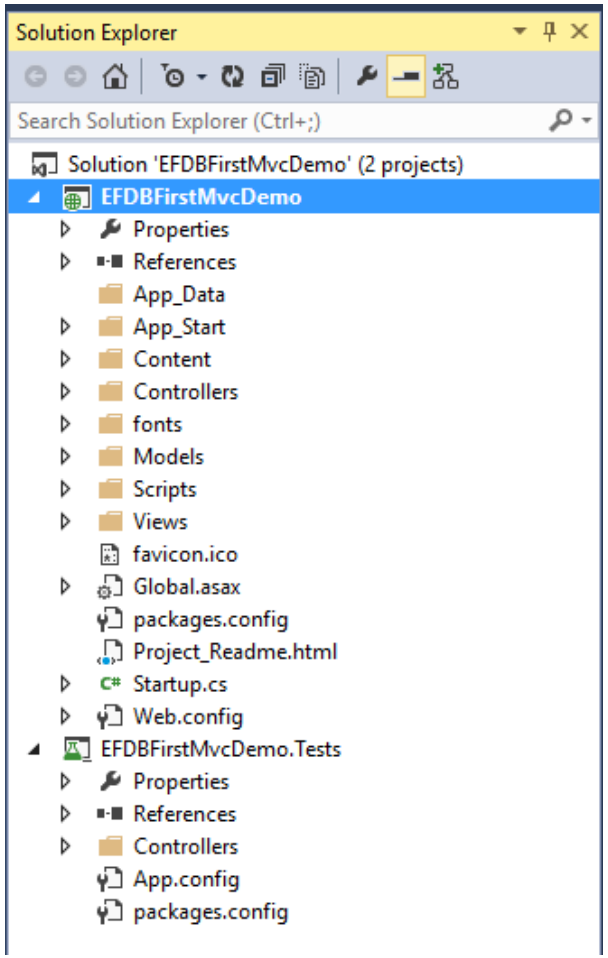
1. Create ASP.NET Web Application



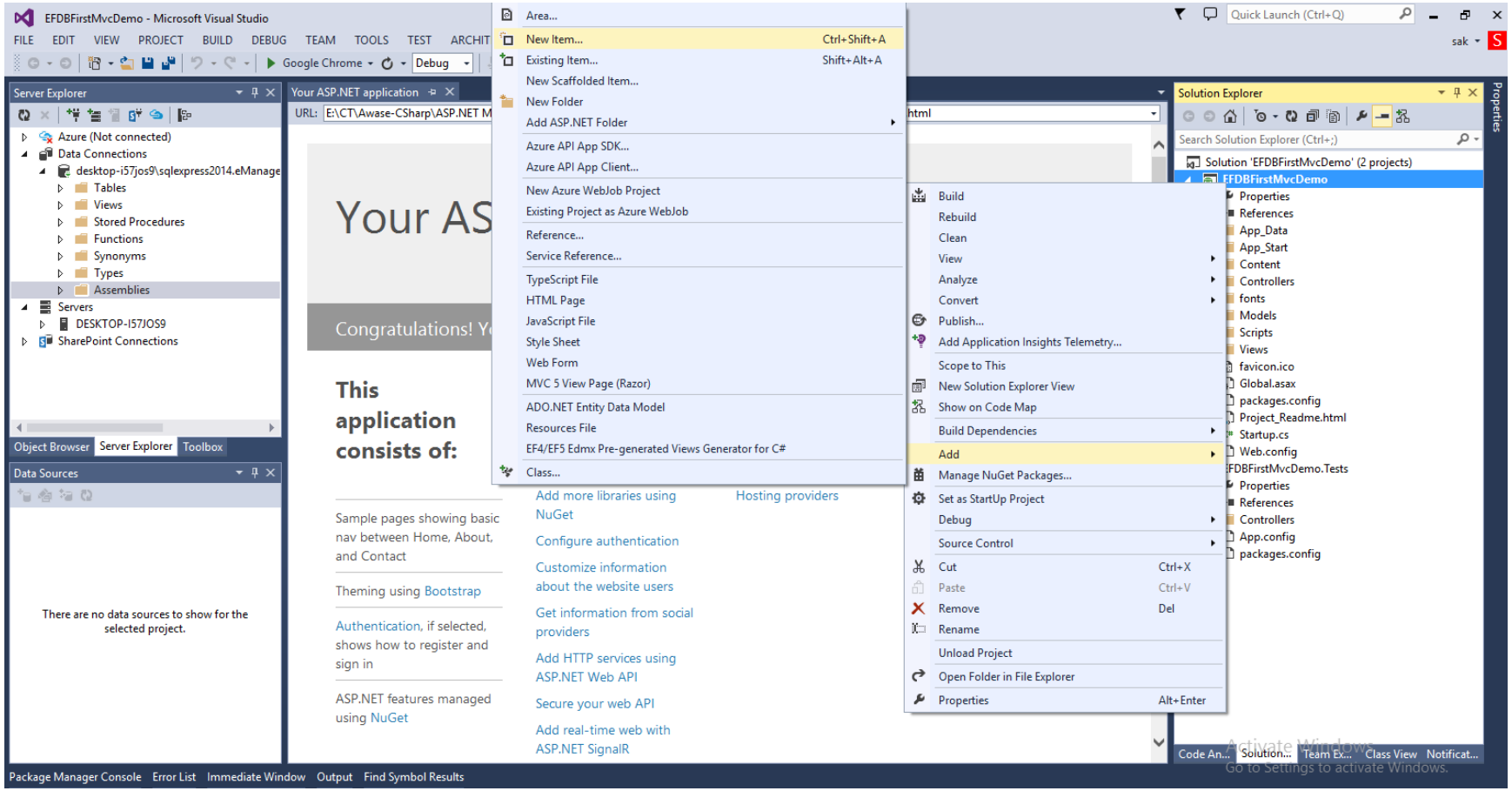
2. Select MVC Web Application with Unit Testing



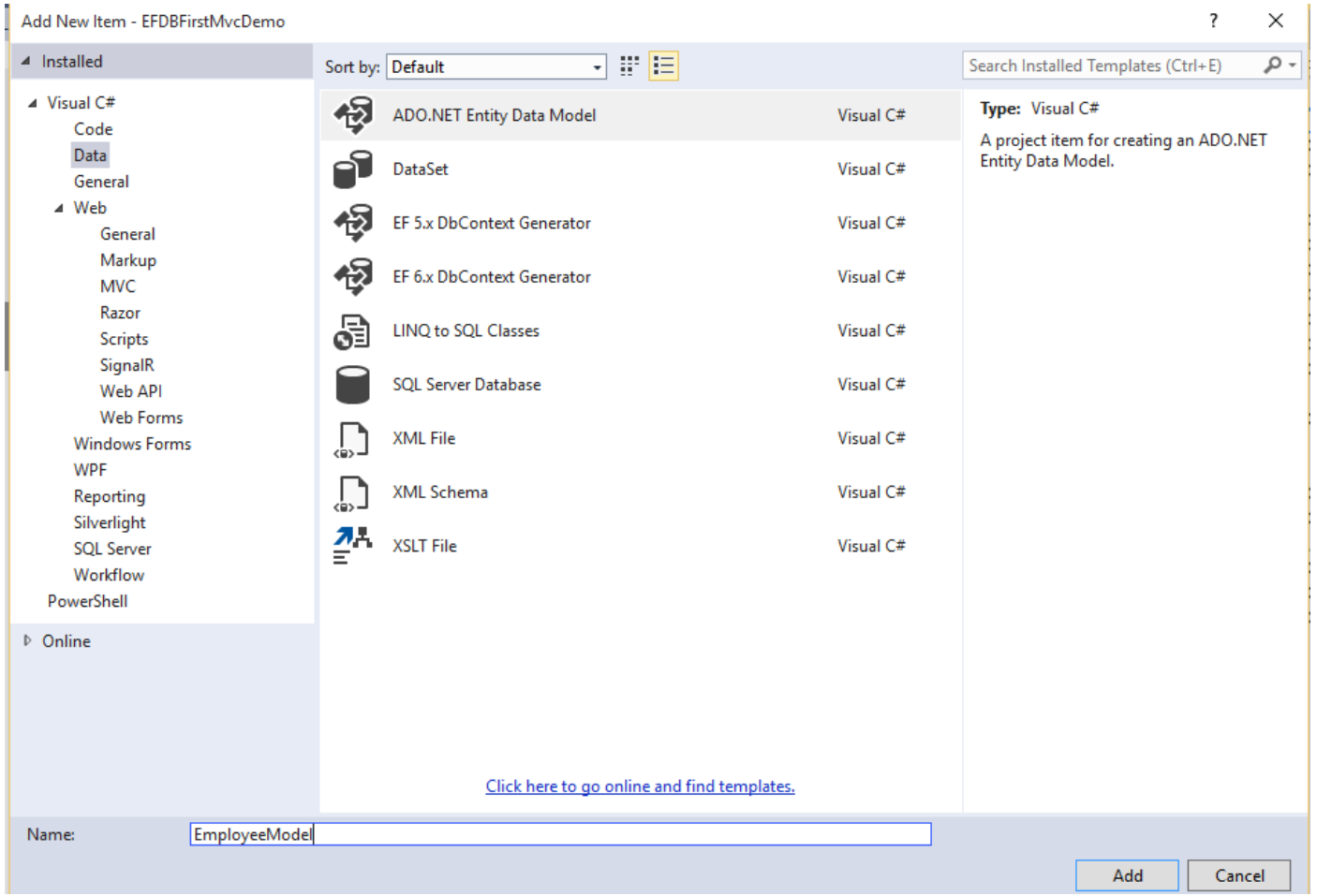
3. Scaffolding MVC Application



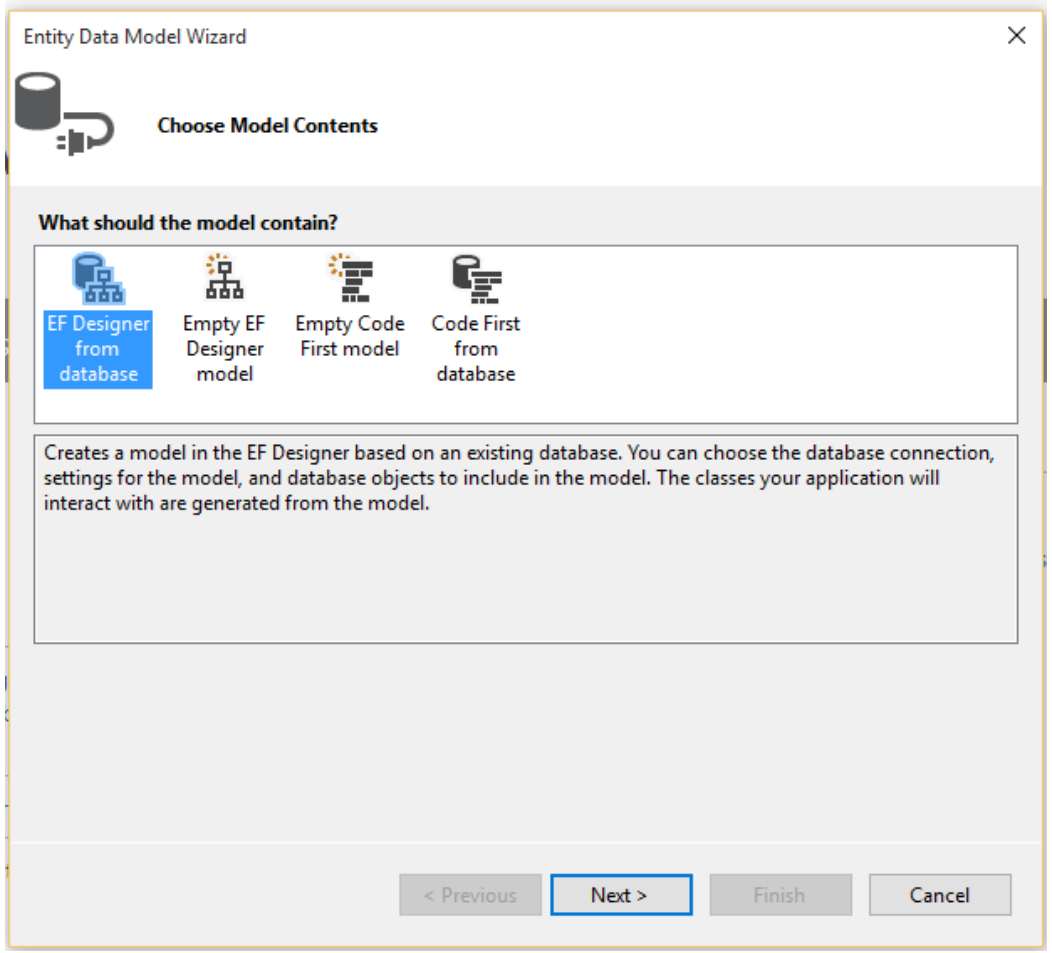
4. Add -> New Item



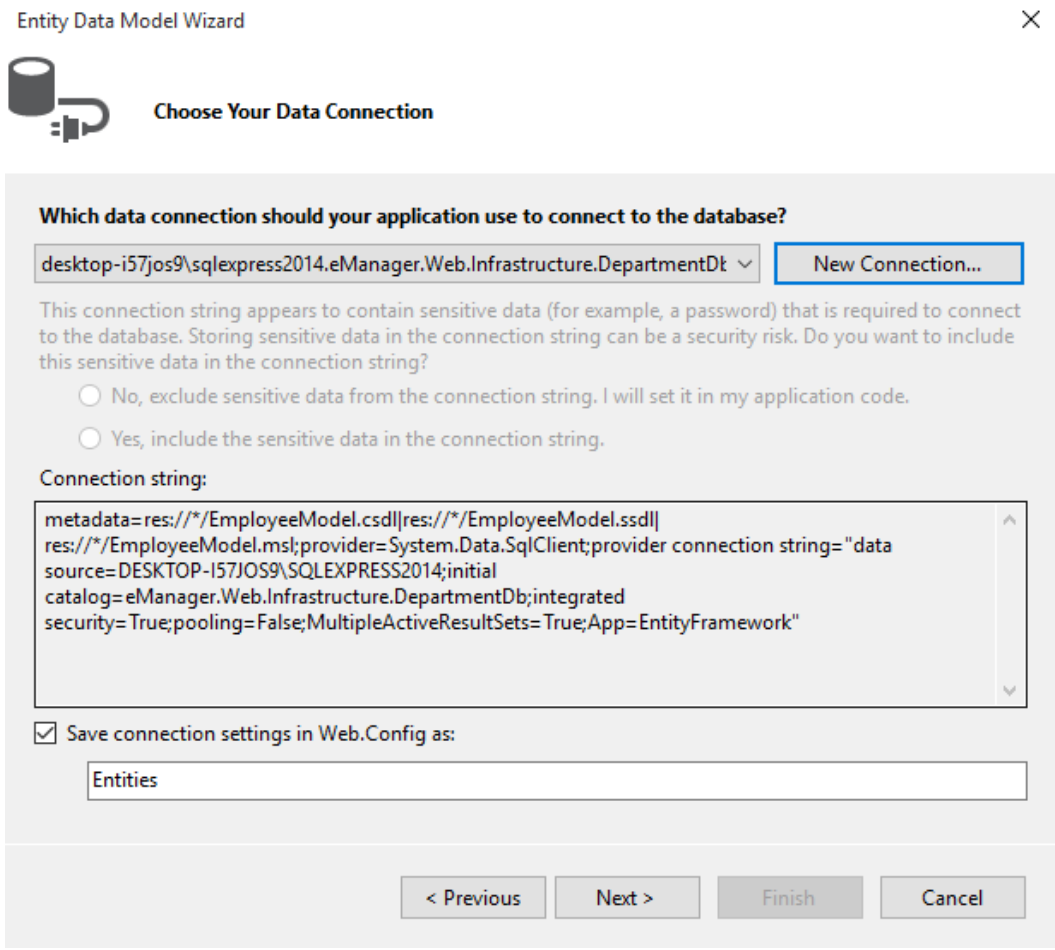
5. Create ADO.NET ENTITY DATA MODEL



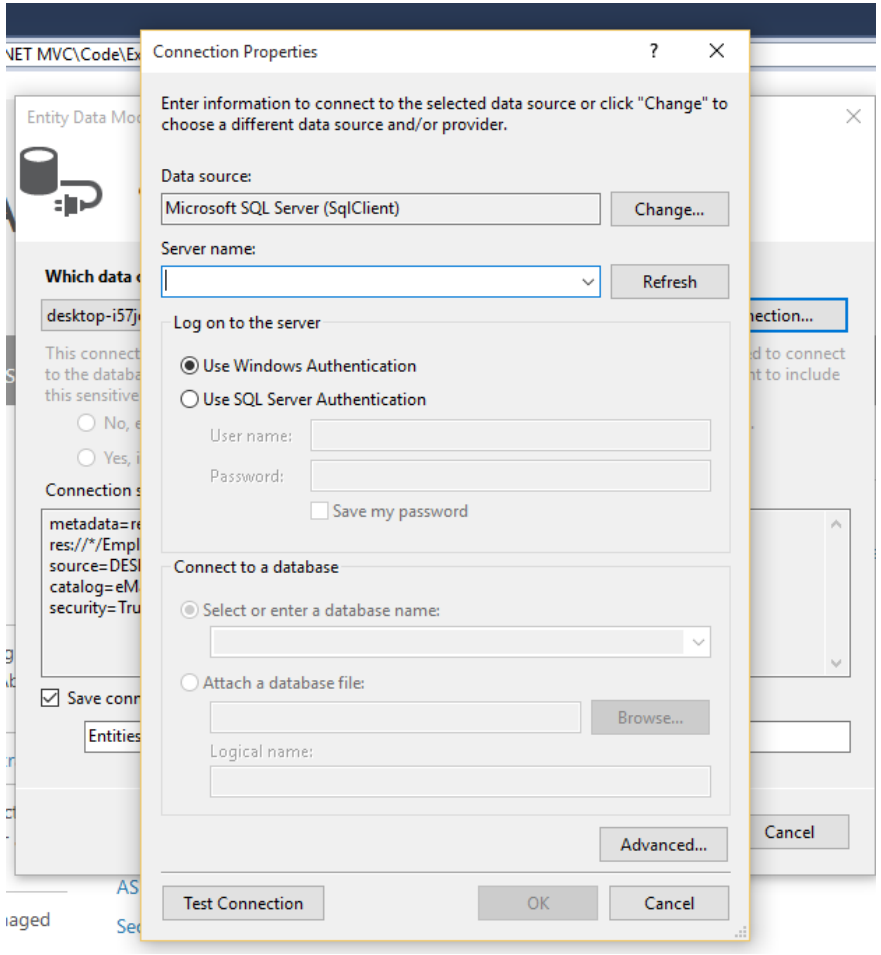
6.EF Designer from Database

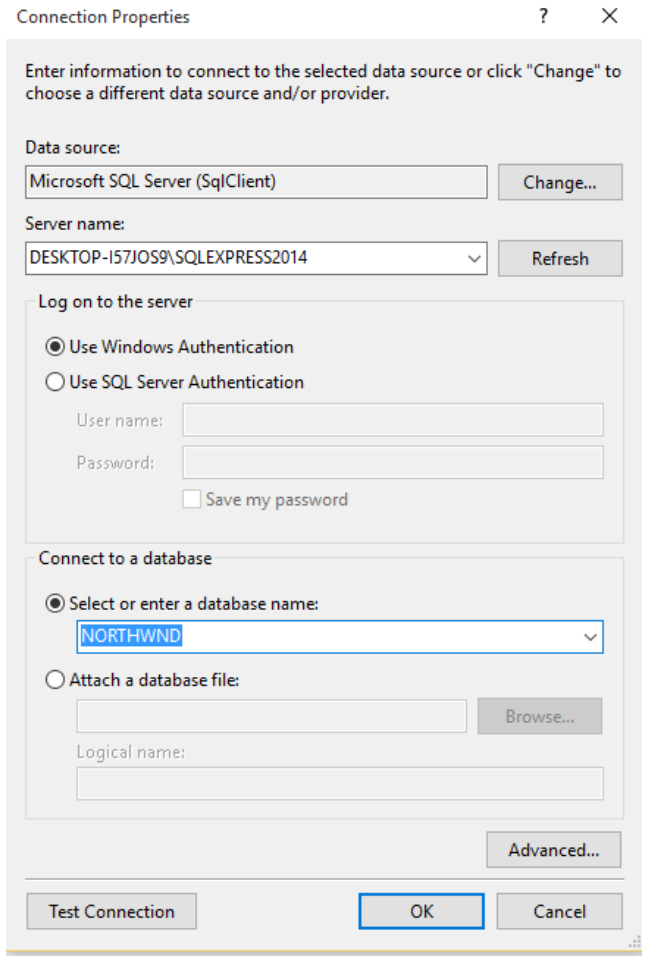


7.DATA CONNECTION

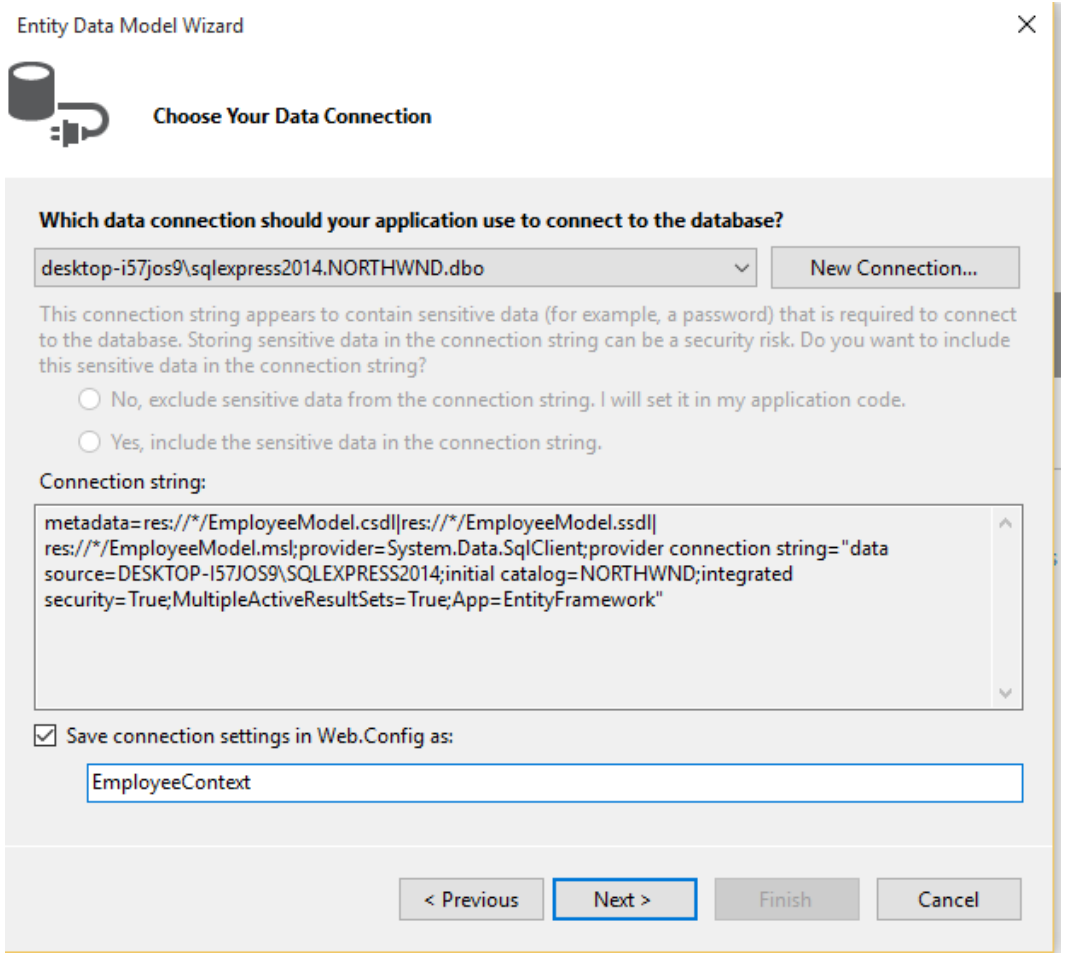


8. DEFINE CONNECTION PROPERTIES

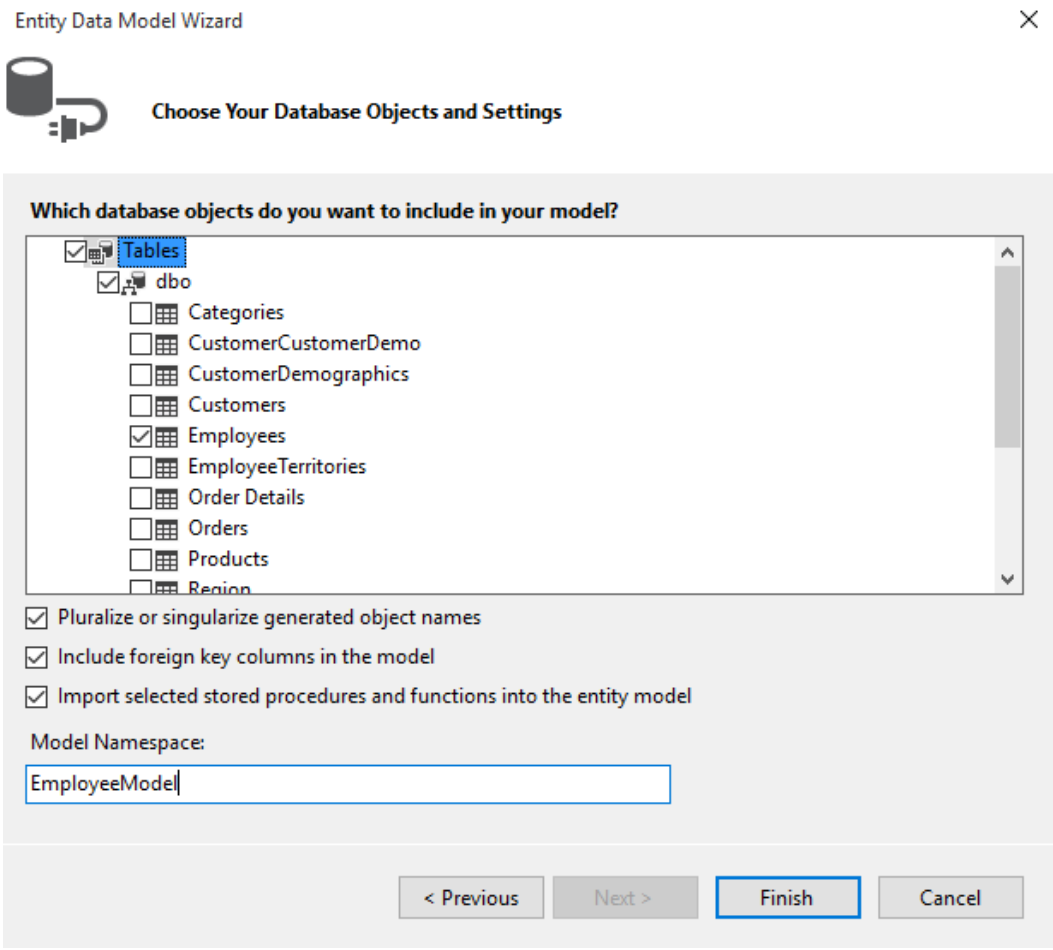




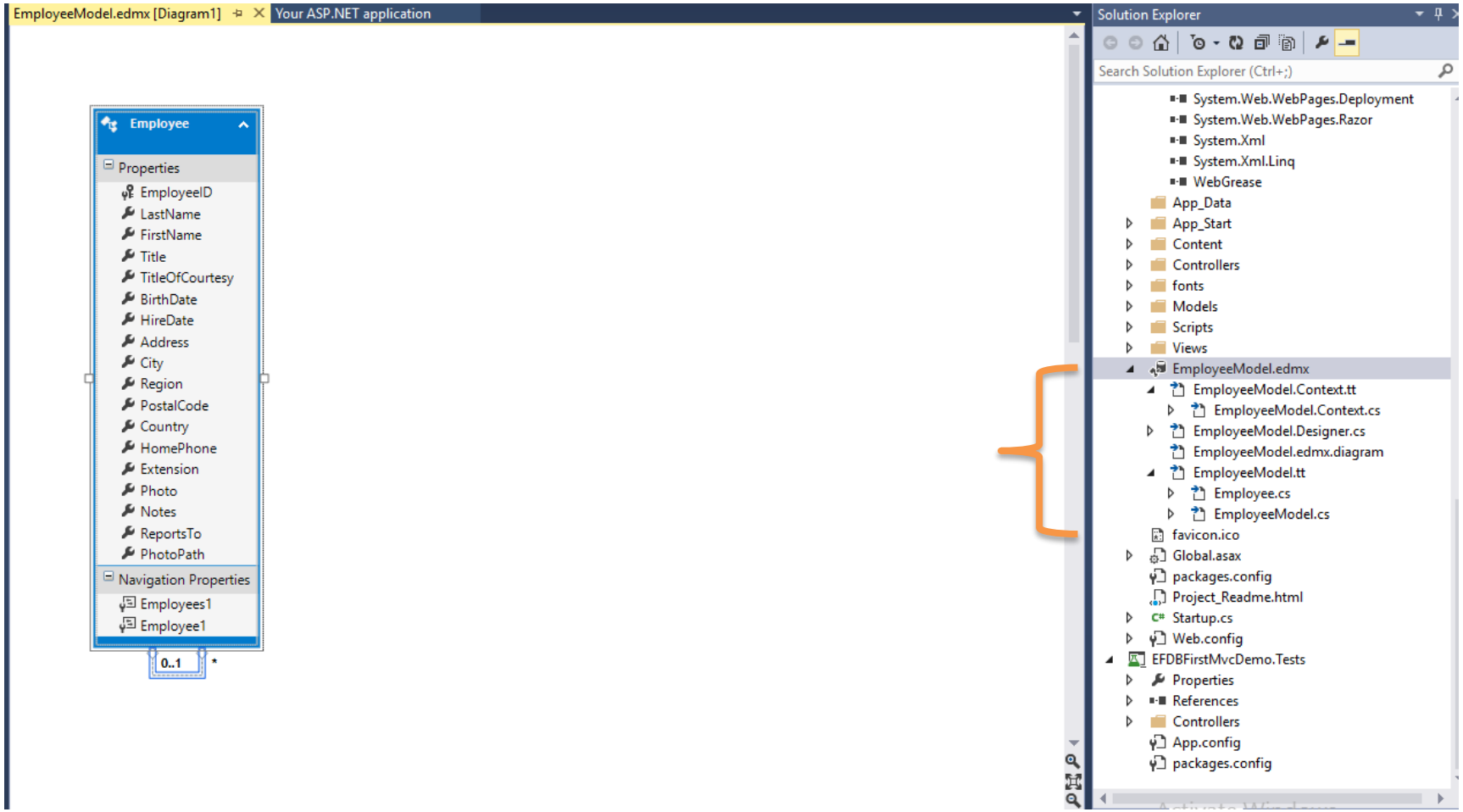
9. Post Connection Setting



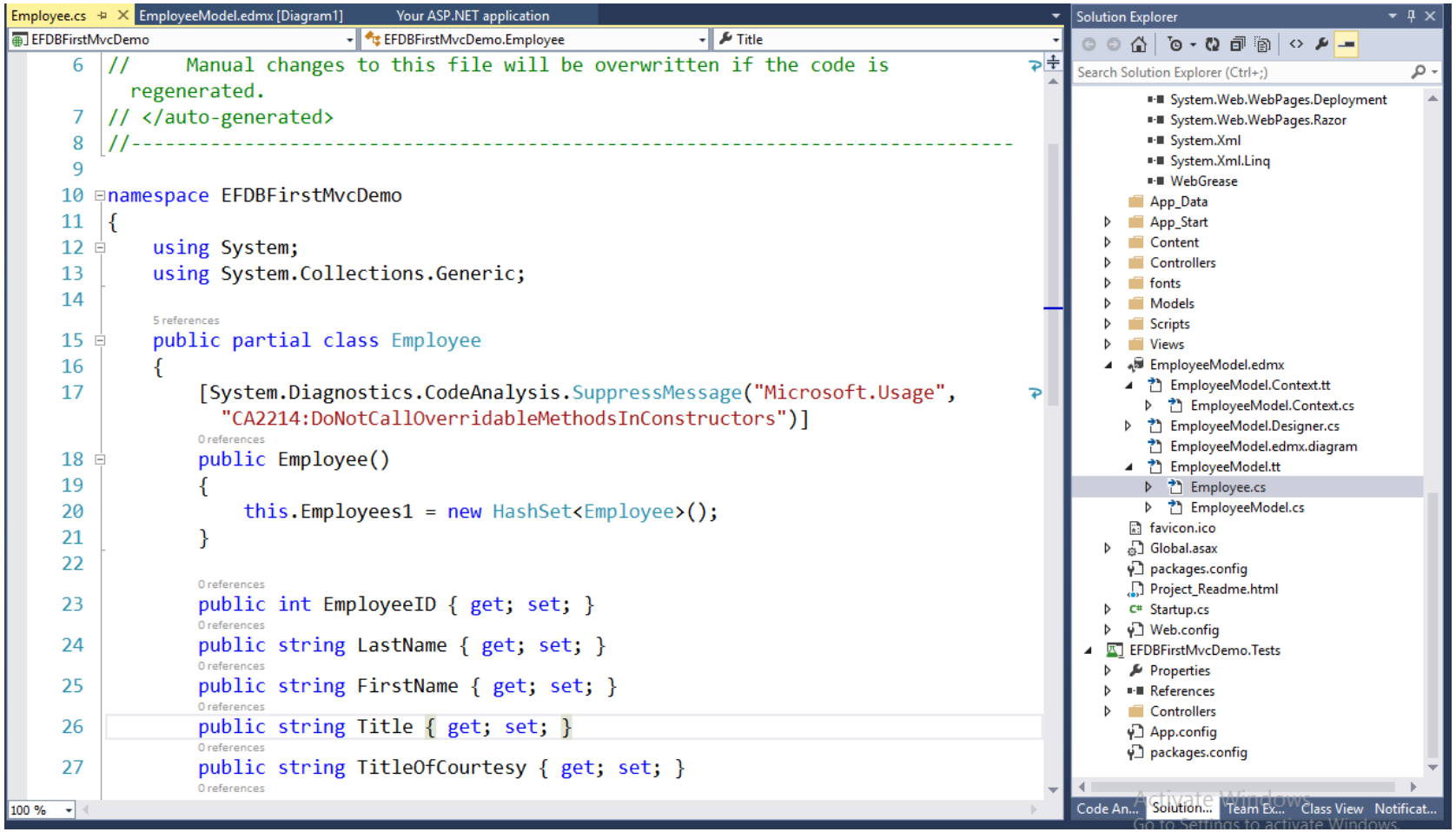
10.Choose Database Objects and Settings



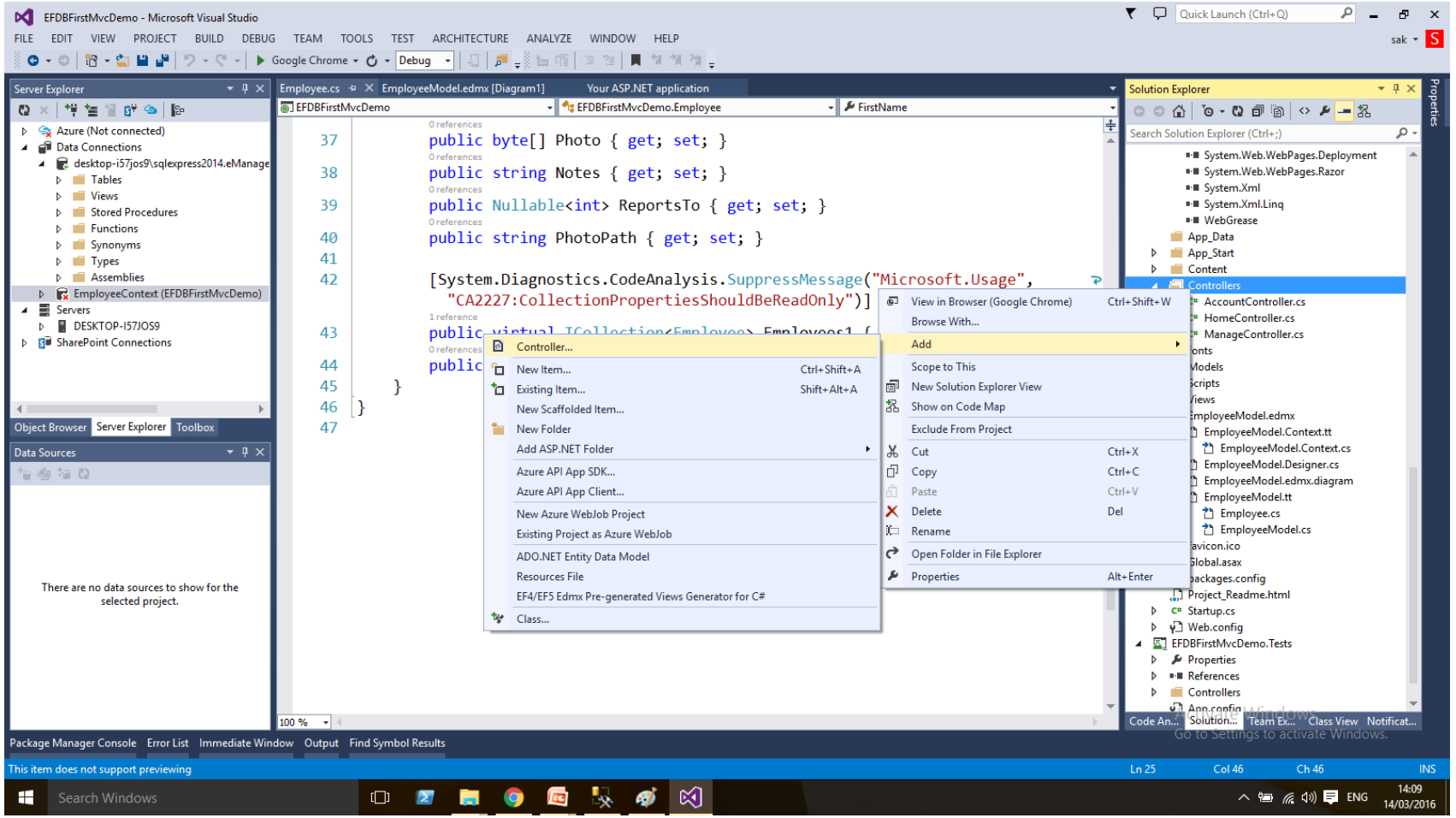
11. Successful Creation of Entity Model



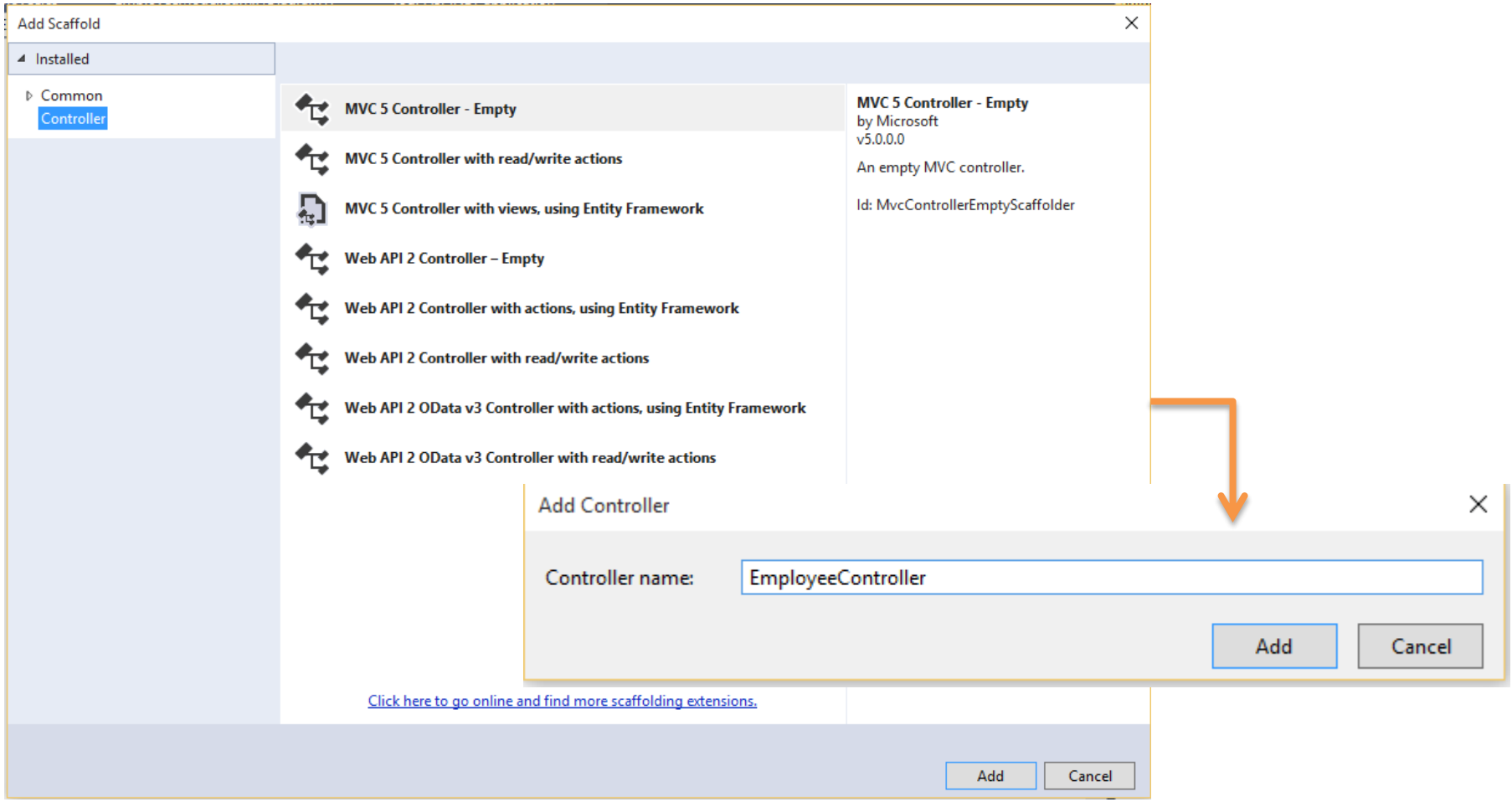
11. Successful Creation of Entity Model



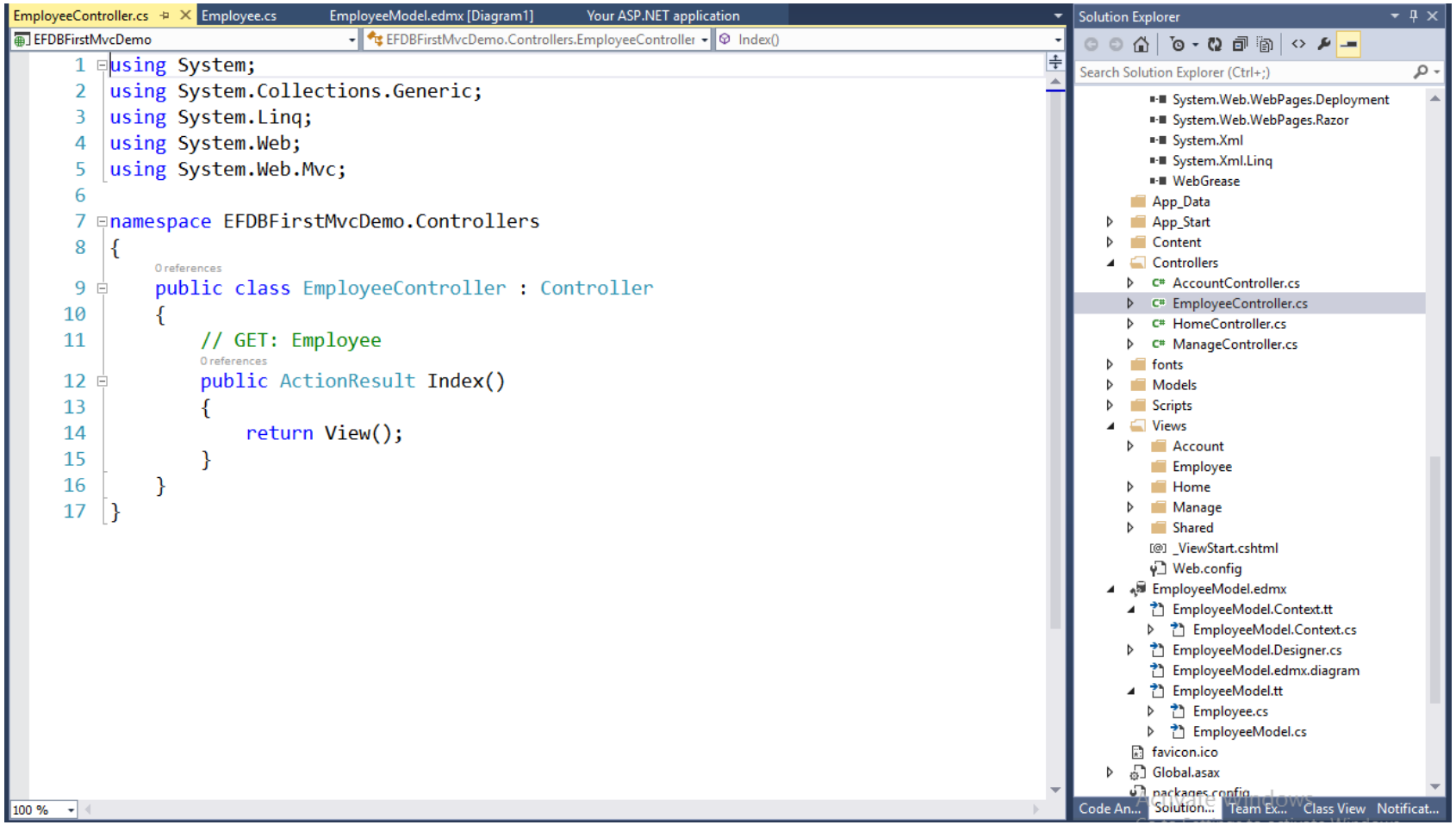
12.Create EmployeeController



13. Empty Controller



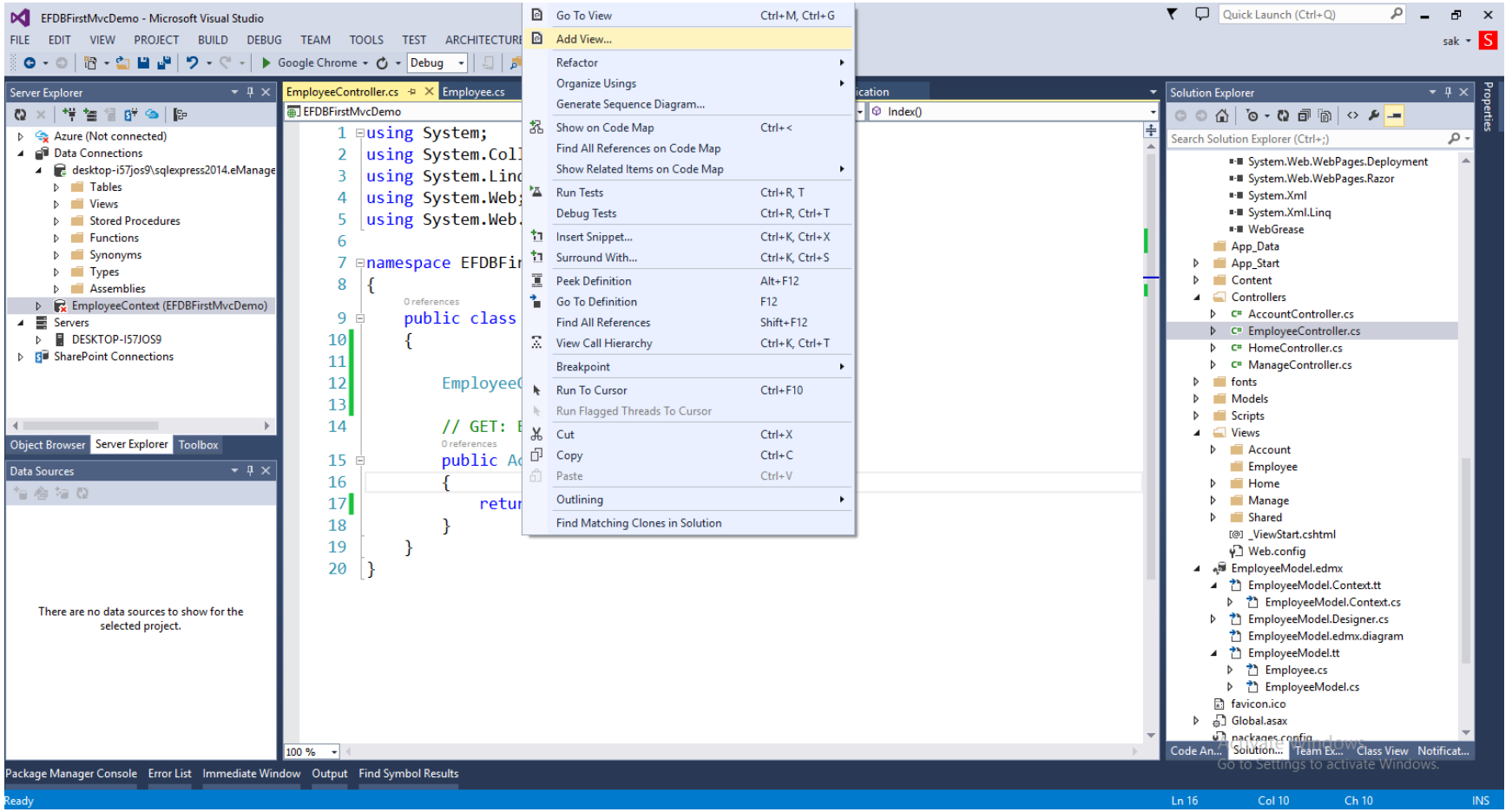
14.EmployeeController



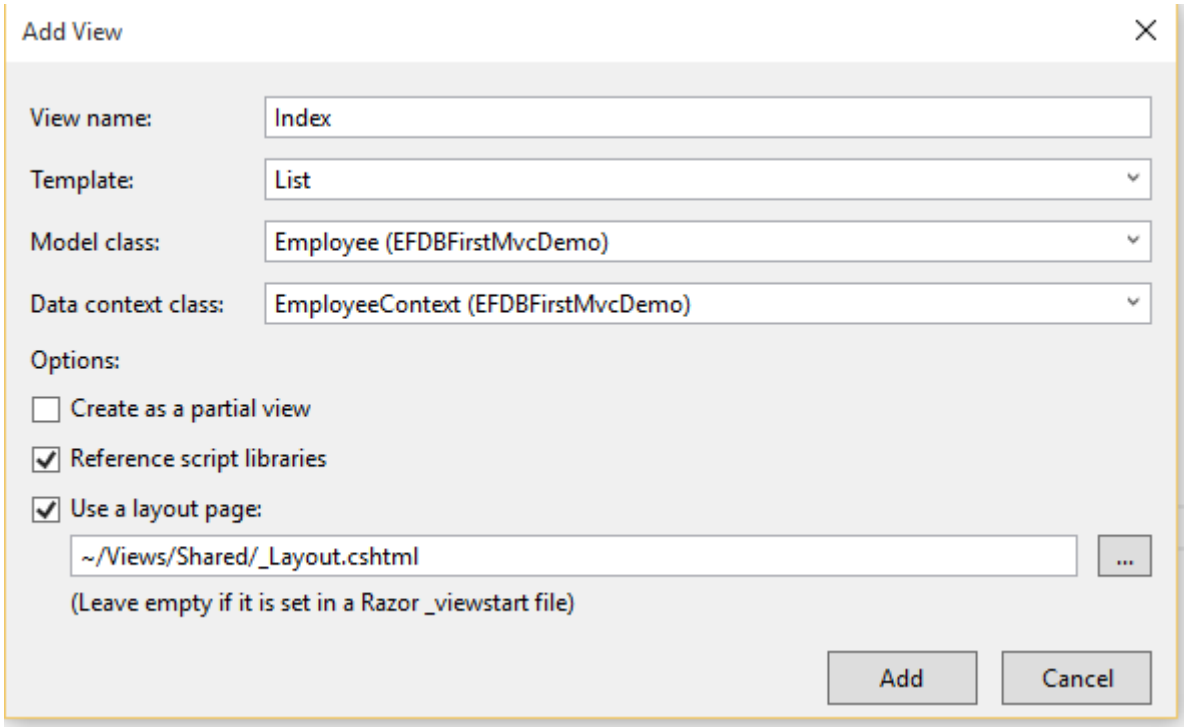
15. Retrieve the List of Employee in Controller


```
public class EmployeeController : Controller
{

    EmployeeContext db = new EmployeeContext();

    // GET: Employee
    0 references
    public ActionResult Index()
    {
        return View(db.Employees.ToList());
    }
}
```

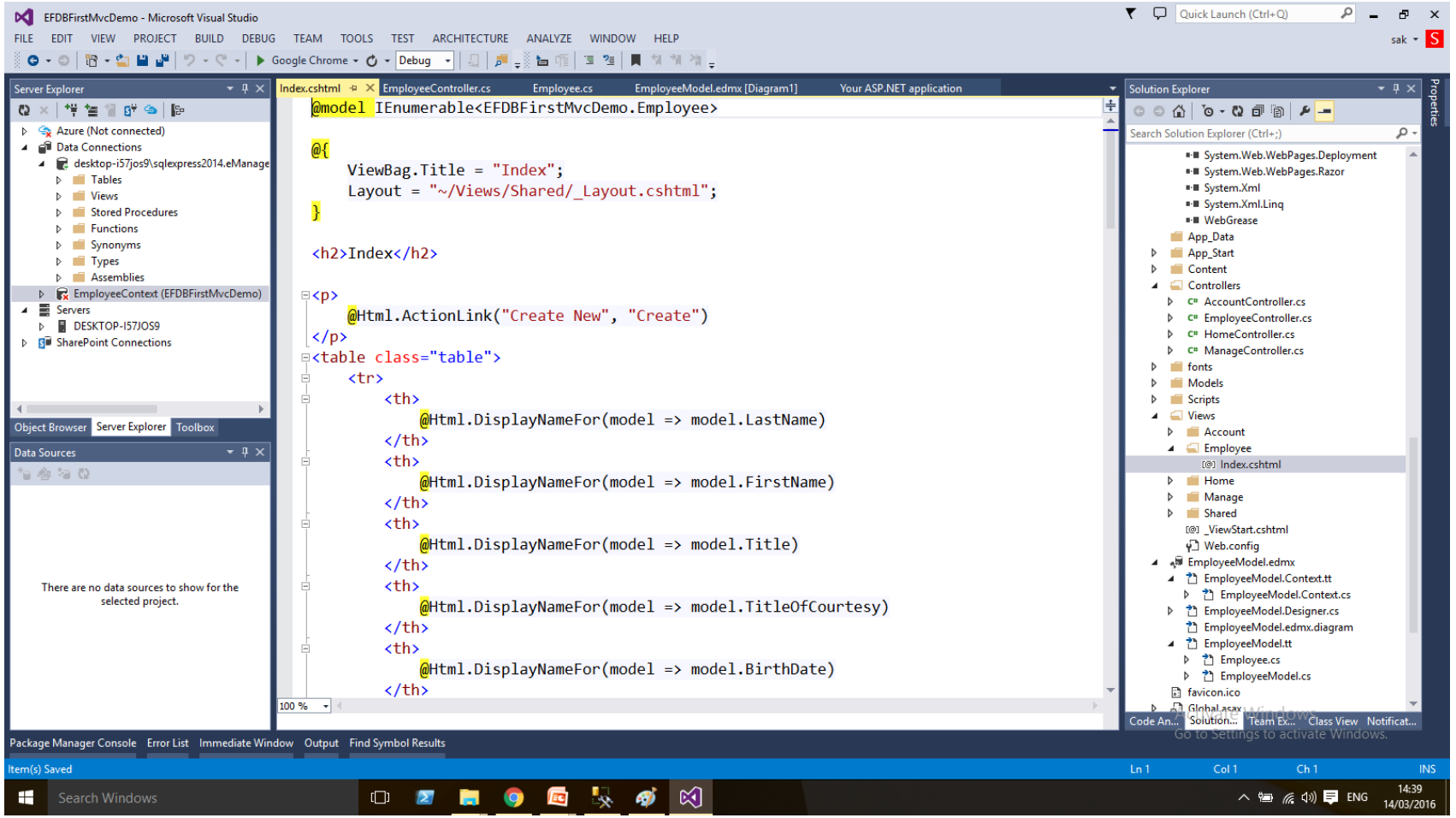
16. Generate View for Index



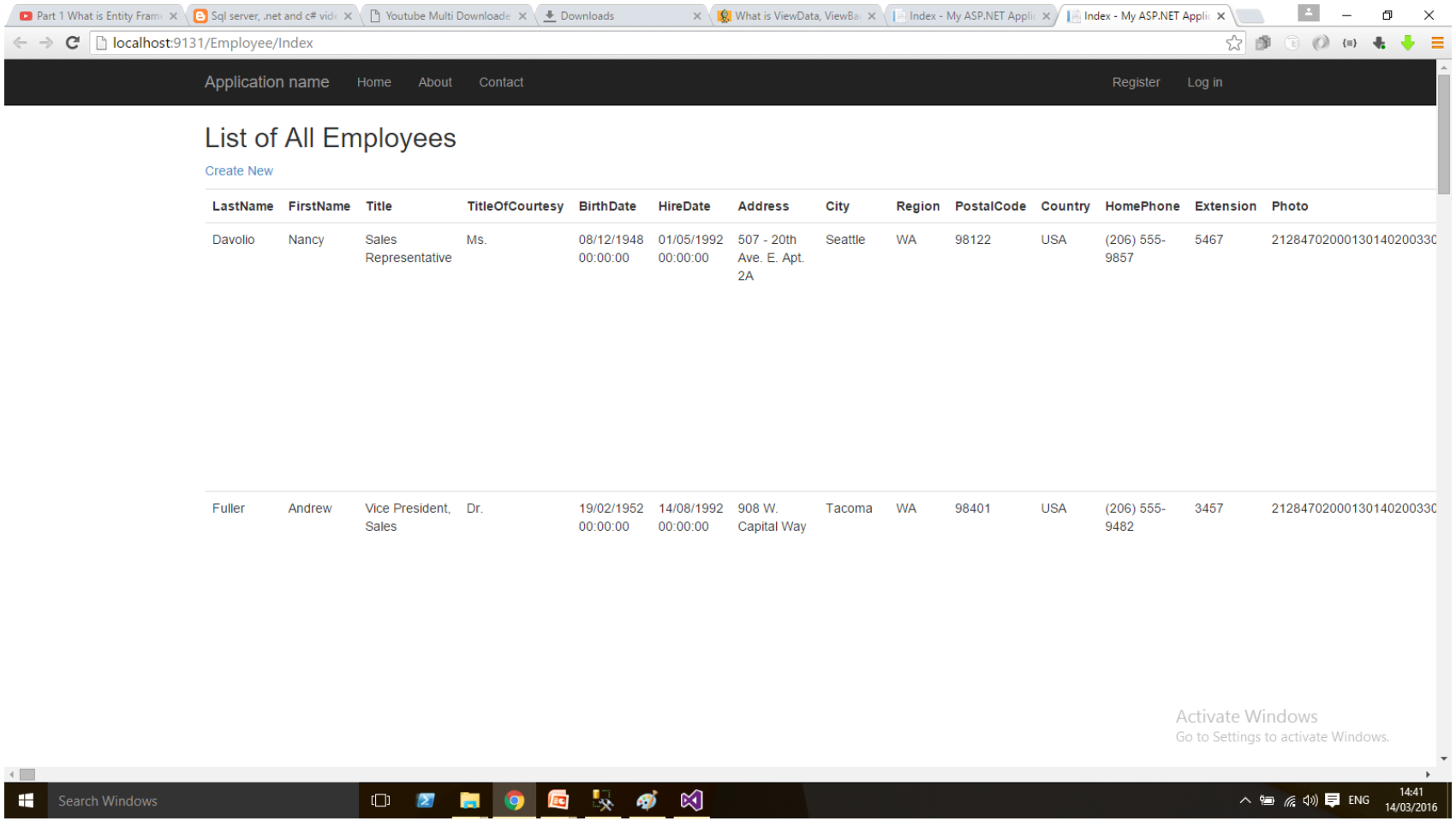
17. Index View



18. Index View Scaffolding Template



19. Result



ModelState

- ModelState is a property of a Controller, and can be accessed from those classes that inherit fromSystem.Web.Mvc.Controller.
- The ModelState represents a collection of name and value pairs that were submitted to the server during a POST.
- It also contains a collection of error messages for each value submitted.
- Despite its name, it doesn't actually know anything about any model classes, it only has names, values, and errors.

- Serves 2 purposes
  - To store the value submitted to the server
  - To store the validation errors associated with those values
- ModelState represents the submitted values and errors in said values during a POST

DataAnnotations

- Data Validation is a key aspect for developing web application. In ASP.NET MVC, we can apply validation to web application by using **Data Annotation Attribute classes to model class.**
- **Data Annotation attribute classes are present in System.ComponentModel.DataAnnotations namespace.**
- They help us to define rules to the model classes or properties for data validation and displaying suitable message to end users.

Annotation Type	Description
Data Type	Specify the datatype of a property
DisplayName	Specify the display name for a property
DisplayFormat	Specify the display format for a property like different format for Date property
Required	Specify a property as required
RegularExpression	Validate the value of a property by specified regular expression pattern
Range	Validate the value of a property within a specified range of values
StringLength	Specify min and max length for a string property
MaxLength	Specify max length for a string property
Bind	Specify fields to include or exclude when adding parameter or form values to model properties
ScaffoldColumn	Specify fields for hiding from editor forms

Data Annotations

Advantages

- Enable you to perform validation simply by adding one or more attributes – such as required or StringLength attribute – to a class property

Update Model vs TryUpdateModel

UpdateModel()

- Throws an exception if validation fails
- Used to update the model with the form values and perform the validations

TryUpdateModel()

- Never throws an exception if a validation fails. But will redirect to the same page with error/validation error message displayed.
- Used to update the model with the form values and perform the validations

Original Series

MVC5 WITH EF CODE FIRST

Code First EF MVC5 Application

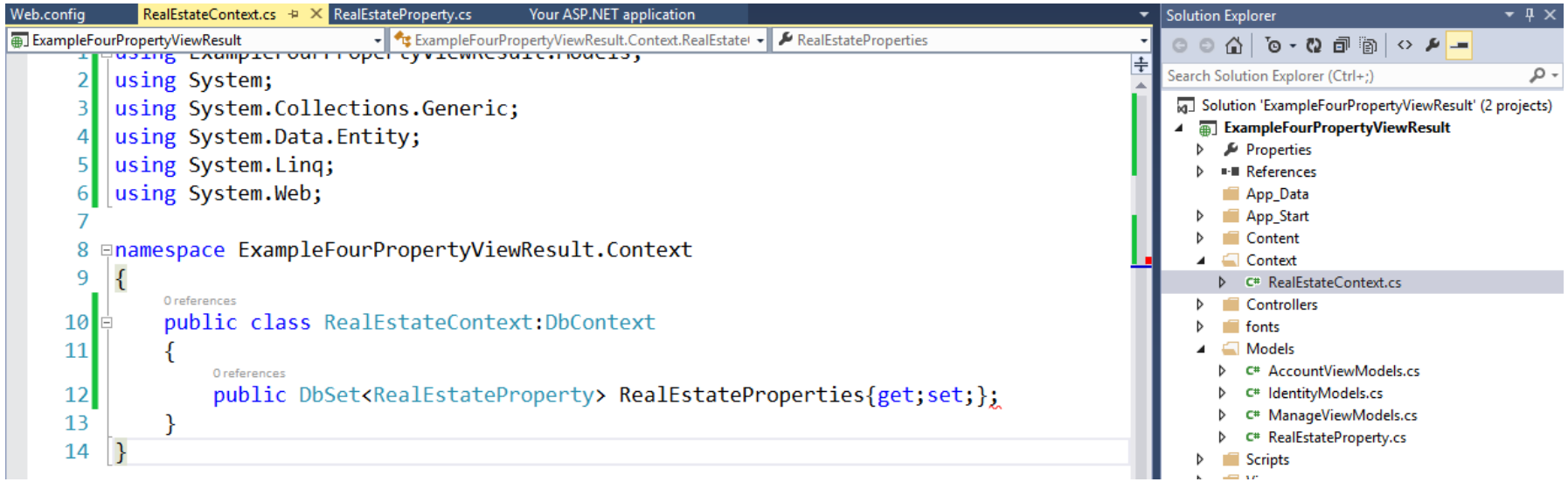
Step 1. Create MVC5 Application

Step 2. Create A folder Context

Step 3. Create a Model – ModelName.cs

```
public class RealEstateProperty
{
    [Key]
    0 references
    public int EstatePropertyId { get; set; }

    [Required]
    [DisplayName("Natural Address Code 30 digit Alphanumeric code")]
    0 references
    public string  NACCode { get; set; }

    0 references
    public string EstatePropertyName { get; set; }


    0 references
    public string OwnerName { get; set; }
    0 references
    public string City { get; set; }

    0 references
    public string Country { get; set; }

    0 references
    public string Zip{get;set;}
```

- Step 4. Create a ModelContext class in Context Folder – ModelContext.cs



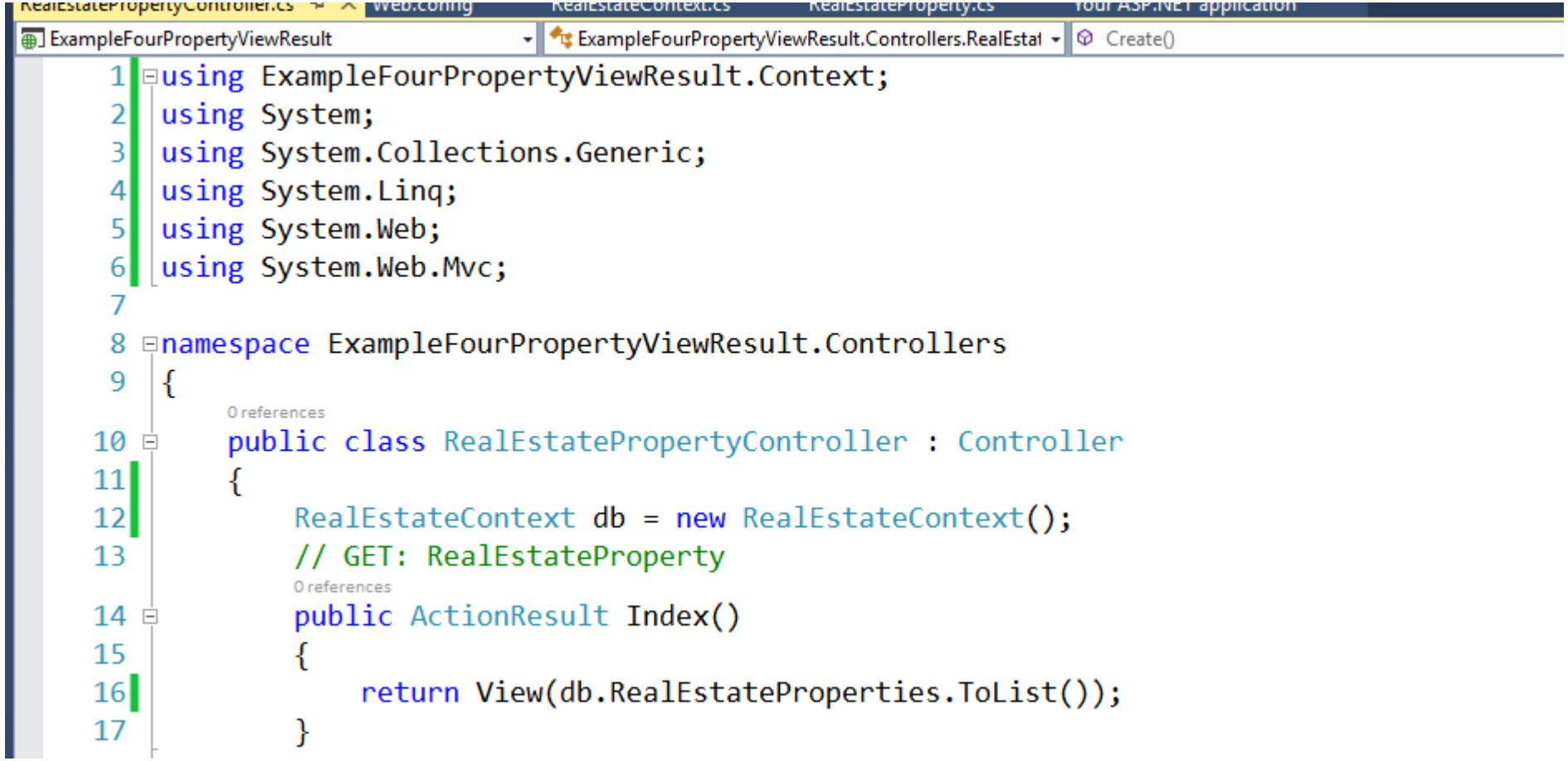
- Step 5. Add ConnectionString in web.config

```
<connectionStrings>
  <add name="DefaultConnection" connectionString="Data Source=(LocalDb)\v11.0;AttachDb
    providerName="System.Data.SqlClient" />
    <add name="RealEstateContext" connectionString="Data Source=(LocalDb)\v11.0;Attach
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

- Step 6. Create a Controller

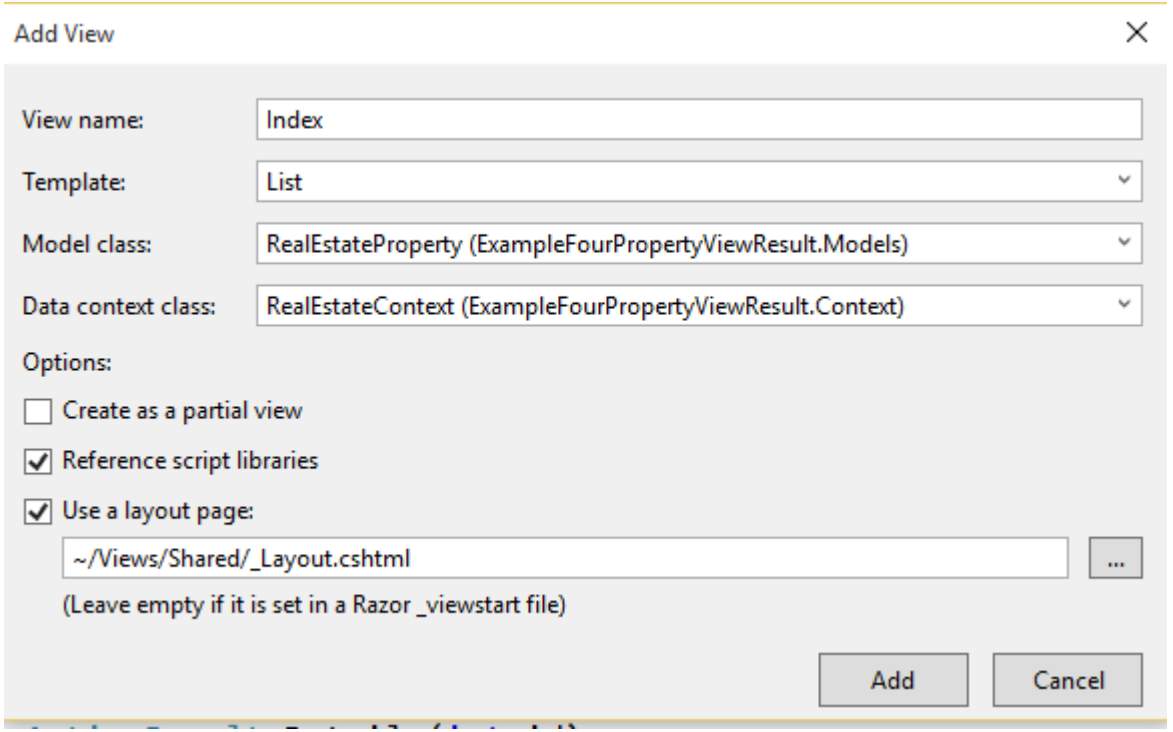
```
namespace ExampleFourPropertyViewResult.Controllers
{
    0 references
    public class RealEstatePropertyController : Controller
    {

        // GET: RealEstateProperty
        0 references
        public ActionResult Index()
        {
            return View();
        }
```

- Step 7. Add DbContext to the Controller and return list to the conventional view



- Step 8. Build the application with ctrl+shift+B

- Step 9. Generate Index View



- Step 10. Run the Application – ctrl +f5

- Step 11. Upon first access to the Controller Action Method, We have a DB generated from **the RealEstateProperty Model**

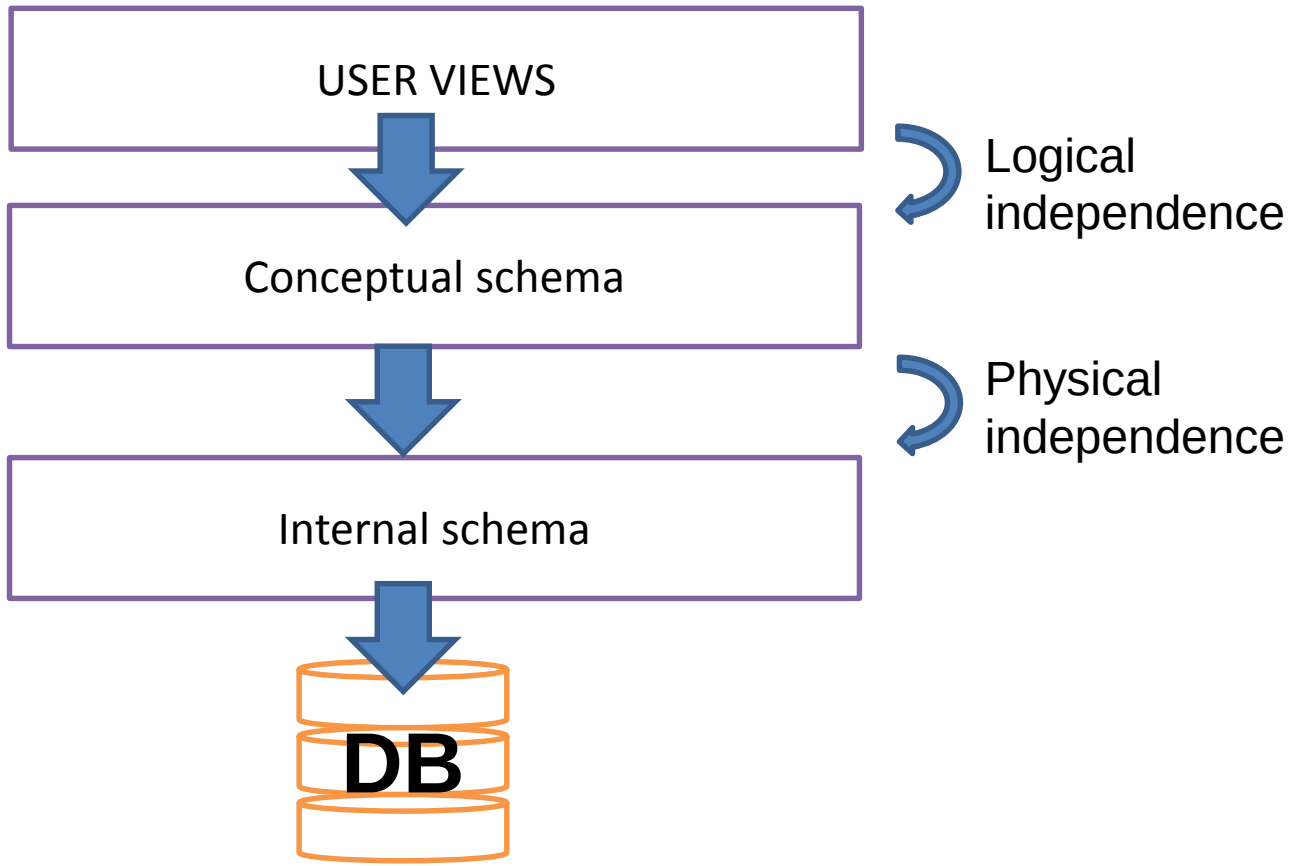
Summary

- Demonstrated how to create EF DB First and Code First approaches
- Demonstrated the use of data annotations for validations

SECTION -XV

N-TIER ARCHITECTURE

Architecture



- Logical independence: the ability to change the logical schema without changing the external schema or application programs.
  - Can add new fields, table without changing views.
  - Can change the structure of the table without changing the view.
- Physical Independence: the ability to change the physical schema without changing the logical schema
  - Storage space can change
  - Types of some data can change for optimizing the application

N-Tier Architectures

- Have
  - Presentation Layer
  - Business/Logic Layer
  - Data Access Layer
  - N-tier architectures try to separate the components into different tiers/layers
    - Tier : physical separation
    - Layer:logical separation

Repository Design Pattern

Martin Fowler: "*An Repository Design Pattern mediates between the domain and data mapping layers, acting like an in-memory collection of domain objects*"

SECTION -XV

WEB API 2.2

DESIRE FOR FAST, LIGHTWEIGHT FRAMEWORKS AND SERVERS

OPEN WEB INTERFACE FOR .NET

SECTION -XVI

OWIN: MIDDLEWARE/KATANA

- OWIN + KATANA
- PORTABLE
- MODULAR
- LIGHTWEIGHT

Open Web Interface for .Net

- The goal of OWIN is to decouple server and application and, by being an open standard, stimulate the open source ecosystem of .NET web development tools.

www.owin.org

Motivation for OWIN

- System.Web basis for All Web
- WebForms bundled
- Regression bugs (fix old bugs, and open new ones)
- Monolithic Architectures
- Slow release cycle
- Performance
- IIS (performance is better as long as we do not load System.Web*)
- Evolutionary steps: MVC, WEBAPI

- Moving from big server application to smaller client/device centric applications, microservices architectures, mobile clients, REST.
- Cloud, Docker, cross-platform
- Web based IDE, lightweight IDEs(sublime text, Atom)
- ASP.NET coupled to IIS
- Hard to do REST
- ASP.NET --- toooo heavy
- Drawn inspiration from Rack(Ruby) and WSGI(Python)

Solutions

- Boat
- Figment
- Fracture
- Frank
- FubuMVC
- KayakHTTP
- NancyFx
- Nina
- Nrack
- OpenRasta
- Suave
- WCF Web API
- Etc..

ASP.NET Evolution

- ASP.NET – released in early 2002 with .NET framework 1.0
- Optimized for 2 primary customers
  - Classis ASP developers
  - Desktop line of business developers (VB6 WINFORMS Developers)
- System.Web.dll (aka ASP.NET)
  - Considered heavy
  - Always executes lots of ASP.NET – specific code
- Result was a monolithic framework that included lots of concerns in a single package(System.Web)
  - HTTP intrinsics
  - Modules
  - Handlers
  - Session
  - Cache
  - Web Forms
  - Controls
- ASP.NET was designed to be run on IIS

Challenges

- ASP.NET shipped as a apart of the .NET Framework – slow ship cycle
- As ASP.NET **evolved -> the size and complexity of System.Web also grew.**
- All of the features were turned on by default – you have to opt out
- Only one hosting option –IIS
- Performance Cost and Processing Costs increase.

- 2007-2008 ASP.NET MVC ships out of band from the .NET Framework
- Enables a more rapid ship cycle
- http://asp.net/mvc
- 2012 – ASP.NET WebAPI released
  - Designed with no dependencies on System.Web
  - Includes self-host capability
- **Self-host realization**
  - ASP.NET Web API enabled self host for Web API applications
  - Additionally, other frameworks had or were writing their own self-host servers
  - A modern web application needs the ability to compose multiple frameworks in a single server.

BareBones Http Server in NodeJS

- Basic http service written in Node.JS
- Full control on the content types and negotiation

```
var http= require('http');
http.createServer(function(req,res){
    res.writeHead(200,{'Content-Type':'text/plain'});
    res.end("Hello Harman\n");
}).listen(9999,'127.0.0.1');
console.log('Server running at http://127.0.0.1:9999');
```

OWIN

- Open Web Interface for dot Net
- Defines interface between app components
  - Decouples app from framework, host and server
- Open standard
- Not revolution but evolution, influenced by other stacks

- A BAREBONES HTTP STACK FOR .NET FRAMEWORK

OWIN

- Specification for Open Web Interface for .Net
- Community standard
- No more System.Web, just dictionary of environment variables (request, response, etc)
- Async
- Microsofts implementation: Katana
  - v1-3 is in ASP.NET vCurrent
  - v4 is vNext
  - MVC 6, WebAPI, SignalR
- Helios, Kestrel, Nowin
- IIS and OS
  - Realease with the OS
  - released with new version of Windows (WebSockets are available only on the latest version of Windows)
- OWIN/Katana Support
  - Many application frameworks support OWIN/Katana
  - Web API
  - SignalR
  - Nancy
  - ServiceStack
  - FubuMVC
  - Simple.Web
  - RavenDB

OWIN Adoption and Support

- Adoption
  - Microsoft.OWIN also called as Katana
  - Fix
  - Freya
  - SimpleOWIN
  - Simple.OWIN
  - Suave
  - WebSharper
- Server Support
  - System.Web
  - HttpListener
  - IIS(Helios)
  - Nowin(cross-platform)
  - Suave(cross-platform)
- Middleware
  - CORS
  - Security
  - Routing(Superscribe)
  - Diagnostics

OWIN Specs:AppFunc


```
using AppFunc = func<IDictionary<string,object>,Task>;
```

```
using AppFunc = func<
    IDictionary<string,object>,//Environment
    Task>; //Done
```

OWIN Specs:Environment

- Keys in the dictionary (8 request, 5 response, 2 others)
  - Owin.RequestBody =>Stream
  - Owin.RequestHeaders =>Idictionary<string,string[]>
  - Owin.Request*
  - Owin.ResponseBody =>Stream
  - Owin.ResponseHeaders =>Idictionary<string,string[]>
  - Owin.ResponseStatusCode =>int
  - Owin.Response*
  - Owin.CallCancelled =>CancellationToken

OWIN Request Keys

Owin.RequestBody	Stream
Owin.RequestHeaders	IDictionary<string,string[]>
Owin.RequestMethod	String
Owin.RequestPath	String
Owin.RequestPathBase	String
Owin.RequestProtocol	String
Owin.RequestQueryString	String
Owin.RequestScheme	String
Owin.RequestId*	Optional String
Owin.RequestUser*	Optional ClaimsPrincipal

OWIN Response Keys

Owin.ResponseBody	Stream
Owin.ResponseHeaders	IDictionary<string,string[]>
Owin.ResponseStatusCode	Optional int(default is 200)
Owin.ResponseReasonPhrase	Optional string(default set by server)
Owin.ResponseProtoco	Optional String

OWIN Other Data

Owin.CallCancelled	CancellationToken
Owin.Version	String

OWIN Specs:Layers

Host

- Startup, Initialization and process management

Server

- Listens to socket and calls the first middle ware

Middleware

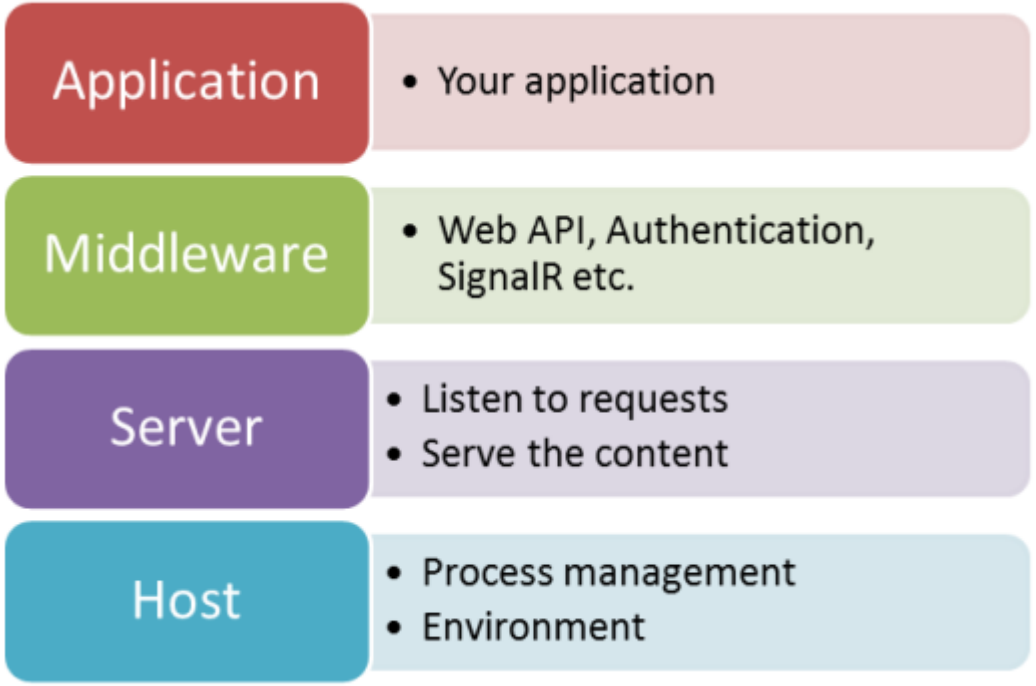
- Pass-through components

Application

- Application code

Katana

- Microsoft's OWIN implementation
  - https://katanaproject.codeplex.com
- Set of hosts and servers
  - IIS or self-hosting
- Set of convenience classes
  - OwinContext, OwinRequest,OwinResponse,etc
  - AppBuilderUseExtensions
  - AuthenticationManager
- Set of middleware for common features
  - Authentication
  - Hosting content(e.g.static files)
  - CORS

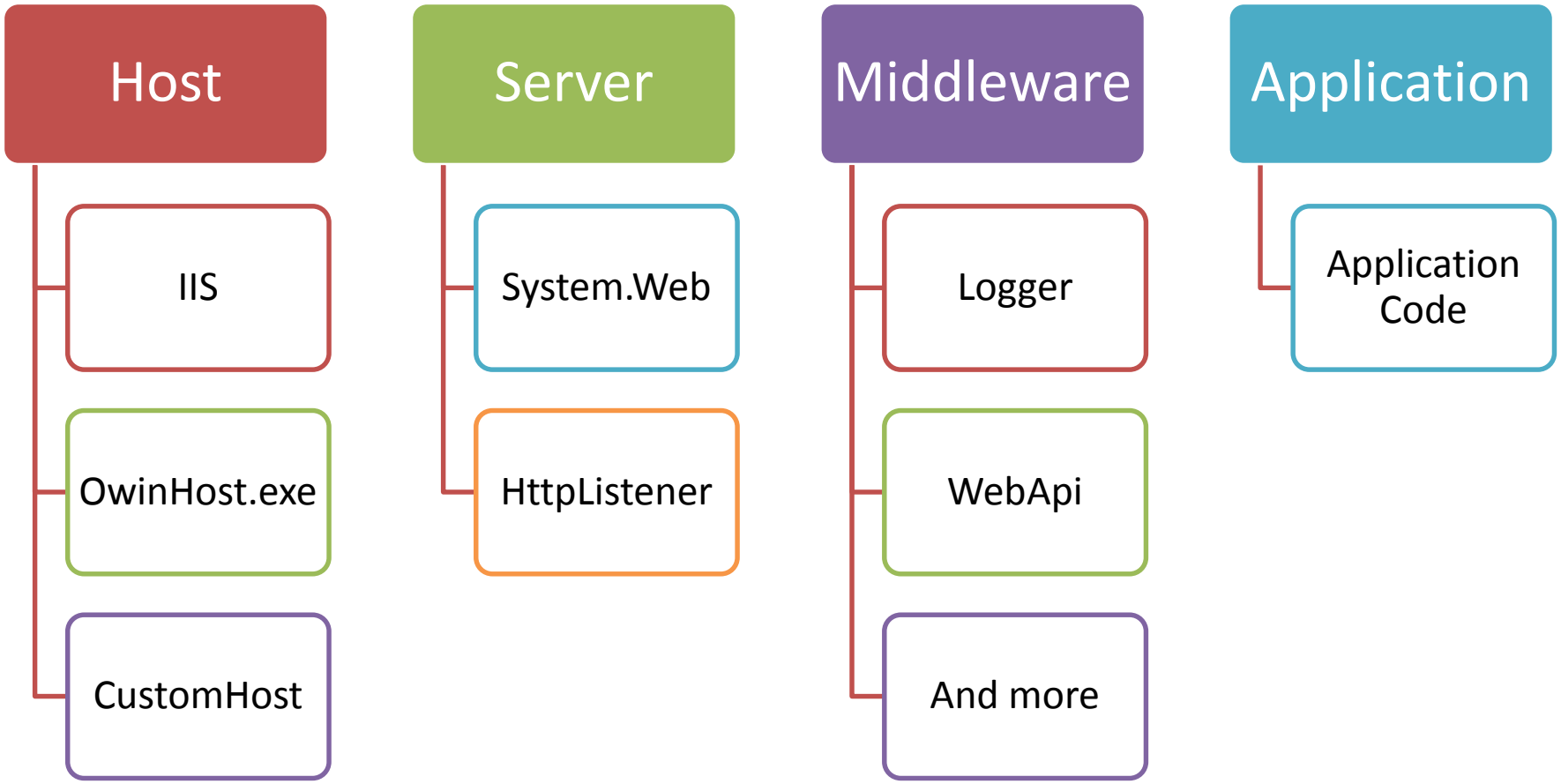


Removing the dependency on IIS

Katana pillars

- It's portable
  - Components should be able to easily substituted for new components as they become available
  - This includes all types of components, from the framework to the server and host
- Lightweight/performant/scalable
  - Fewer computing resources
  - Explicit decision to the application developer to include on-demand features from the underlying infrastructure to the OWIN pipeline.
- It's Modular/flexible
  - Unlike many frameworks which include a myriad of features that are turned on by default, katana project components should be **small and focused**, giving control over to the application developer in determining which components to use in their applications

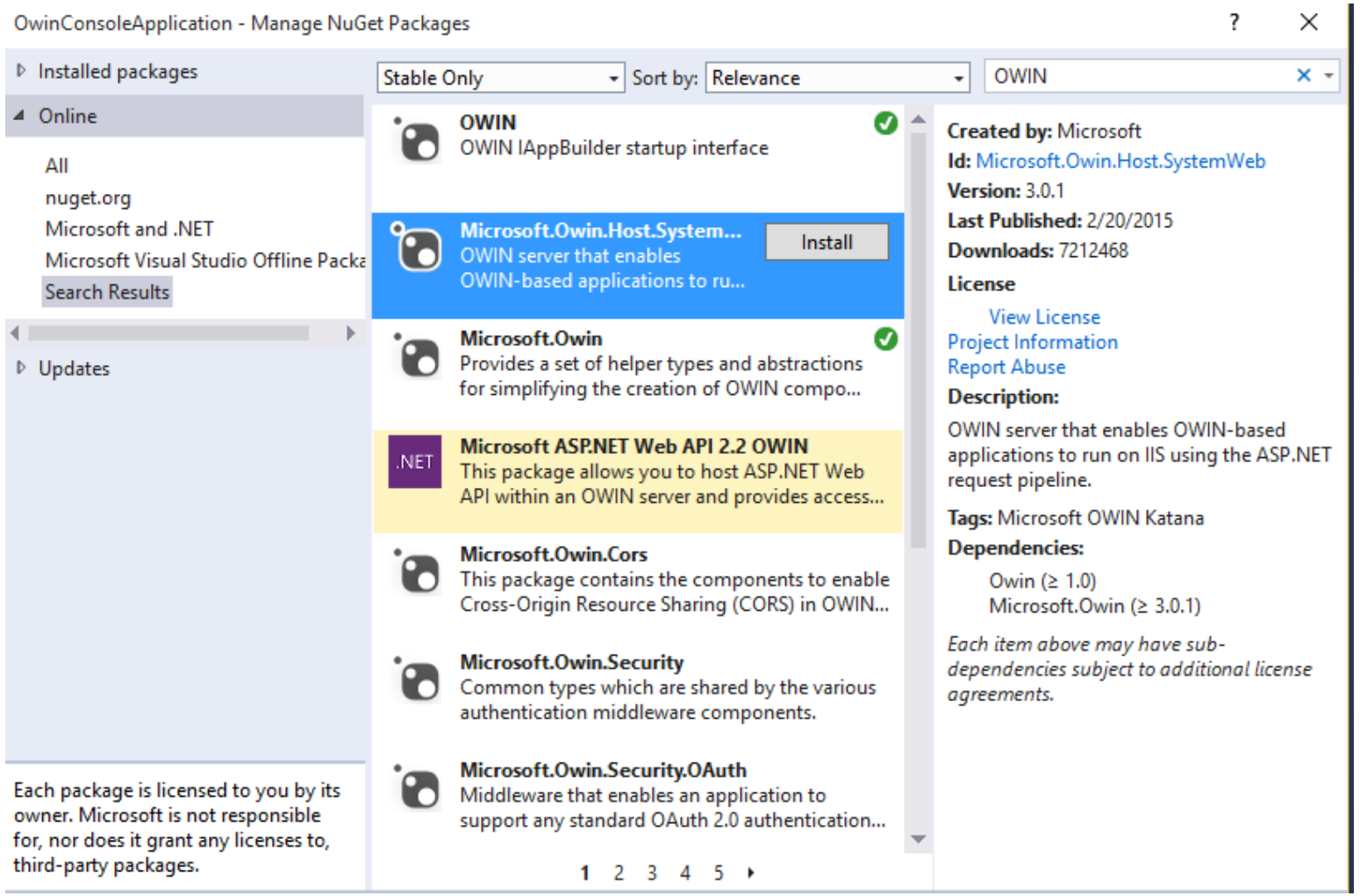
Katana Pipeline



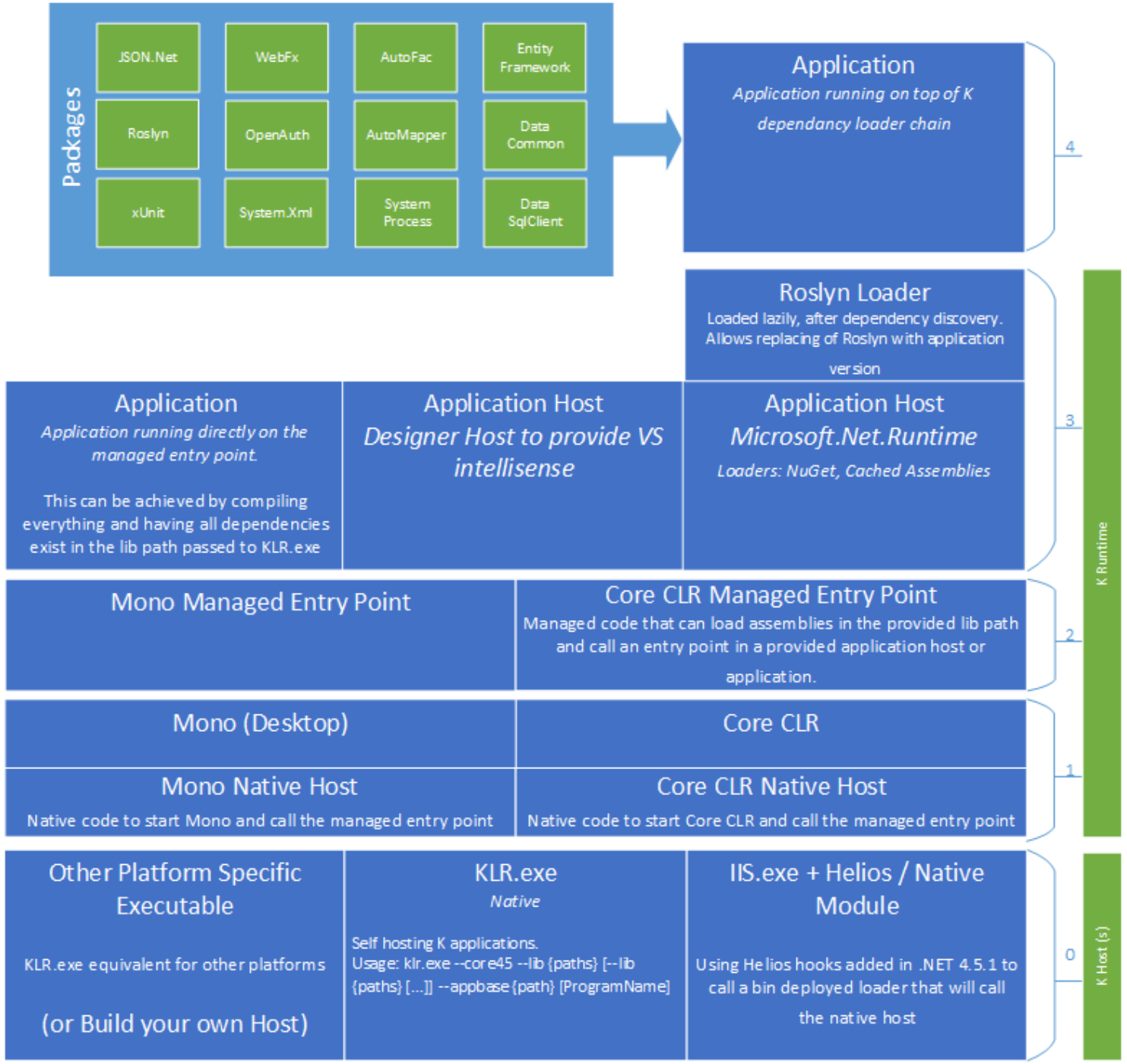
Microsoft ASP.Net Web API 2.2 OWIN

1. Create an empty web application and corresponding models and controllers.
2. Create OWIN startup.cs file
3. Create WebApiConfig.cs file
4. Import the Assembly references from Nuget Packages
    a. Microsoft ASP.Net Web Api 2.2 OWIN
    b. Microsoft.Owin
    c. Microsoft.Owin.Host.SystemWeb
    d. Microsoft.Owin.SelfHost
    e. OwinHost

Hosting on IIS



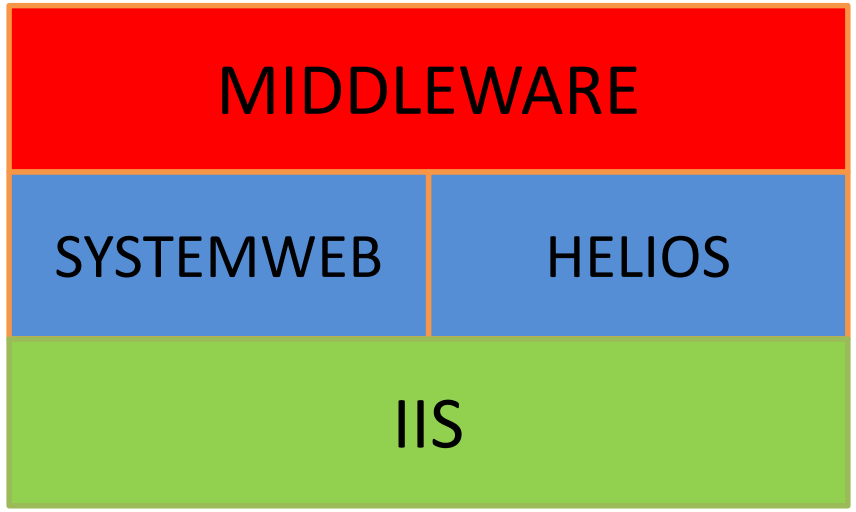
VNEXT



NOTE: KRE = XRE , K = DOTNET , KVM = DOTNET SDK, KPM = NUGET

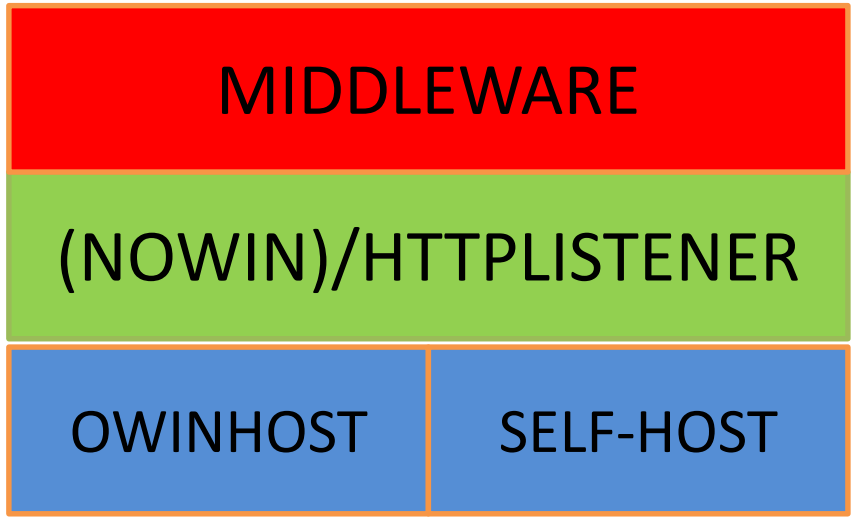
KATANA HOSTS AND SERVERS

IIS HOST



- IIS BASED HOSTING

NON-IIS HOST



FAST LIGHTWEIGHT CLR RE-WRITE

CROSS PLATFORM MODULAR

OPEN SOURCE

SINGLE STACK NUGET

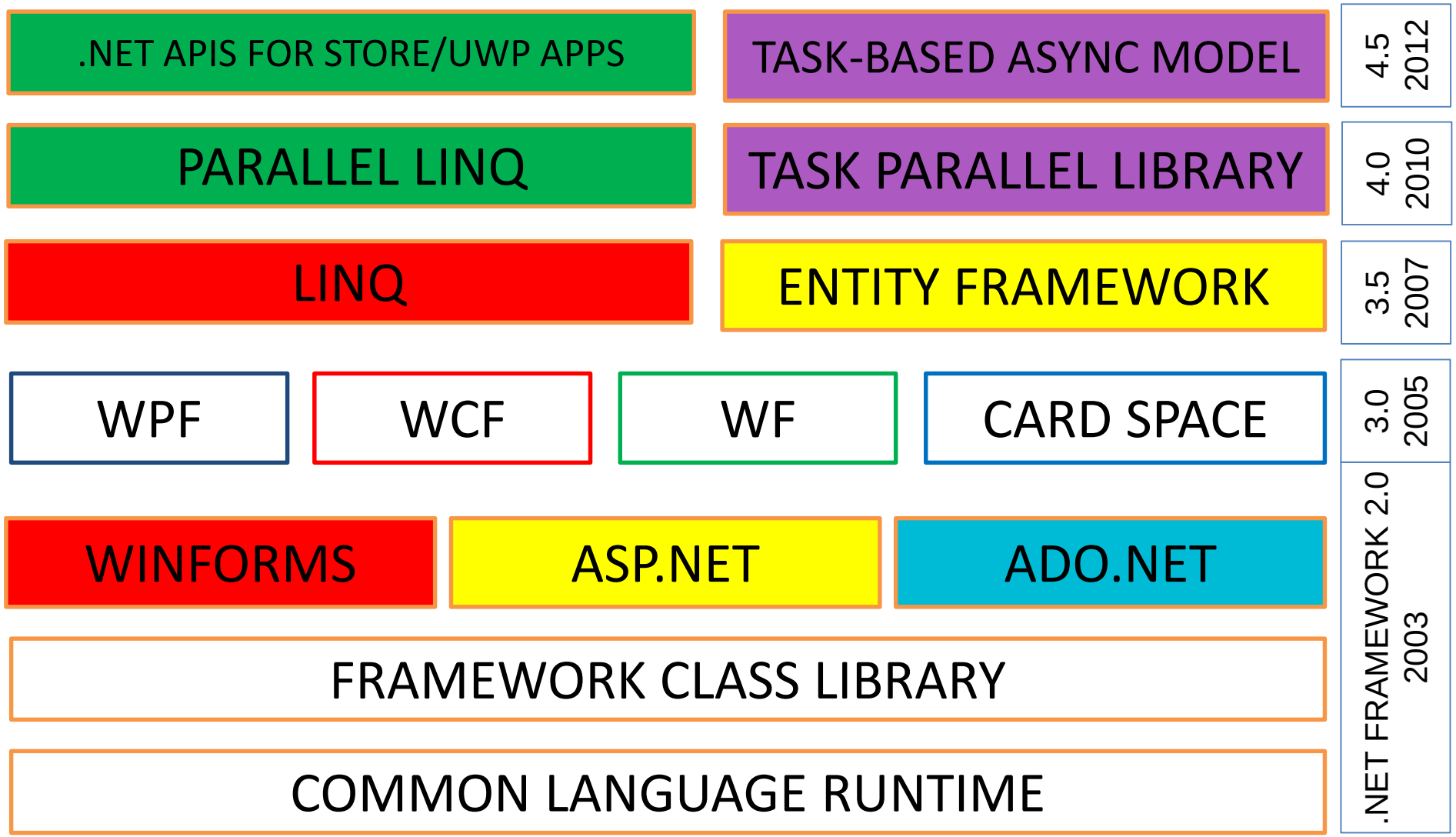
MULTIPLE RUNTIME VERSIONS

SECTION -XVII

.NET Core

https://dotnet.github.io

Any App, Any Platform, Any Developer



.NET RUNTIME

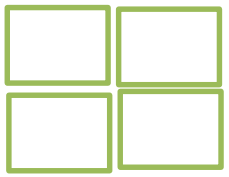
https://www.ecma-international.org/publications/standards/Ecma-335.htm

- COMMON LANGUAGE INFRASTRUCTURE (CLI)
  - FORMAL SPECIFICATION FOR .NET RUNTIME
  - ECMA 335
- SEVERAL DIFFERENT HISTORICAL IMPLEMENTATIONS
  - COMMON LANGUAGE RUNTIME
    - RUNTIME FOR THE .NET FRAMEWORK
    - MOST COMMONLY USED
- COMPACT FRAMEWORK
  - USED WITH MOBILE DEVICES
- SHARED SOURCE CLI(ROTOR)
- MONO (LINUX PLATFORMS)
- SILVERLIGHT
- WINDOWS PHONE
- XAMARIN (CROSS PLATFORM SUPPORT)

.NET CLASS LIBRARIES

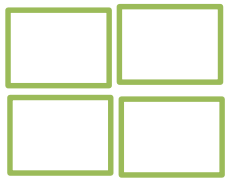
- FRAMEWORK CLASS LIBRARIES (FCL)
  - Libraries used by the .NET framework
- Silverlight
  - Deprecated subset of the FCL for building web and mobile Uis
- Portable class libraries (PCL)
  - Libraries for building cross platform applications
  - Target platforms
  - Windows
  - Windows Phone
  - Silverlight

.NET 2015



.NET FRAMEWORK

ASP.NET 5
ASP.NET 4.6
WPF
WINDOWS FORMS



.NET CORE

ASP.NET 5
.NET NATIVE

ASP.NET 5 FOR MAC
AND LINUX

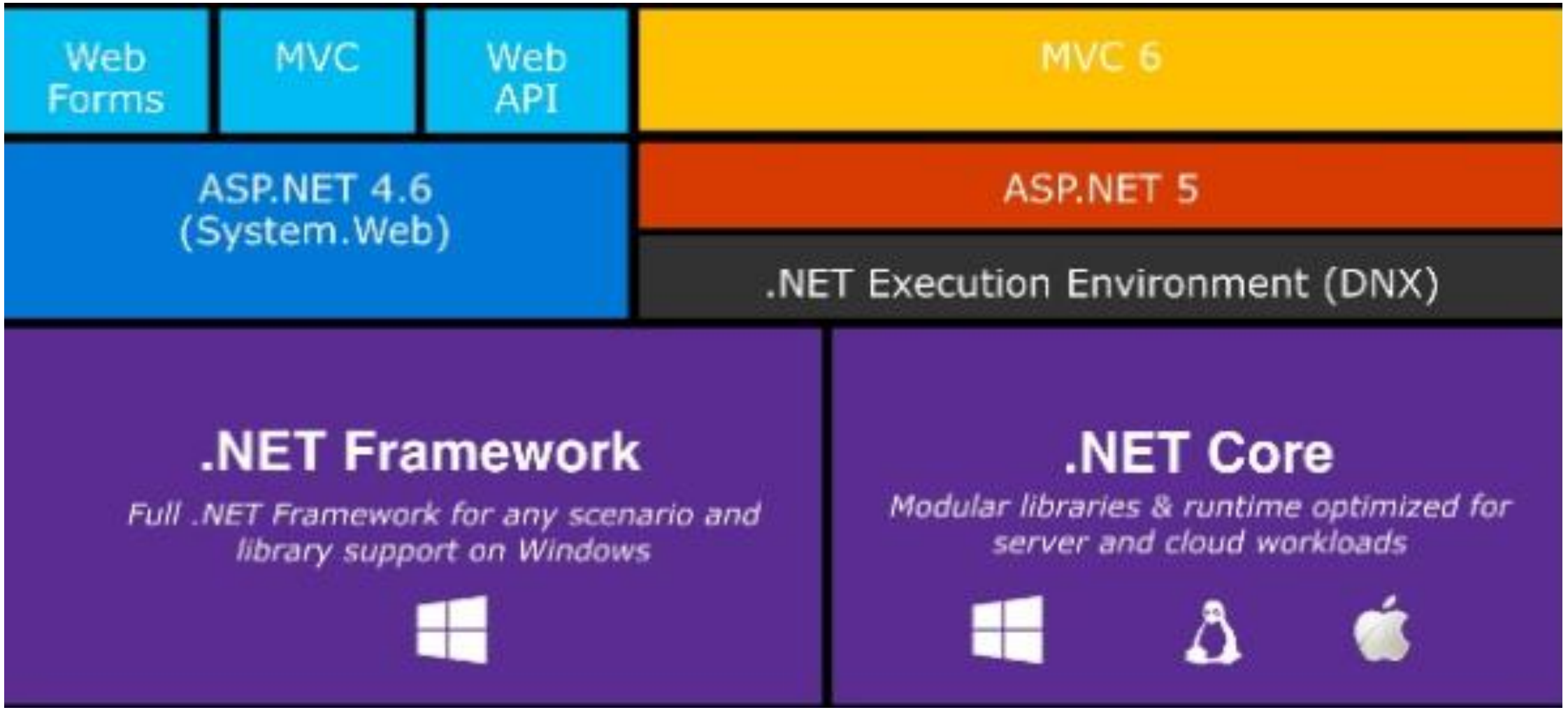
COMMON

RUNTIME
-NEXT GEN JIT

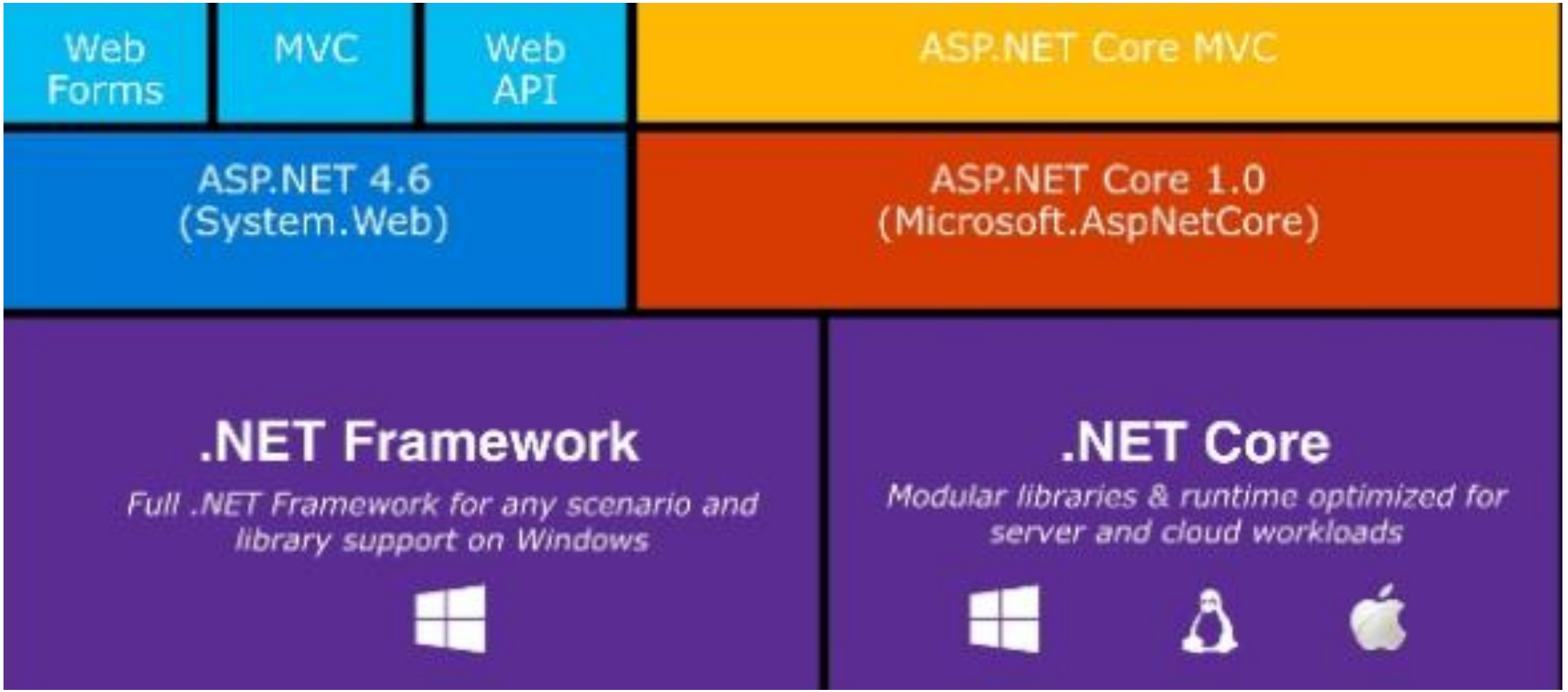
COMPILERS
- .NET COMPILER PLATFORM
- .LANGUAGE

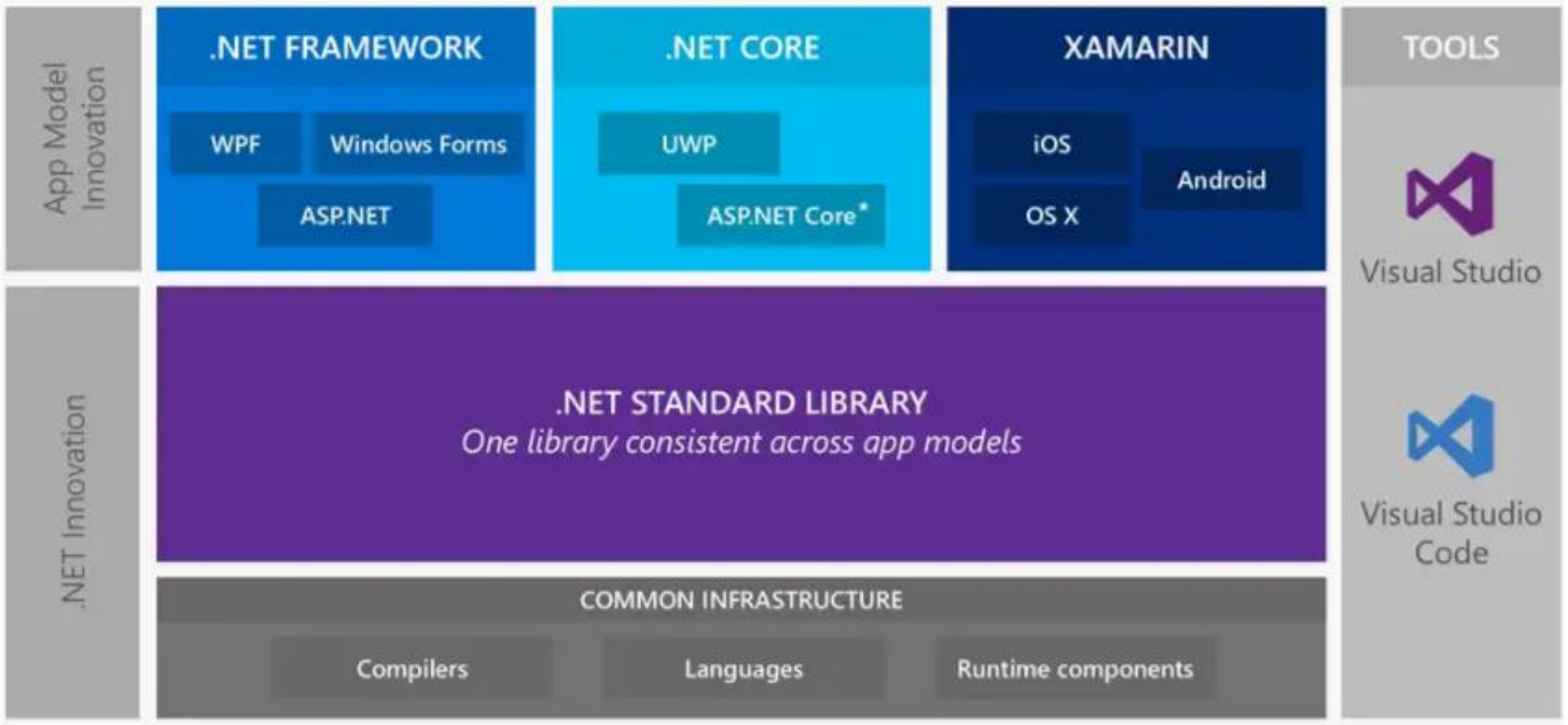
NUGET PACKAGES
-.NET CORE 5 LIBRARIES
.NET FRAMEWORK 4.6 LIB

ASP .NET 5



ASP.NET CORE

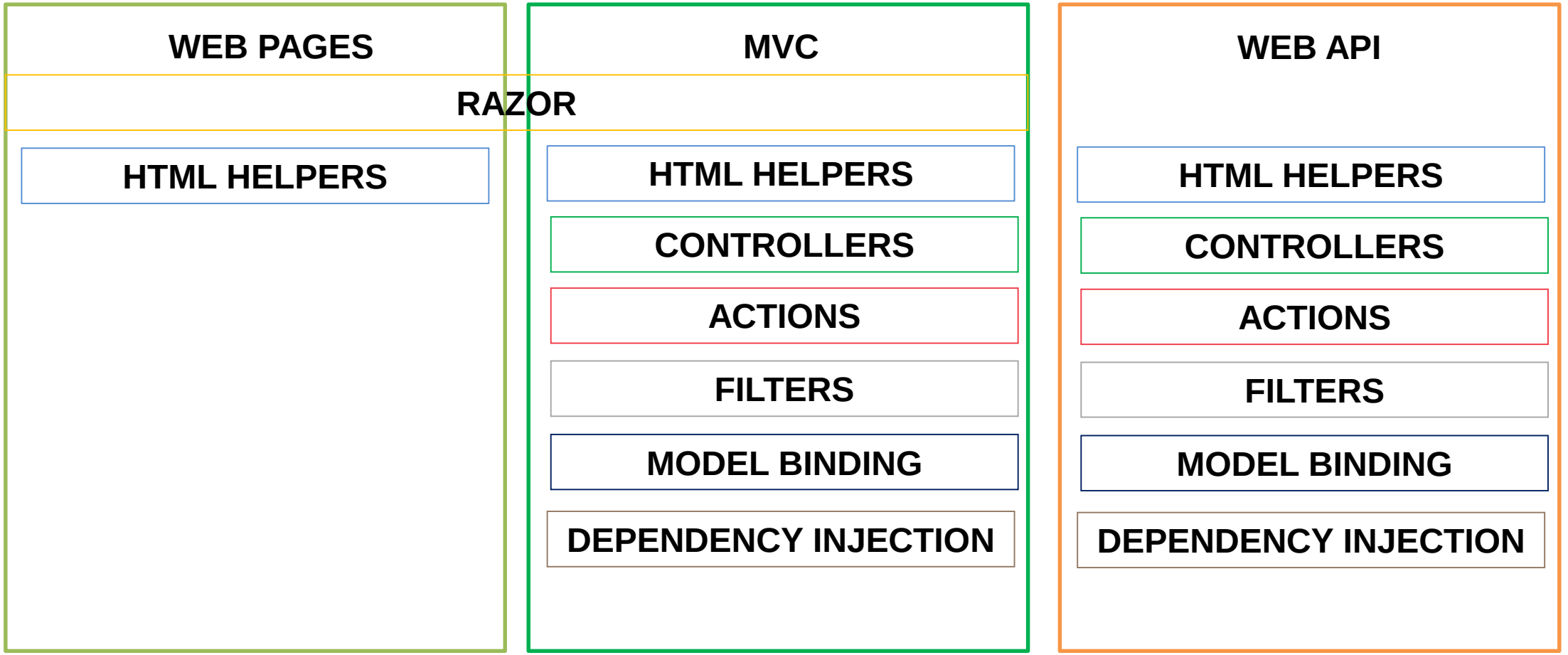




ASP.NET CORE FEATURE

- HOSTING
  - KESTREL,STARTUP
- MIDDLEWARE
  - ROUTING
  - AUTHENTICATION
  - STATIC FILE RENDERING
  - DIAGNOSTICS
  - ERROR HANDLING
  - SESSION MANAGEMENT
  - CORS
  - LOCALIZATION
  - CUSTOM
- DEPENDENCY INJECTION
- CONFIGURATION MANAGEMENT
- LOGGIN
- APPLICATION FRAMEWORKS
  - MVC
  - IDENTITY
  - SIGNALR

ASP.NET FRAMEWORKS



Motivation for Change in .NET

- Shift in the EcoSystem from Windows to Azure Cloud
- IIS is tightly coupled with
  - System.Web
  - System.Net
- Legacy Baggage
  - XML
  - Remoting
  - Enterprise Services

- Lightweight, stripped down version of application that can run from desktop to IOT platforms.

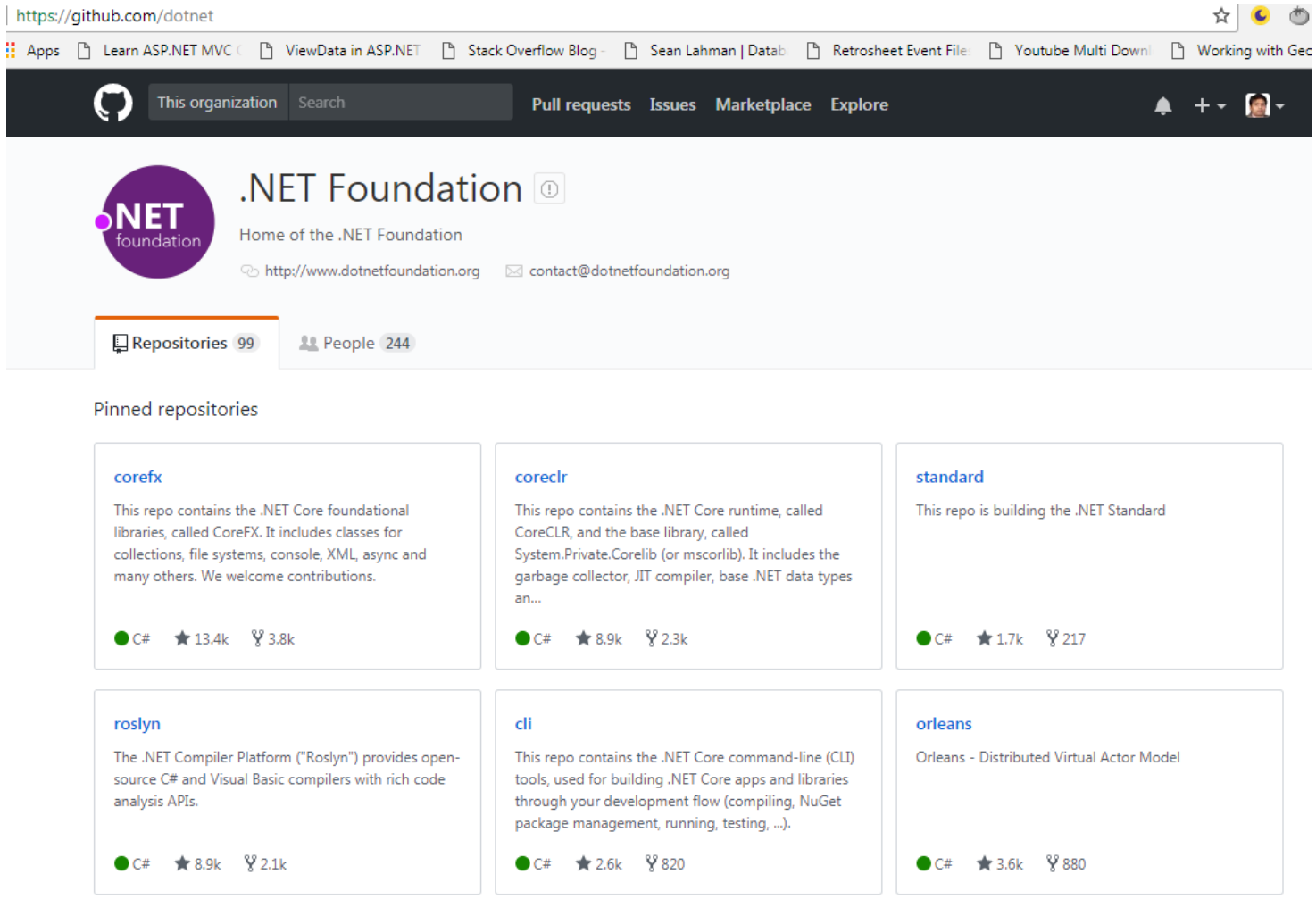
.NET CORE 1.0

- CROSS PLATFORM
- LIGHT-WEIGHT
- NEW CLI TOOLING
- UNIT TESTING SUPPORT WITH XUNIT, NUNIT
- DOCKER DEPLOYMENT POSSIBLE.

- 2016
  - Container Ecosystems
    - Docker adoption in .NET Development Environments
    - Azure cloud support for containers with different services
    - Windows Server 2016.
      - Support for Native Windows Containers and Hyper V Containers
      - Created Separate Version for Containers – server core and nano server
  - Microservices
    - Application architecture tuned for smaller services.

.NET Foundation

https://github.com/dotnet



.NET STANDARD VERSIONS

http://immo.landwerth.net/netstandard-versions/#

.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0
.NET Framework	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1 ~~4.6.2~~	4.6.1 ~~vNext~~	4.6.1
Mono	4.6	4.6	4.6	4.6	4.6	4.6	4.6	5.4
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.14
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.8
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0	7.0	8.0
Universal Windows Platform	10.0	10.0	10.0	10.0	10.0	10.0.16299	10.0.16299	10.0.16299
Windows	8.0	8.0	8.1					
Windows Phone	8.1	8.1	8.1					
Windows Phone Silverlight	8.0							

.NET STANDARD VERSIONS

- 1.4
  - DEFAULT CHOSEN BY 1.X COMMAND LINE TOOLS AND VISUAL STUDIO
  - LOWEST COMPATIBLE VERSION WITH .NET 4.6.1
- 1.6
  - SHIPPED WITH .NET CORE 1.0
  - HIGHEST VERSION SUPPORTED BY .NET CORE 1.0
  - .NET CORE 1.X SDK ORIGINALLY INCOMPATIBLE WITH .NET 4.6.1
- 2.0
  - RELEASED ON AUG 9,2017
  - COMMAND LINE TOOLS
    - TARGET 2.0 BY DEFAULT AFTER .NET CORE 2.0 SDK INSTALLATION
  - VISUAL STUDIO
    - SET TARGET TO 2.0 AFTER .NET CORE 2.0 SDK AND VS TEMPLATE INSTALLATION.
  - SUPPORTED BY .NET CORE 2.0
  - **.NET FRAMEWORK 4.6.1 IS NOW COMPATIBLE WITH .NET STANDARD 2.0**

.NET CORE

- MICROSOFT'S CROSS PLATFORM IMPLEMENTATION OF .NET
- Supported multiple operating systems
  - Windows
  - Linux
  - macOS
- Supported on multiple processors
  - X64, X86 on windows, ARM64, ARM32
- Open Source contribution by Microsoft through .NET Development Foundation.
- Components
  - Uses the CoreCLR runtime
  - Implements the .NET standard library
- Languages C#,F#,Visual Basic

.NET CORE PROJECTS

https://blog.rendle.io/a-guide-to-the-net-projects-on-github/

CORE FX

- FOUNDATION LIBRARIES FOR .NET CORE
- PLATFORM NEUTRAL

CORE CLR

- .NET CORE RUNTIME IMPLEMENTATION
- WRITTERN IN C/C++
- PLATFORM SPECIFIC

CORE RT

- RUNTIME OPTIMIZED FOR AHEAD OF TIME (AOT) COMPILATION
- TRANSPILES .NET CODE INTO NATIVE C++
- ALLOWS .NET CODE TO RUN WITHOUT THE .NET FRAMEWORK

.NET CORE PROJECTS

https://blog.rendle.io/a-guide-to-the-net-projects-on-github/

CLI

- COMMAND LINE INTERFACE
- PACKAGES TO CREATE PROJECTS, INSTALL PACKAGES, BUILD AND RUN APPLICATIONS.

LLILC

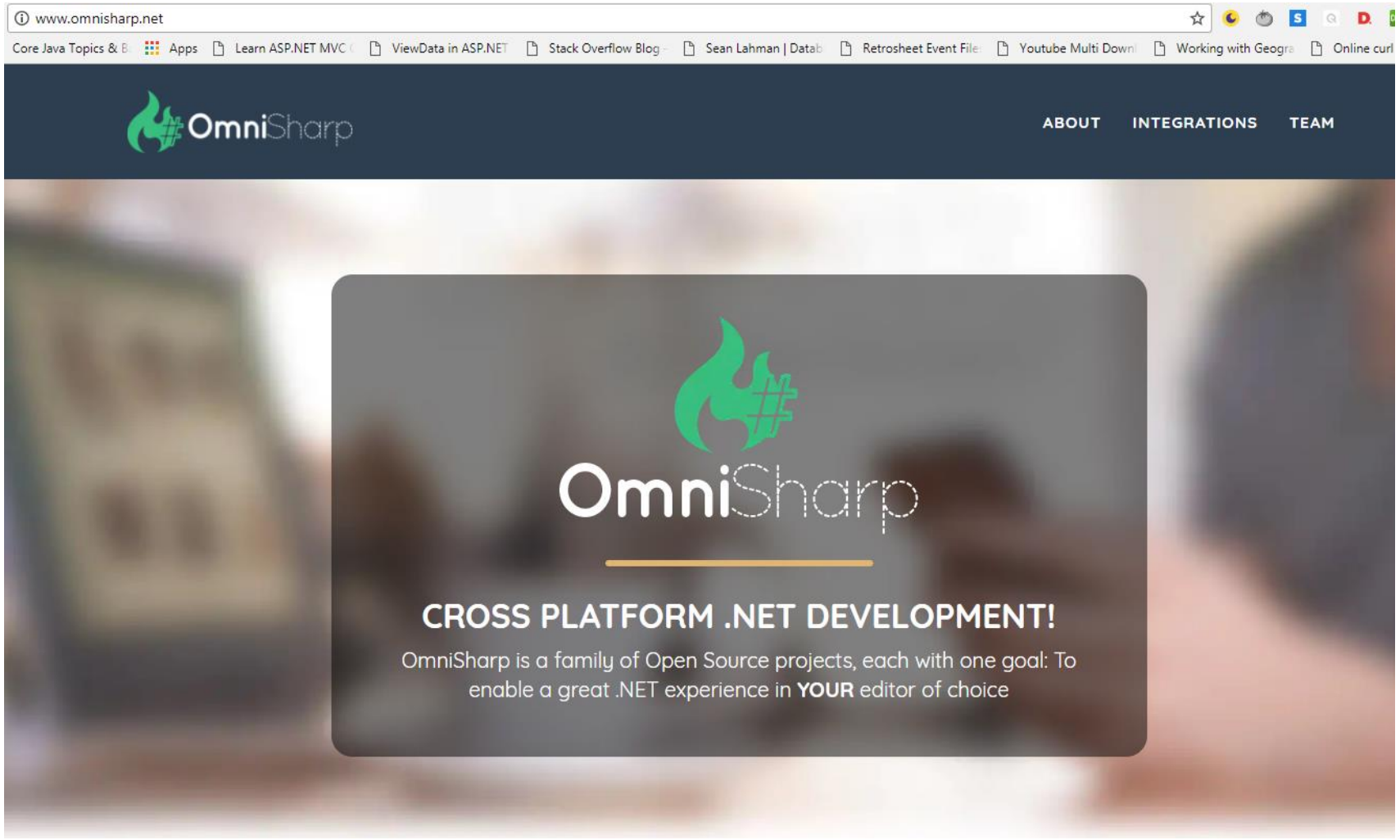
- LLVM MSIL COMPILER IS AN ALTERNATIVE COMPILER FOR TURNING .NET BYTECODE INTO MACHINE CODE USING THE LLVM COMPILER INFRASTRUCTURE.

ROSLYN

- THE C# AND VB COMPILERS AND TOOLS
- APACHE 2.0 LICENSE.
- AN OPEN ,CROSS-PLATFORM COMPILER FRAMEWORK IS FUNDAMENTAL TO ALL THE OTHER .NET CORE PROJECTS.

OMNISHARP

http://www.omnisharp.net/



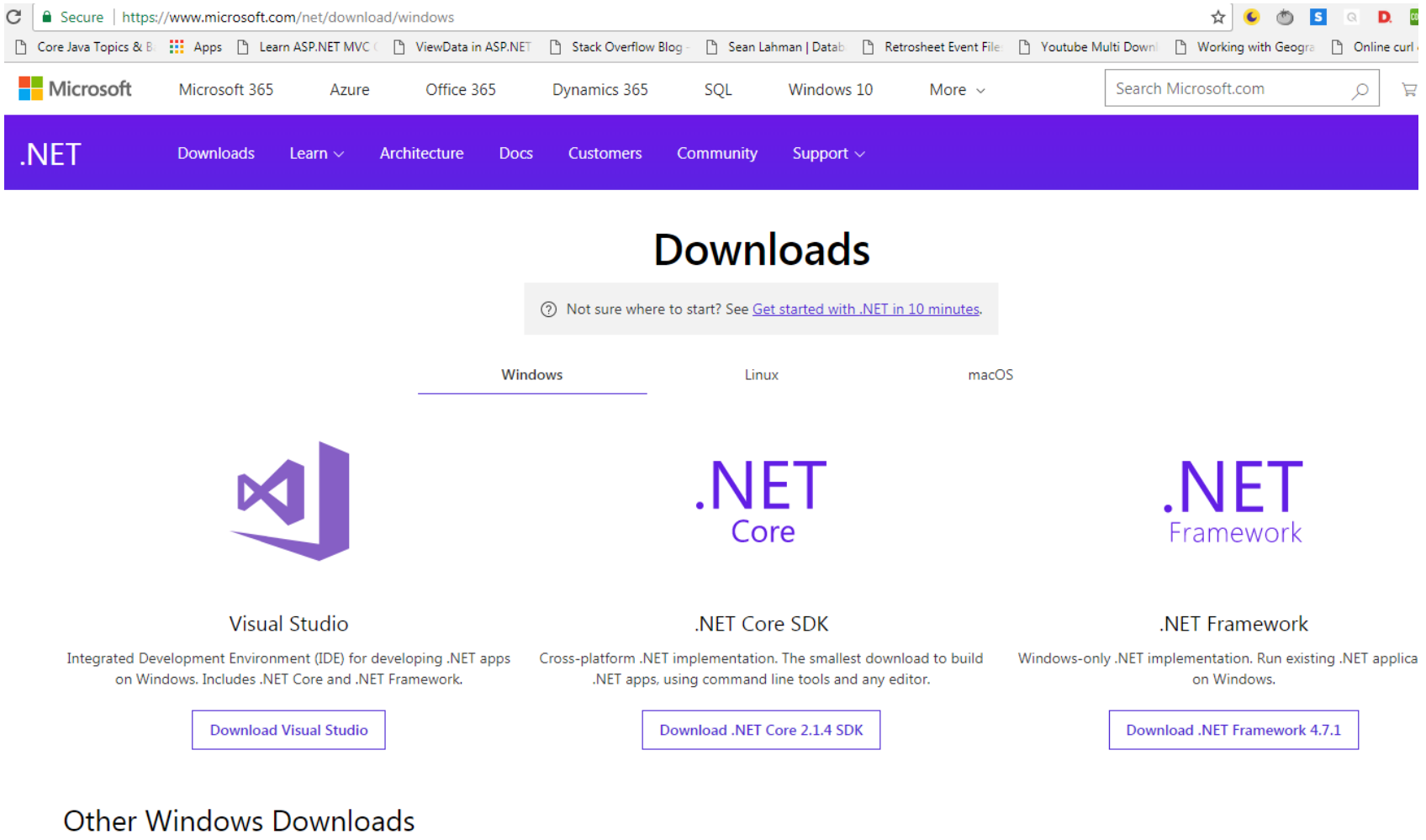
DOTNET.EXE

- Driver for executing .NET Core commands
- Host for .NET core applications
  - Platform-specific exe for executing .NET core programs.

- MSBUILD is now the foundation
  - Visual Studio 2017 uses *csproj*
  - *Dotnet.exe commands often forward to MSBUILD equivalent.*

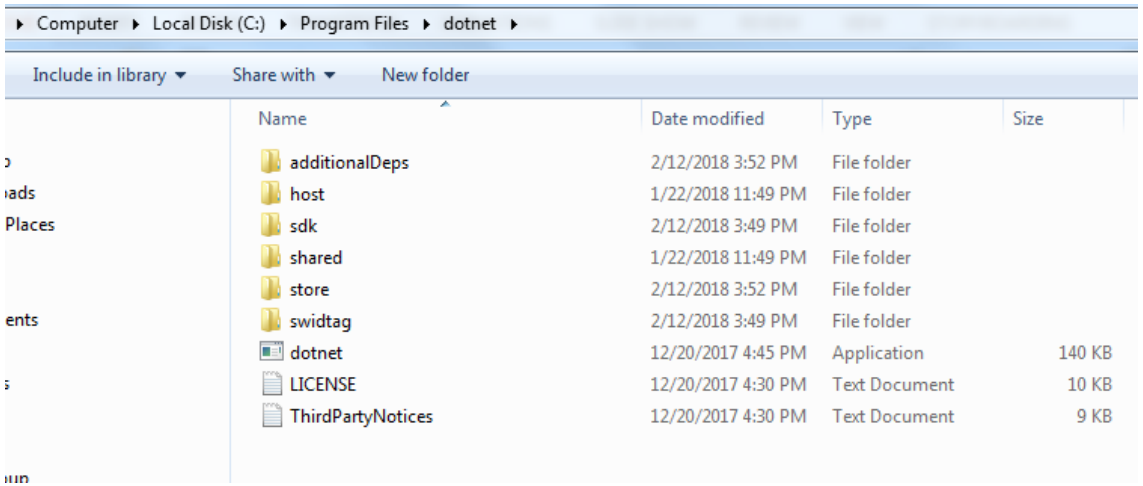
.NET CORE 2.0 SDK

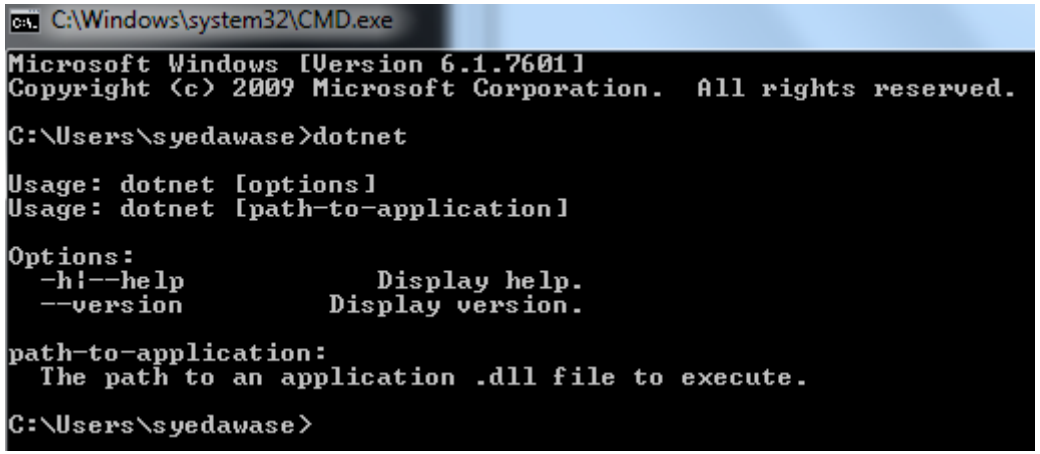
https://www.microsoft.com/net/download/windows



.NET CORE SDK INSTALLATION

C:\Program Files\dotnet





- Dotnet /?
- Dotnet –version
- Dotnet –info
- Donet --diagnostics

Dotnet sdk commands

Commands	description
New	Initialize .NET projects
Restore	Restore dependencies specified in the .NET project
Run	Compiles and immediately executes a .NET Project
Build	Builds a .NET Project
Publish	Publishes a .NET project for deployment including the runtime.
Test	Runs unit tests using the test runner specified in the project.
Pack	Creates a NuGet package
Migrate	Migrates a project.json based project to a msbuild based project.

Dotnet sdk commands

Commands	description
Clean	Clean build output
Sln	Modify solution files
Add	Add reference to the project
Remove	Remove reference from the project
List	List reference in the project
Nuget	Provides additional NuGet Commands
Msbuild	Runs Microsoft build engine <MSBUILD>
Vstest	Runs Microsoft test execution command line tool
-v | -verbosity	Set the verbosity level of the command, normal/diagnostic

Dotnet new


```
PS C:\Users\syedawase> dotnet new
Usage: new [options]

Options:
  -h, --help          Displays help for this command.
  -l, --list          Lists templates containing the specified name. If no name is specified, lists all t
  -n, --name          The name for the output being created. If no name is specified, the name of the cur
is used.
  -o, --output        Location to place the generated output.
  -i, --install       Installs a source or a template pack.
  -u, --uninstall     Uninstalls a source or a template pack.
  --type              Filters templates based on available types. Predefined values are "project", "item"
  --force             Forces content to be generated even if it would change existing files.
  -lang, --language   Specifies the language of the template to create.


Templates                                           Short Name       Language          Tags
----------------------------------------------------------------------------------------------------------
Console Application                                 console          [C#], F#, VB      Common/Console
Class library                                       classlib         [C#], F#, VB      Common/Library
Unit Test Project                                   mstest           [C#], F#, VB      Test/MSTest
xUnit Test Project                                  xunit            [C#], F#, VB      Test/xUnit
ASP.NET Core Empty                                  web              [C#], F#          Web/Empty
ASP.NET Core Web App (Model-View-Controller)        mvc              [C#], F#          Web/MVC
ASP.NET Core Web App                                razor            [C#]              Web/MVC/Razor Pages
ASP.NET Core with Angular                           angular          [C#]              Web/MVC/SPA
ASP.NET Core with React.js                          react            [C#]              Web/MVC/SPA
ASP.NET Core with React.js and Redux                reactredux       [C#]              Web/MVC/SPA
ASP.NET Core Web API                                webapi           [C#], F#          Web/WebAPI
global.json file                                    globaljson                         Config
NuGet Config                                        nugetconfig                        Config
Web Config                                          webconfig                          Config
Solution File                                       sln                                Solution
Razor Page                                          page                               Web/ASP.NET
MVC ViewImports                                     viewimports                        Web/ASP.NET
MVC ViewStart                                       viewstart                          Web/ASP.NET

Examples:
    dotnet new mvc --auth Individual
    dotnet new razor --auth Individual
    dotnet new --help
PS C:\Users\syedawase>
```

EXERCISE DEMO: 1.1

LEARNING OUTCOMES

- Creating a basic dotnet core 2 **console application**
- Understanding the folder structure
- Building dotnet and deploying a dotnet core 2 console application.

USE CASES

- Use case 1
- Use case 2
- Use case 3
- Use case 4

.NET CORE CONSOLE APP:CREATE/RESTORE

1

- Create a new console application using dotnet core 2.0
- **Dotnet new console**

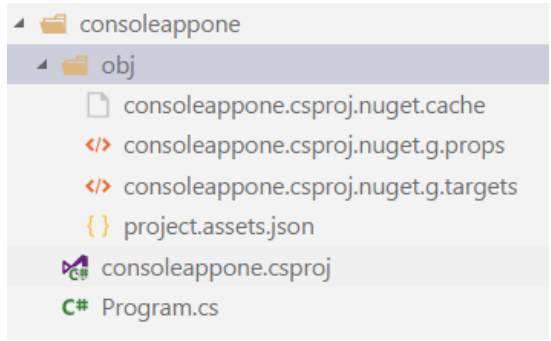
```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\consoleappone> dotnet new console
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\consoleappone\cc
..
  Restoring packages for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\consoleappone\consc
  Generating MSBuild file H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\consoleappone\obj\
j.nuget.g.props.
  Generating MSBuild file H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\consoleappone\obj\
j.nuget.g.targets.
  Restore completed in 721.33 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\console
e.csproj.

Restore succeeded.
```

2

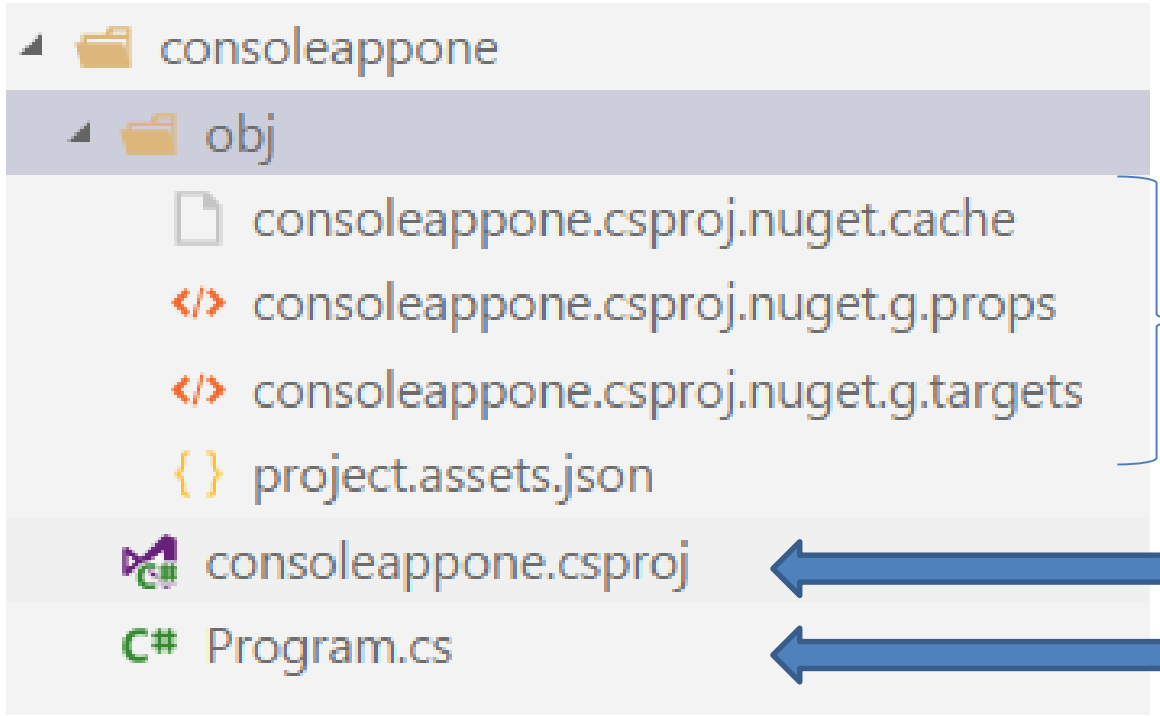
- Runs post-creation actions
- Dotnet restore (in dotnet 1.x we had to explicitly call this)

3

- Creates the following folder structure
  - Obj
  - projectName.csproj
  - Program.cs



.NET CORE 2.0 CONSOLE APP



- Project manifest files
- Nuget project properties

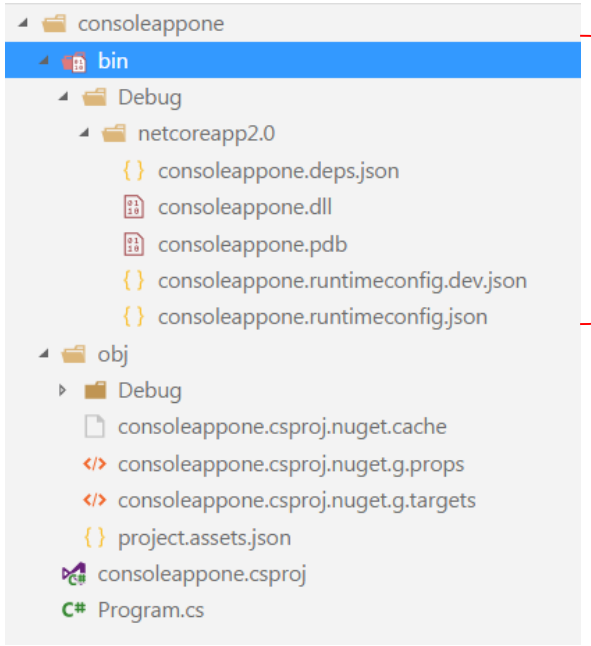
- Indicates output type
- Target framework information

```
using System;

namespace consoleappone
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

.NET CORE 2.0 CONSOLE APP:BUILD

- Building your dotnet core application
  - dotnet build

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\consoleappone> dotnet build
Microsoft (R) Build Engine version 15.5.180.51428 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 42.23 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscena
ppone.csproj.
  consoleappone -> H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\consoleappon
onsoleappone.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:12.55
```



Created post build operation

- Performs a debug build of the code.
- Use “-c” to build a “release build” of the application.
- .NET core application is a dll

.NET CORE 2.0 CONSOLE APP:RUN

- Running your dotnet console app using
  - dotnet ./bin/Debug/netcoreapp2.0/app.dll

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\consoleappone> dotnet .\bin\Debug\netcoreapp2.0\
consoleappone.dll
Hello World!
```

- Alternatively navigating to the project folder and running it
  - dotnet run

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\consoleappone> dotnet run
Hello World!
```

EXERCISE DEMO: 1.2

LEARNING OUTCOMES

- Creating a dotnet core 2 **class library**
- Understand the basic structure of the dotnet core 2 class library application
- Building and executing your application

USE CASES

- Use case 1
- Use case 2
- Use case 3
- Use case 4

.NET CORE CLASS LIBRARY:CREATE

1

- Create a new class library using dotnet core 2.0
- **Dotnet new classlib**

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\classapptwo> dotnet new classlib
The template "Class library" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\classappt
wo\classapptwo.csproj...
  Restoring packages for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\classapptwo\
classapptwo.csproj...
  Generating MSBuild file H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\classapptwo
\obj\classapptwo.csproj.nuget.g.props.
  Generating MSBuild file H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\classapptwo
\obj\classapptwo.csproj.nuget.g.targets.
  Restore completed in 619.79 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\
classapptwo\classapptwo.csproj.

Restore succeeded.
```

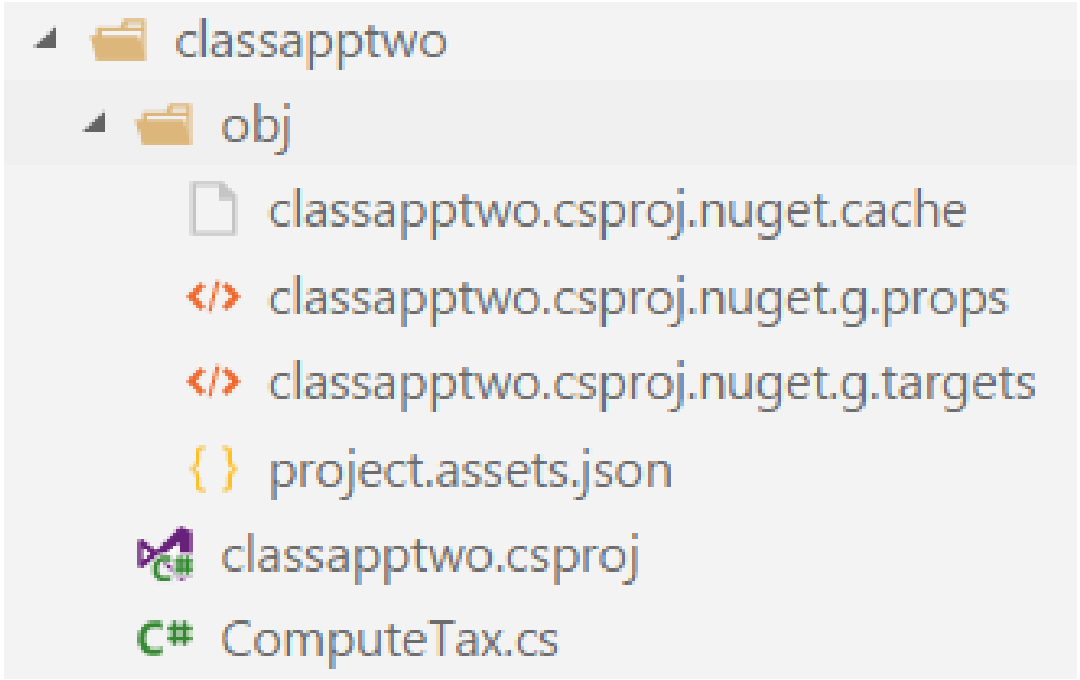
2

- Runs post-creation actions
- Dotnet restore (in dotnet 1.x we had to explicitly call this)

3

- Creates the following folder structure
  - Obj
  - projectName.csproj
  - Class1.cs

.NET Core Class: Compute Tax

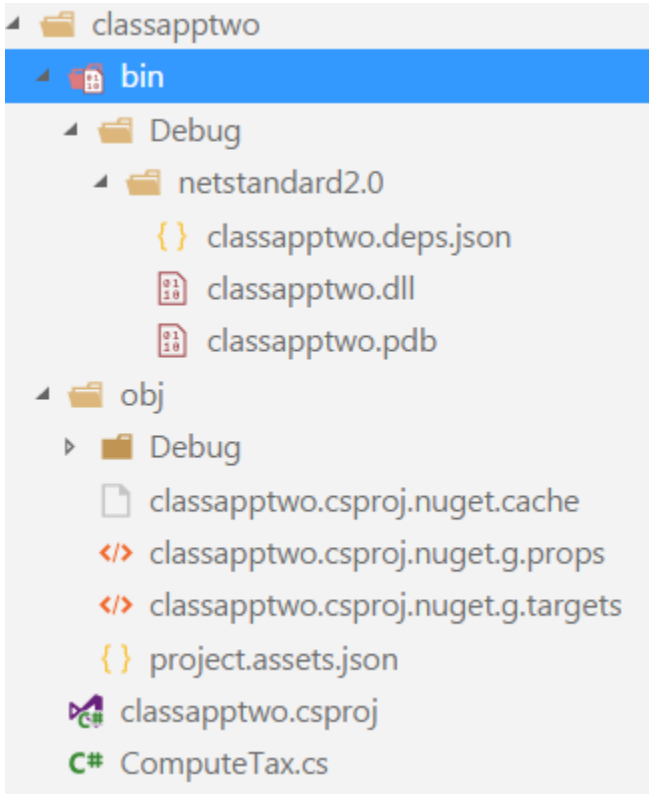



```
using System;

namespace classapptwo
{
    public class ComputeTax
    {
        public int StateTax(int amount, int sgstrate){
            return amount*sgstrate/100;
        }

        public int CentralTax(int amount; int cgstrate){
            return amount*cgstrate/100;
        }
    }
}
```

- Build using => dotnet build

Building your class library

- Building the class library
  - dotnet build




```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\classapptwo> dotnet build
Microsoft (R) Build Engine version 15.5.180.51428 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 33.49 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appsce
roj.
  classapptwo -> H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\classapptwo\
wo.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:04.52
```

EXERCISE DEMO: 1.3

LEARNING OUTCOMES

- Creating a dotnet core 2 **solution to consume the class library created.**
- Understand the process and structure of the solution
- Build and execute the solution.

USE CASES

- Use case 1
- Use case 2
- Use case 3
- Use case 4

.NET CORE SOLUTION:CREATE

1

- Create a new solution using dotnet core 2.0
- **Dotnet new sln**

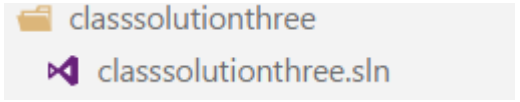
```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\classsolutionthree> dotnet new sln
The template "Solution File" was created successfully.
```

2

- Creates a solution file

3

- Just creates a solution file



Adding Projects to the Solution

- Adding existing projects (console application and class library) to the solution
  - dotnet sln add ../project/project.csproj

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\classsolut
ionthree> dotnet sln add ..\consoleappone\consoleappone.csproj
Project `..\consoleappone\consoleappone.csproj` added to the solution.
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\classsolut
ionthree> dotnet sln add ..\classapptwo\classapptwo.csproj
Project `..\classapptwo\classapptwo.csproj` added to the solution.
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\classsolut
ionthree>
```

Adding a reference to the library

- Adding reference to the solution.
  - dotnet add reference ./prj/prj.csproj

```
Usage: dotnet add <PROJECT> reference [options] <args>
```

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\classsolutionthree>  dotnet add ../consoleappone reference ..\classapptwo\cl
assapptwo.csproj
Reference `..\classapptwo\classapptwo.csproj` added to the project.
```

```
<Project Sdk="Microsoft.NET.Sdk">

  <ItemGroup>
    <ProjectReference Include="..\classapptwo\classapptwo.csproj" />
  </ItemGroup>

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp2.0</TargetFramework>
  </PropertyGroup>

</Project>
```

Consoleapp.csproj

Consuming the class library


```
using System;
using classapptwo;

namespace consoleappone
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
            ComputeTax ct=new ComputeTax();
            var mytax =ct.StateTax(20000,18);
            Console.WriteLine(mytax);
        }
    }
}
```

- Building the solution
  - dotnet build
- Executing the project
  - Cd consoleappone
  - Dotnet run

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios
\consoleappone> dotnet run
Hello World!
3600
```

Adding package to existing project

```
dotnet add package Microsoft.AspNetCore
```

- Used to add additional functionality to the existing project to render web application functionality.

EXERCISE DEMO: 1.4

LEARNING OUTCOMES

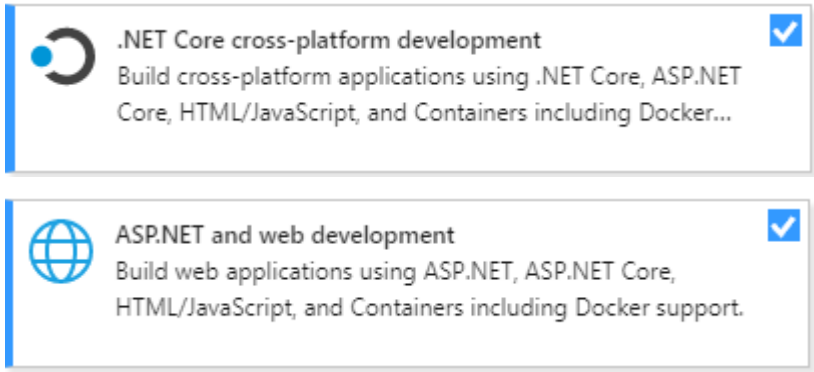
- Creating .NET core apps with Visual Studio 2017.
- Install .NET core 2.0 SDK
  - http://www.Microsoft.com/net/download/core
- Install project templates through Visual Studio Installer
  - .NET Core Cross-platform development
  - ASP.NET and Web Development

USE CASES

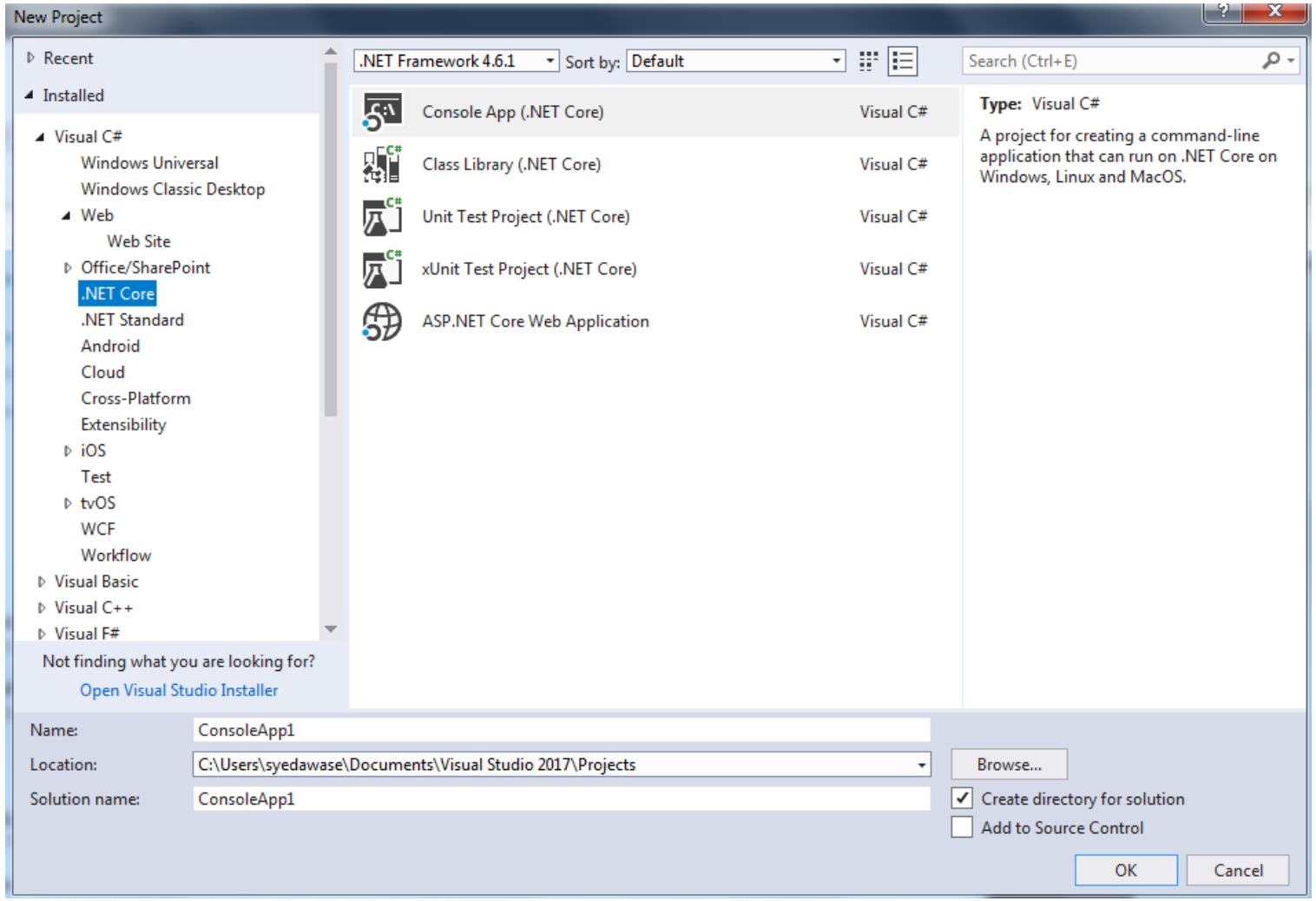
- Use case 1
- Use case 2
- Use case 3
- Use case 4

Installing .NET CORE for Visual Studio 2017

1. Open Visual Studio Installer
2. Modify Installation
3. Select
    1. .NET CORE CROSS-PLATFORM DEVELOPMENT
    2. ASP.NET and Web Development



Creating a new .NET Core Application in VS2017



NuGet Packages

- Fine-grained packages
  - Represent one assembly whose name is the same as package.
  - Mostly independent of other packages.
  - Support for different platforms
- Packages support multiple frameworks.

- NuGet Metapackages
  - Represents a set of related packages.
- Each framework implicitly references a metapackage
- .NETStandard.Library
  - Implicitly referenced by .NET standard framework
  - Ability to run on multiple runtimes.
- Microsoft.NETCore.App
  - Implicitly referenced by .NET CORE
  - Dependent on NETStandard.Library

Packaging your library as NuGet Package.

https://docs.microsoft.com/en-us/nuget/quickstart/create-and-publish-a-package-using-visual-studio

- Create a NuGet package
  - A standard approach for distribution of libraries in .NET Core
  - **dotnet pack**
  - Creates a zip file with ***.nupkg*** extension
- Publishing the NuGet package
  - Register/Sign in to www.nuget.org
  - Upload the nugget package file
- Configure the owners of the package.

EXERCISE DEMO: 1.5

LEARNING OUTCOMES

- Deployment in .Net Core application

USE CASES

- Use case 1
- Use case 2
- Use case 3
- Use case 4

Two types of Deployment in .NET CORE

Framework-dependent Deployment FDD

- **Assumes** that .NET Core installed on the target machine.
- Contains only the application and third party library dependency
- dotnet publish –c Release
  - Creates a publish folder in bin/Release
  - Contains .dlls for app and dependent assemblies.
  - Contains pdbs for app and dependent assemblies.

Self-contained Deployment (SCD)

- Contains .NET Core libraries and runtime.
- Contains the application and third party library dependencies.
- Developers need to specify the target platform where the app will run
  - Add RuntimeIdentifier element
    - Add to the PropertyGroup section of csproj file
  - Indicate platform using Runtime Identifier(RID)
  - [os].[version]-[architecture]

Self Contained Deployment

- Execute the publish command for each RID
  - Dotnet publish –c Release –r [RID]
- Creates a folder named publish in bin/release/netcoreapp2.0/RID with
  - Renamed version of dotnet.exe
    - Which is used to launch the apps
    - Named the same as the application name on Windows.
  - .NET Core binaries and runtime.
  - Dlls for app and dependent assemblies
  - Pdbs for app and dependent assemblies.

FDD vs SCD

Framework-dependent Deployment FDD

- Advantages
  - Smaller deployment package
  - Single .NET Core installation shared by multiple apps on target machine
  - Target platform does not need to be specified in advance.
- Disadvantages
  - Correct version of .NET Core must be present on the target machine.
  - Application behavior is dependent on the version.

Self-contained Deployment (SCD)

- Advantages
  - Not dependent on what is installed on the target machine
  - Guaranteed that application will run as intended.
- Disadvantages
  - Large deployment package
  - No sharing of .NET Core installations by multiple apps on target machine.
  - Target platform must be specified in advance.

SECTION -XVIII

ASP.NET CORE 2.0

ASP.NET CORE 2 Features

- Cross-platform support
- Micro service architecture
- Working with docker and containers
- Performance and scalability
- Side-by-side deployments
- Technology restrictions.
- ASP.NET MVC and Web API have been combined into a single framework.
- Runtime Store : it contains compiled packages, which were compiled using the native machine language and it is key for improved performance.
- Fully supports Razor engine.
- Seamless integration with client-side frameworks including Angular, Knockout and Bootstrap.

ASP.NET CORE Features

- Envrionment based configuration system ready for cloud hosting
- Built-in dependency injection functionalities.
- Light-weight and modular HTTP request pipeline.
- Host the same application in IIS, self-host, Docker, Cloud
- Side-by-side deployments with older versions.

Not found currently in ASP.NET core

- ASP.NET Web Forms applications.
- ASP.NET Web Pages applications.
- WCF Services , it only has support for WCF Client
- WorkFlow services: Windows Workflow foundation, workflow Services and WCF data services are not supported.
- WPF and Windows forms applications.

When to choose ASP.NET CORE 2.0

- Migrating your existing application to ASP.NET CORE 2.0 straight away is not easy. As there is not much support.
- But for new application development, ASP.NET Core 2.0 is ideal choice.
- IOT and Cross Platform Application development ASP.NET CORE would be an ideal choice.
- Specifically targeted microservices.
- Docker containers.
- High performance and highly scalable applications.
- Multiple applications with different .NET versions side by side.

ASP.NET Core 2.0 Empty Web App

EXERCISE DEMO: 2.1

LEARNING OUTCOMES

- Creating a ASP.NET Core Empty web application using CLI

ASP.NET Empty Web Application:Create

1

- Create a new solution using dotnet core 2.0
- **Dotnet new web**

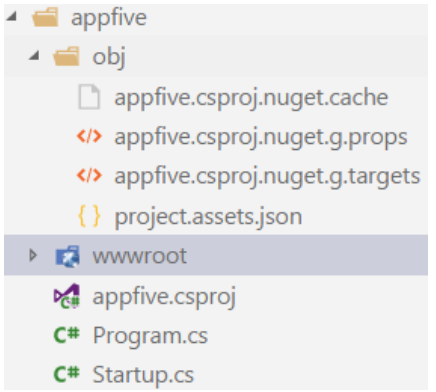
```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetco
re-appscenarios\appfive> dotnet new web
The template "ASP.NET Core Empty" was created succ
essfully.
This template contains technologies from parties o
ther than Microsoft, see https://aka.ms/template-3
pn for details.

Processing post-creation actions...
Running 'dotnet restore' on H:\awase-harddisk\CT\C
#2017\DOTNETCORE\dotnetcore-appscenarios\appfive\a
ppfive.csproj...
  Restoring packages for H:\awase-harddisk\CT\C#2017\DOTN
  Generating MSBuild file H:\awase-harddisk\CT\C#2017\DOT
  Generating MSBuild file H:\awase-harddisk\CT\C#2017\DOT
  Restore completed in 29.38 sec for H:\awase-harddisk\CT

Restore succeeded.
```

2

- Runs post-creation actions
- Dotnet restore (in dotnet 1.x we had to explicitly call this)

3

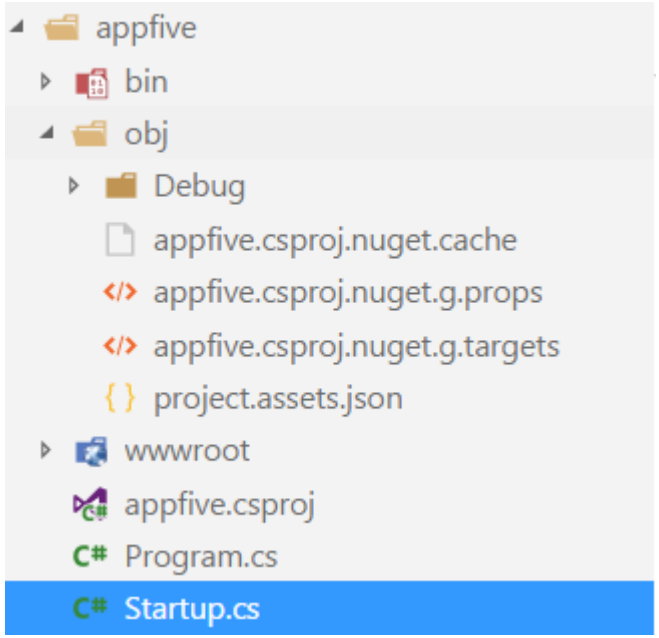
- Creates an application structure with nuget packages
  - Wwwroot
  - Project.csproj
  - Program.cs
  - Startup.cs(Global.asax)



Building your ASP.NET Core: build

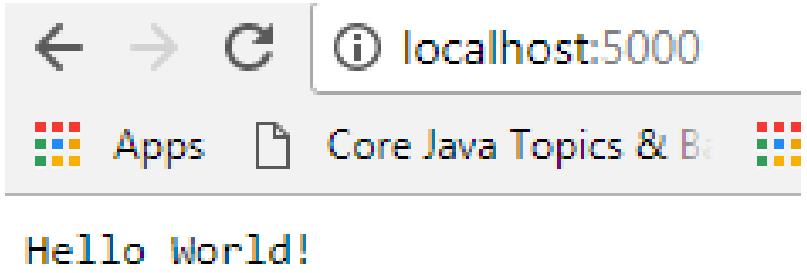
- Building your ASP.NET Core application
  - dotnet build

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appfive> dotnet build
Microsoft (R) Build Engine version 15.5.180.51428 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 259.12 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-app
  appfive -> H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appfive\bin\De

Build succeeded.
    0 Warning(s)
    0 Error(s)
```



Running your ASP.NET Core: run

- Running your ASP.NET Core
  - Dotnet run




```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appfive> dotnet run
Hosting environment: Production
Content root path: H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appfive
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
info: Microsoft.AspNetCore.Hosting.Internal.WebHost[1]
      Request starting HTTP/1.1 GET http://localhost:5000/
info: Microsoft.AspNetCore.Hosting.Internal.WebHost[2]
      Request finished in 144.5773ms 200
info: Microsoft.AspNetCore.Hosting.Internal.WebHost[1]
```

ASP.NET MVC Core 2.0

EXERCISE DEMO: 2.2

LEARNING OUTCOMES

- Creating an ASP.NET Core MVC 2.0 Application
- Understanding the application structure
- Building and executing the ASP.NET Core MVC 2.0 Application

ASP.NET MVC 2.0 Application :Create

1

- Create a new solution using dotnet core ASP.NET MVC 2.0
- **Dotnet new mvc**

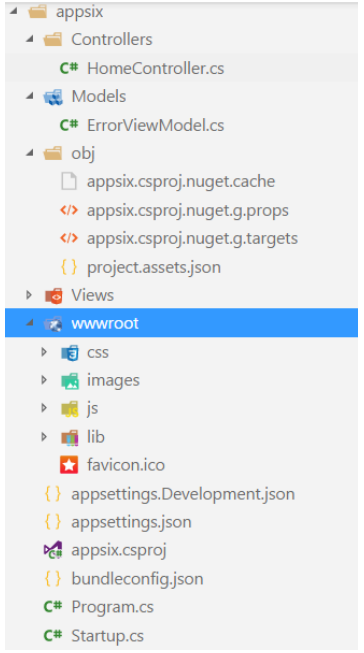
```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appsix> dotnet new mvc
The template "ASP.NET Core Web App (Model-View-Controller)" was created successfully.
This template contains technologies from parties other than Microsoft, see https://aka.ms/template-3pn for details.

Processing post-creation actions...
Running 'dotnet restore' on H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appsix\appsix.csproj...
  Restoring packages for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appsix\appsix.csproj...
  Restoring packages for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appsix\appsix.csproj...
  Generating MSBuild file H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appsix\obj\appsix.csproj.nuget.g.props
.
  Generating MSBuild file H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appsix\obj\appsix.csproj.nuget.g.targe
ts.
  Restore completed in 11.15 sec for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appsix\appsix.csproj.
  Restore completed in 11.42 sec for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appsix\appsix.csproj.

Restore succeeded.
```

2

- Runs post-creation actions
- Dotnet restore (in dotnet 1.x we had to explicitly call this)

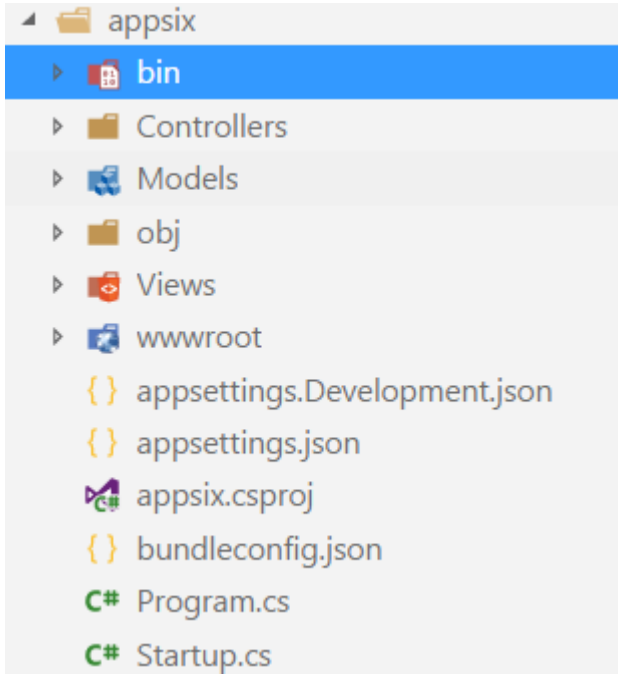
3

- Creates the following application structure



ASP.NET MVC Core 2.0:Build

- Building your ASP.NET MVC Core 2.0 application
  - Dotnet build

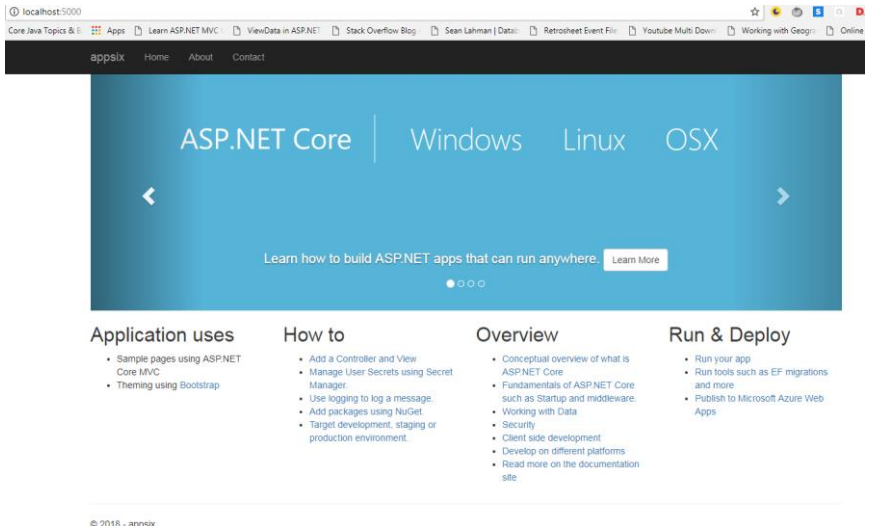



```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appsix> dotnet build
Microsoft (R) Build Engine version 15.5.180.51428 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 266.57 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore
.
  Restore completed in 189.68 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore
.
  appsix -> H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appsix\bin\

Build succeeded.
    0 Warning(s)
    0 Error(s)
```

ASP.NET Core MVC 2.0:Run

- Running your ASP.NET Core MVC 2.0 application
  - Dotnet run
  - http://localhost:5000/

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appsix> dotnet run
Hosting environment: Production
Content root path: H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appsix
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```



ASP.NET MVC Core 2.0 (Razor)

EXERCISE DEMO: 2.3

LEARNING OUTCOMES

- Creating an ASP.Net Core 2.0 with razor view engine
- Understanding the application structure
- Building and running your application.

ASP.NET MVC 2.0 Razor Application :Create

1

- Create a new solution using dotnet core ASP.NET MVC 2.0
- **Dotnet new razor**

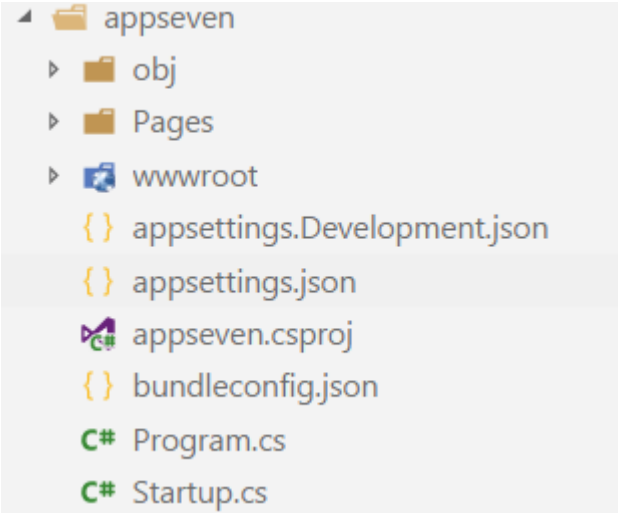
```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appseven> dotnet new razor
The template "ASP.NET Core Web App" was created successfully.
This template contains technologies from parties other than Microsoft, see https://aka.ms/templa

Processing post-creation actions...
Running 'dotnet restore' on H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appse
  Restoring packages for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appseven
  Restore completed in 172.03 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenar
proj.
  Generating MSBuild file H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appseve
get.g.props.
  Generating MSBuild file H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appseve
get.g.targets.
  Restore completed in 4.01 sec for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenari
roj.

Restore succeeded.
```

2

- Runs post-creation actions
- Dotnet restore (in dotnet 1.x we had to explicitly call this)

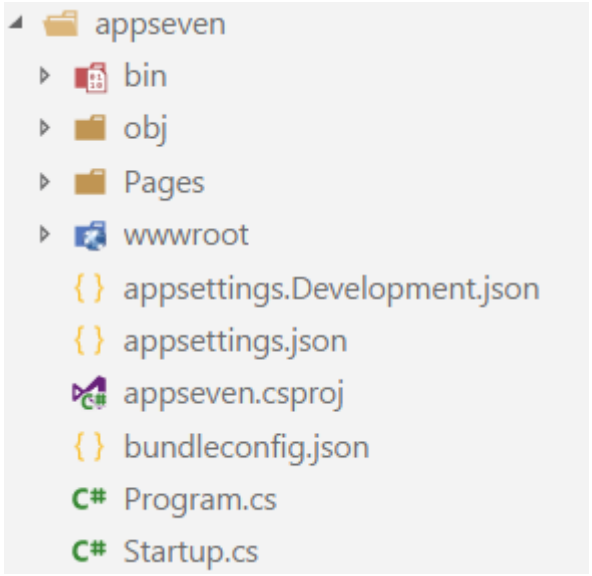
3

- Creates the following application structure



ASP.NET MVC 2.0 Razor Application :Build

- Building your ASP.NET MVC 2.0 Razor Application
  - Dotnet build

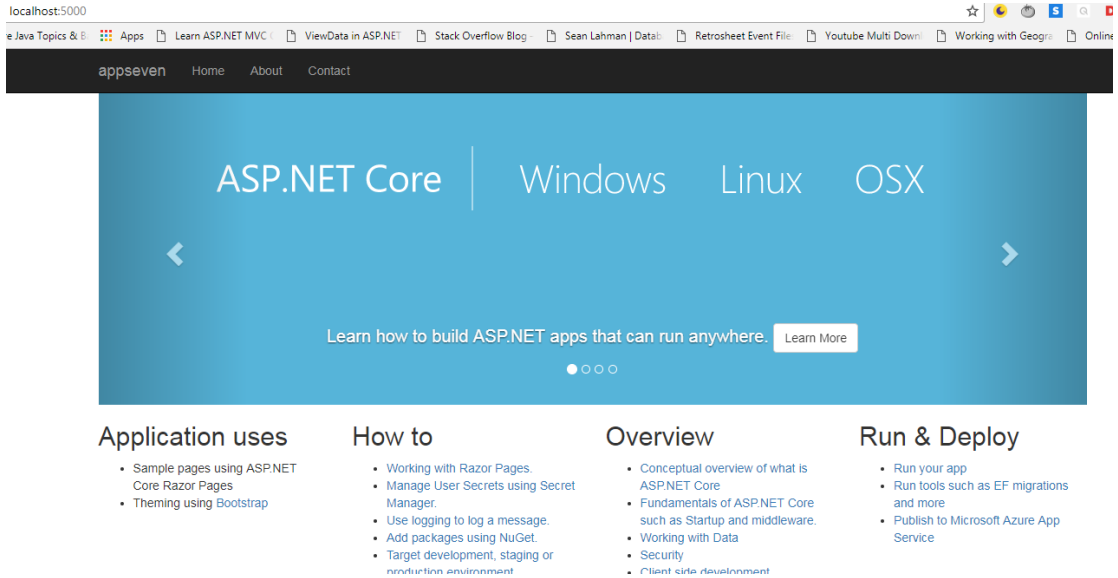



```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appseven> dotnet build
Microsoft (R) Build Engine version 15.5.180.51428 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 186.59 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-ap
proj.
  Restore completed in 150.59 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-ap
proj.
  appseven -> H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appseven\bin
ll

Build succeeded.
    0 Warning(s)
    0 Error(s)
```

ASP.NET MVC 2.0 Razor Application :run

- Running your ASP.NET MVC 2.0 razor application
  - Dotnet run

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appseven> dotnet run
Hosting environment: Production
Content root path: H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appseven
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```



ASP.NET MVC Core 2.0 (Angular)

EXERCISE DEMO: 2.4

LEARNING OUTCOMES

- Creating an ASP.NET Core 2.0 with Angular
- Understanding the application structure
- Building and running your application.

ASP.NET MVC 2.0 Angular Application :Create

1

- Create a new solution using dotnet core ASP.NET Core with Angular
- **Dotnet new angular**

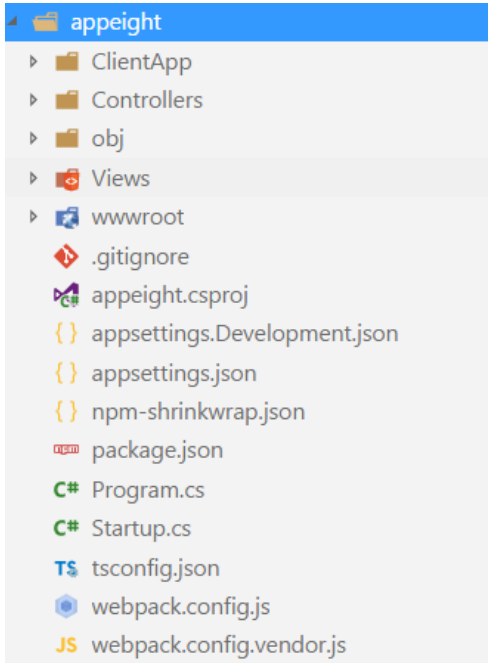
```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appeight> dotnet new angular
The template "ASP.NET Core with Angular" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appe
  Restoring packages for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appeigh
```

2

- Runs post-creation actions
- Dotnet restore (in dotnet 1.x we had to explicitly call this)

3

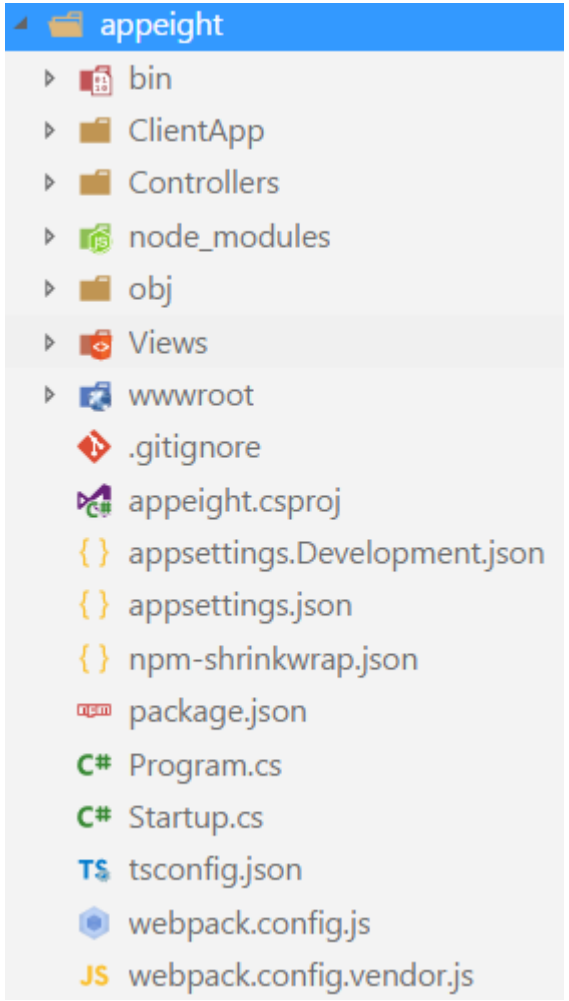
- Creates the following application structure
- Now run npm install –save



ASP.NET MVC 2.0 Angular Application :build

- Building your ASP.NET MVC 2.0 Angular Application
  - Dotnet build

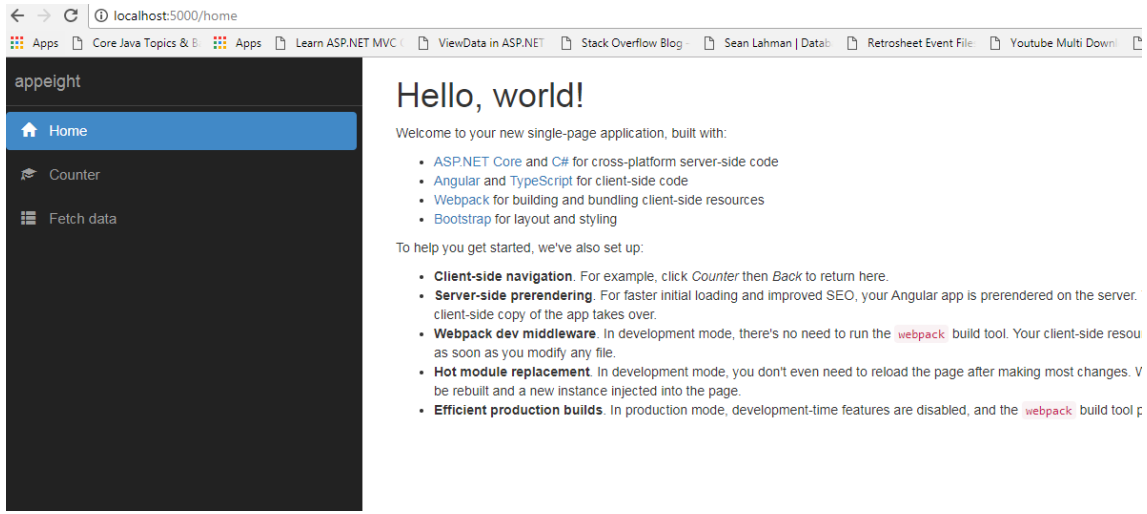
```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appeight> dotnet build
Microsoft (R) Build Engine version 15.5.180.51428 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 235.44 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-app
proj.
  Restore completed in 130.41 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-app
proj.
  appeight -> H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appeight\bin\
ll
  v8.1.3
  Performing first-run Webpack build...
  Hash: 4adf5b975b06d7f766a2316997213570377444fe
  Version: webpack 2.5.1
```



ASP.NET Core 2.0 Angular: run

- Executing ASP.NET Core 2.0 Angular
  - Dotnet run

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appeight> dotnet run
Hosting environment: Production
Content root path: H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appeight
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```



ASP.NET MVC Core 2.0 (React)

EXERCISE DEMO: 2.5

LEARNING OUTCOMES

- Creating your application with ASP.NET Core with React.js
- Understanding the application structure
- Building and executing your application.

USE CASES

- Use case 1
- Use case 2
- Use case 3
- Use case 4

ASP.NET MVC 2.0 React Application :Create

1

- Create a new solution using dotnet core ASP.NET Core with ReactJS
- **Dotnet new react**

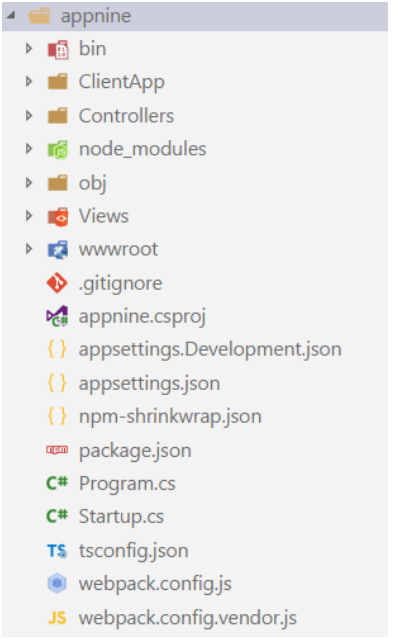
```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appnine> dotnet new react
The template "ASP.NET Core with React.js" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\
  Restoring packages for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appn
```

2

- Runs post-creation actions
- Dotnet restore (in dotnet 1.x we had to explicitly call this)

3

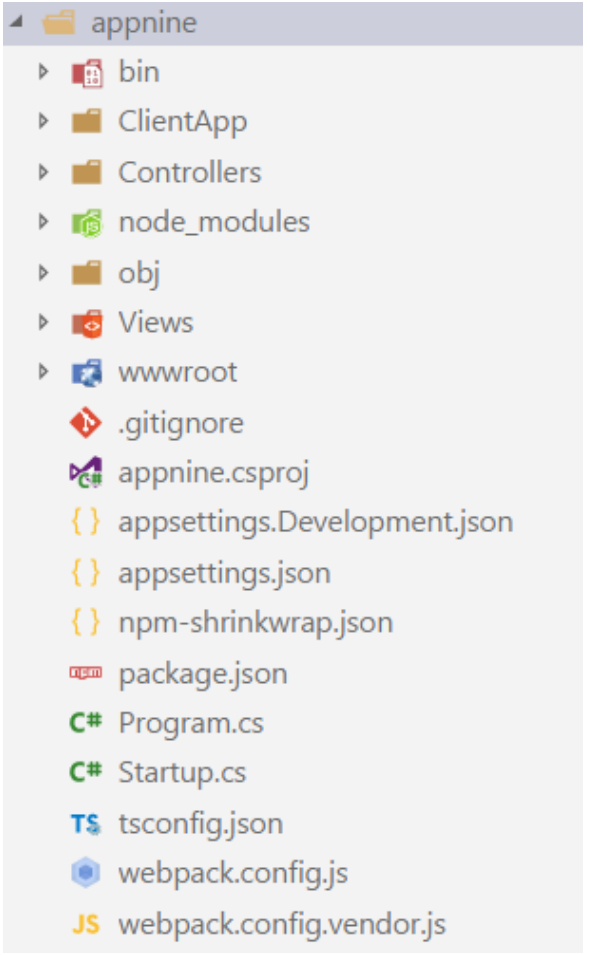
- Creates the following application structure
- Now run npm install –save



ASP.NET CORE 2.0 React Application:build

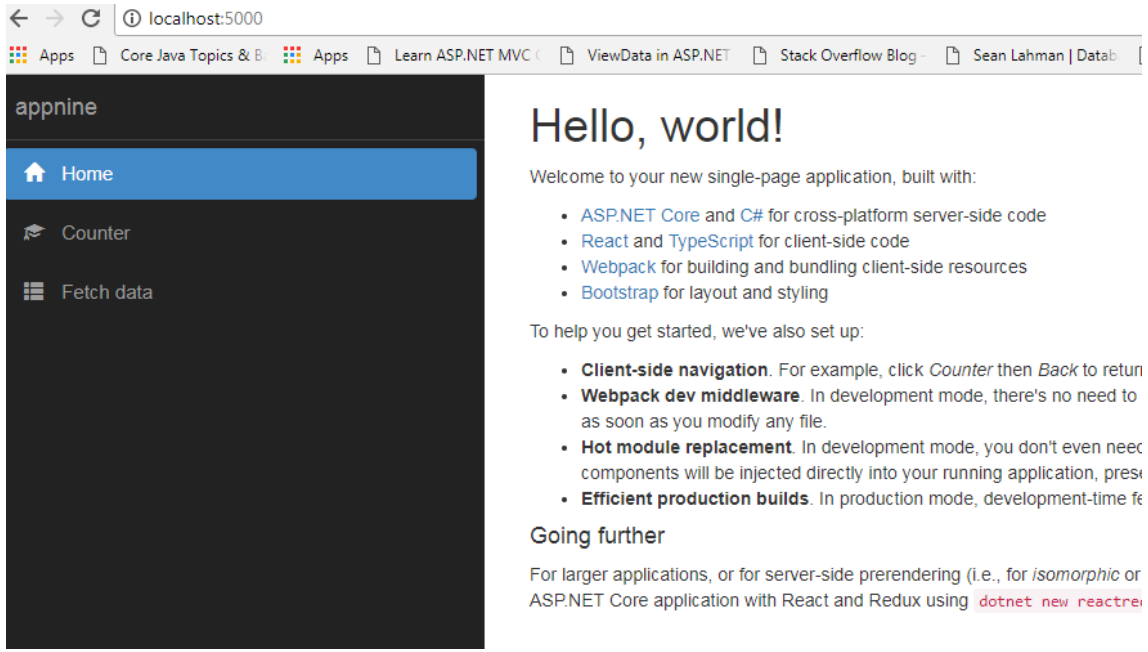
- Building your ASP.NET CORE 2.0 React Application
  - Dotnet build

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appnine> dotnet build
Microsoft (R) Build Engine version 15.5.180.51428 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 209.03 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-ap
oj.
  Restore completed in 160.58 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-ap
```



ASP.NET CORE 2.0 React Application:run

- Executing your ASP.NET Core 2.0 React Application
  - Dotnet run




```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appnine> dotnet run
Hosting environment: Production
Content root path: H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appnine
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```

ASP.NET MVC Core 2.0 (ReactJS +Redux)

EXERCISE DEMO: 2.6

LEARNING OUTCOMES

- Creating your application with ASP.NET Core with React.js with Redux
- Understanding the application structure
- Building and executing your application.

ASP.NET CORE MVC 2.0 ReactRedux Application

:Create

1

- Create a new solution using dotnet core ASP.NET Core with ReactJS Redux
- **Dotnet new reactredux**

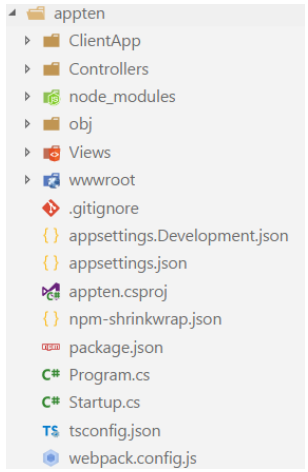
```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appten> dotnet new reactredux
The template "ASP.NET Core with React.js and Redux" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appter
  Restoring packages for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appten\ap
  Restore completed in 239.91 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenari
.
  Generating MSBuild file H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appten\c
g.props.
  Generating MSBuild file H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appten\c
g.targets.
  Restore completed in 3.27 sec for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenaric
```

2

- Runs post-creation actions
- Dotnet restore (in dotnet 1.x we had to explicitly call this)

3

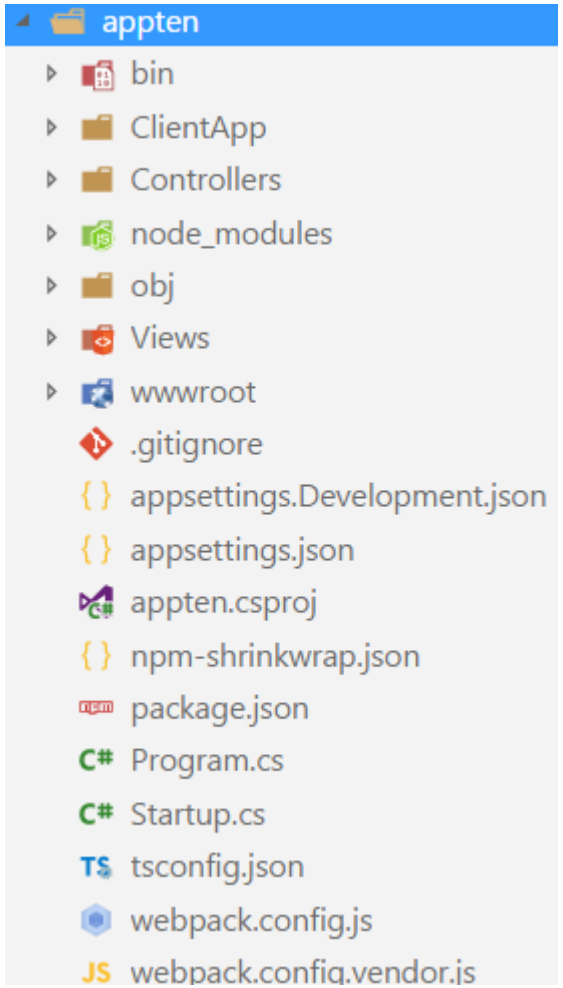
- Creates the following application structure
- Now run npm install –save



ASP.NET CORE MVC 2.0 ReactRedux Application :build

- Building your ASP.NET MVC 2.0 ReactRedux Application
  - Dotnet build

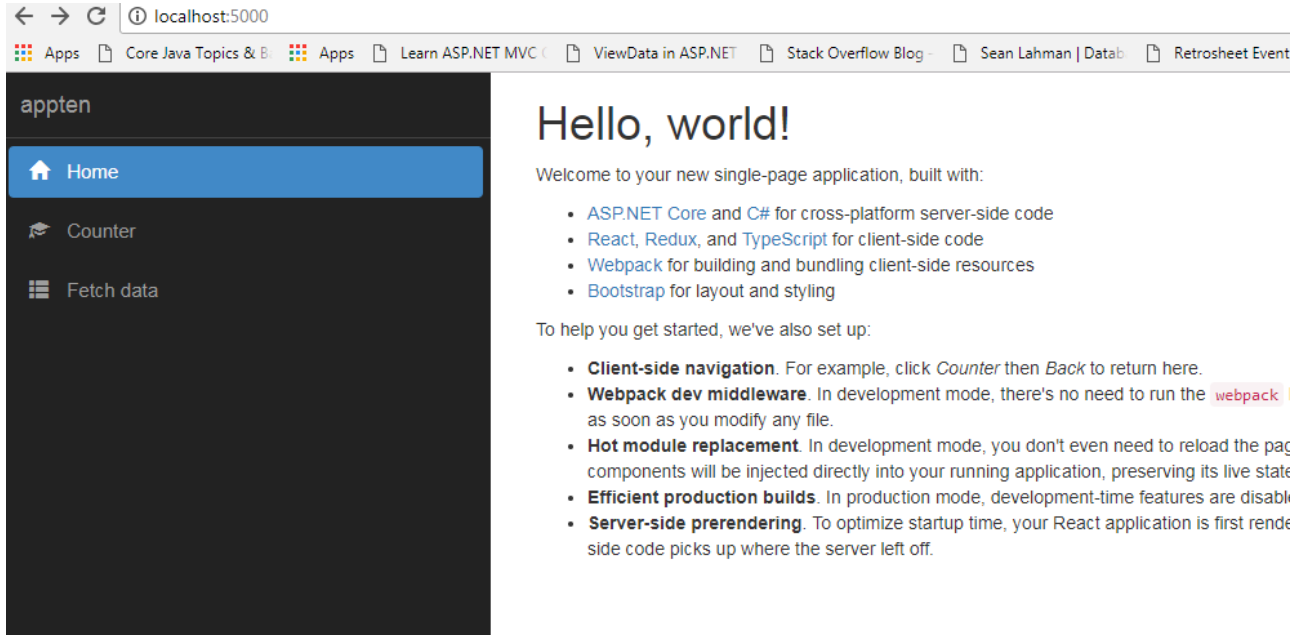
```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appten> dotnet build
Microsoft (R) Build Engine version 15.5.180.51428 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 146.56 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-app
.
  Restore completed in 153.93 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-app
.
  appten -> H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appten\bin\Debu
  v8.1.3
  Performing first-run Webpack build...
  Hash: 029fc5b0d9182e0674fbbfd69e4414711995fa51
```



ASP.NET CORE MVC 2.0 REACTREDUX:RUN

- Executing your ASP.NET CORE 2.0 reactredux
  - Dotnet run

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appten> dotnet run
Hosting environment: Production
Content root path: H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appten
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```



ASP.NET WEBAPI Core 2.0 (Razor)

EXERCISE DEMO: 2.7

LEARNING OUTCOMES

- Creating an ASP.NET Core WebAPI 2.0 application
- Understanding the application structure
- Building and executing the ASP.NET Core WebAPI Application

ASP.NET CORE 2.0 WEBAPI:CREATE

1

- Create a new solution using dotnet core ASP.NET Core with web api
- **Dotnet new webapi**

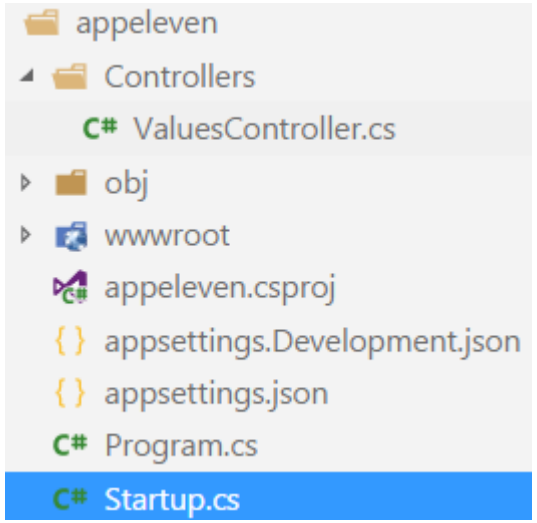
```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appeleven> dotnet new webapi
The template "ASP.NET Core Web API" was created successfully.
This template contains technologies from parties other than Microsoft, see https://aka.ms/templa

Processing post-creation actions...
Running 'dotnet restore' on H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appel

  Restoring packages for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appeleve
  Restore completed in 182.77 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenar
csproj.
```

2

- Runs post-creation actions
- Dotnet restore (in dotnet 1.x we had to explicitly call this)

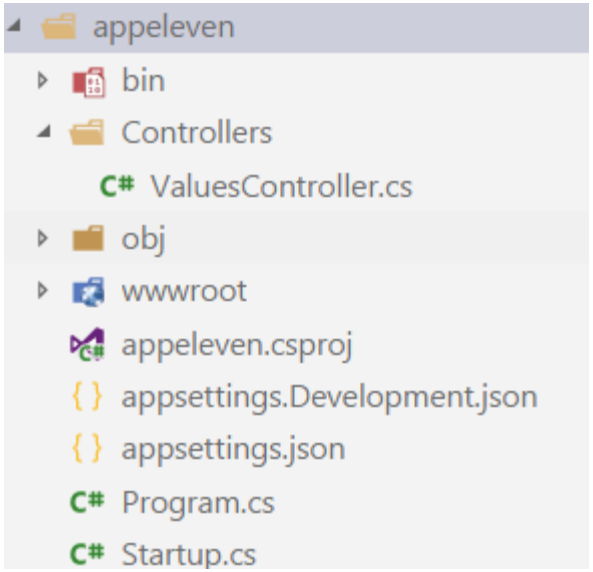
3

- Creates the following application structure



ASP.NET CORE 2.0 WEBAPI:BUILD

- Building your ASP.NET CORE 2.0 webapi
  - Dotnet build

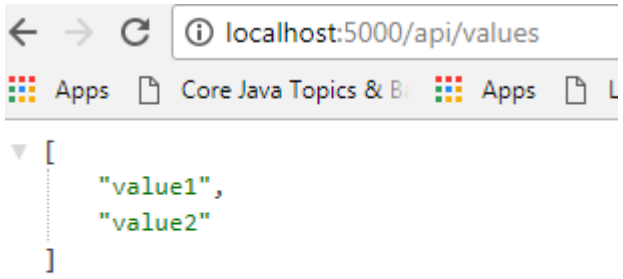



```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appeleven> dotnet build
Microsoft (R) Build Engine version 15.5.180.51428 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 212.5 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appsc
sproj.
  Restore completed in 226.98 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-apps
csproj.
  appeleven -> H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appeleven\bi
n.dll

Build succeeded.
```

ASP.NET CORE 2.0 WEBAPI:RUN

- Executing your ASP.NET Core 2.0 webapi
  - Dotnet run

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appeleven> dotnet run
Hosting environment: Production
Content root path: H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appeleven
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```



ASP.NET MVC Core 2.0 (Authentication)

EXERCISE DEMO: 2.8

LEARNING OUTCOMES

- Creating an ASP.NET Core MVC with Authentication Individual
- Understanding the application structure
- Building and executing the ASP.NET Core MVC 2.0 Application

ASP.NET CORE 2.0 MVC WITH INDIVIDUAL AUTH :CREATE

1

- Create a new solution using dotnet core ASP.NET Core MVC with individual authentication
- **dotnet new mvc --auth Individual**

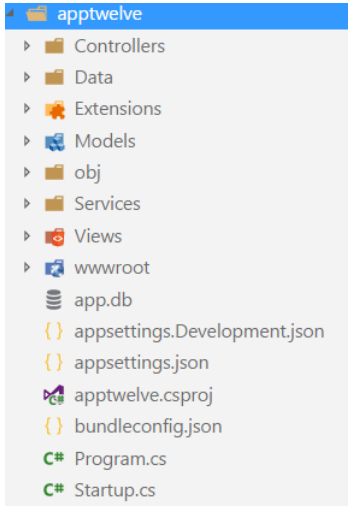
```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\apptwelve>
ve> dotnet new mvc --auth Individual
The template "ASP.NET Core Web App (Model-View-Controller)" was created s
uccessfully.
This template contains technologies from parties other than Microsoft, se
e https://aka.ms/template-3pn for details.

Processing post-creation actions...
Running 'dotnet restore' on H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnet
core-appscenarios\apptwelve\apptwelve.csproj...
  Restoring packages for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcor
```

2

- Runs post-creation actions
- Dotnet restore (in dotnet 1.x we had to explicitly call this)

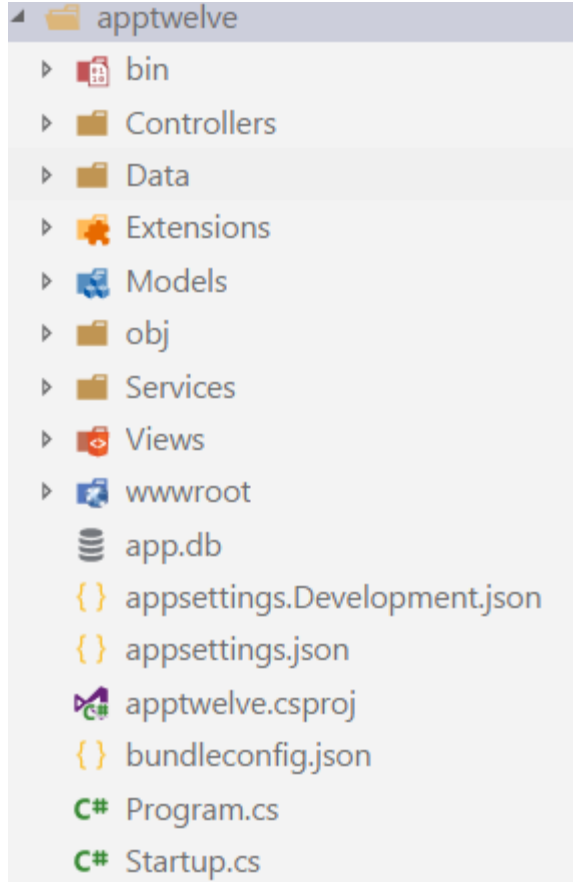
3

- Creates the following application structure
- Now run npm install –save



ASP.NET CORE 2.0 MVC WITH INDIVIDUAL AUTH :BUILD

- Building your ASP.NET Core 2.0 MVC application with individual authentication
  - dotnet new mvc --auth Individual

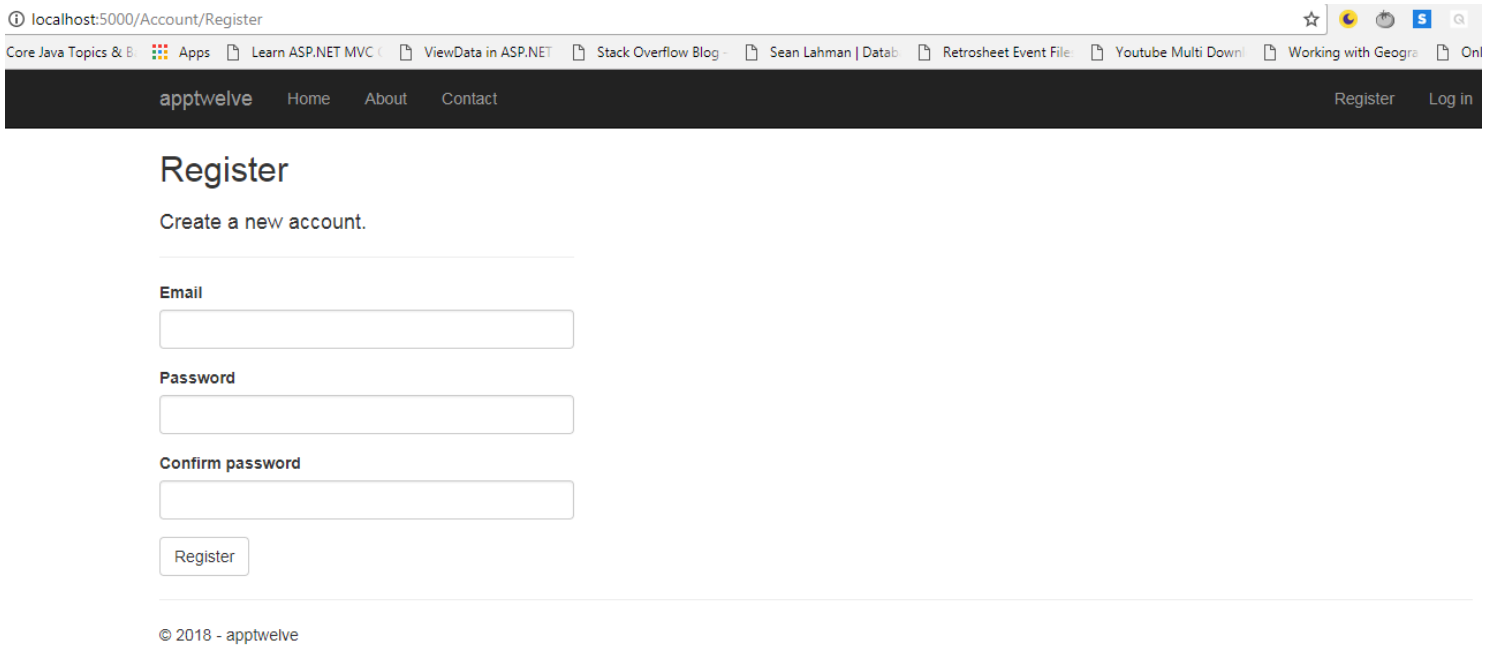



```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\apptwelve> dotnet build
Microsoft (R) Build Engine version 15.5.180.51428 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 256.89 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-apps
csproj.
```

ASP.NET CORE 2.0 MVC WITH INDIVIDUAL AUTH :RUN

- Executing your ASP.NET Core 2.0 MVC with individual Authentication
  - Dotnet run

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\apptwelve> dotnet run
Hosting environment: Production
Content root path: H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\apptwelve
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```



ASP.NET MVC Core 2.0 (Razor with Authentication)

EXERCISE DEMO: 2.9

LEARNING OUTCOMES

- Creating an ASP.NET Core MVC 2.0 Razor View Engine with Authentication Individual
- Understanding the application structure
- Building and executing the ASP.NET Core MVC 2.0 Application

ASP.NET CORE 2.0 Razor WITH INDIVIDUAL AUTH :CREATE

1

- Create a new solution using dotnet core ASP.NET Core with razor view and individual authentication.
- **dotnet new razor --auth Individual**

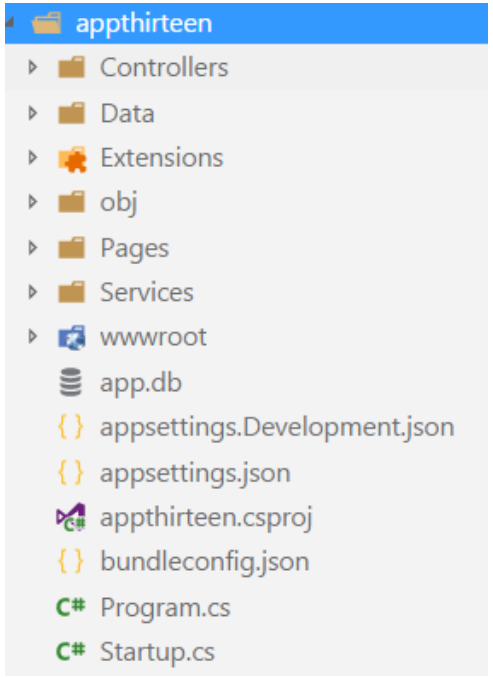
```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appthirteen> dotnet new razor --auth Individual
The template "ASP.NET Core Web App" was created successfully.
This template contains technologies from parties other than Microsoft, see https://aka.ms/template-3pn for details.

Processing post-creation actions...
Running 'dotnet restore' on H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appthirteen\appthirteen.cs
j...
  Restoring packages for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appthirteen\appthirteen.cspr
```

2

- Runs post-creation actions
- Dotnet restore (in dotnet 1.x we had to explicitly call this)

3

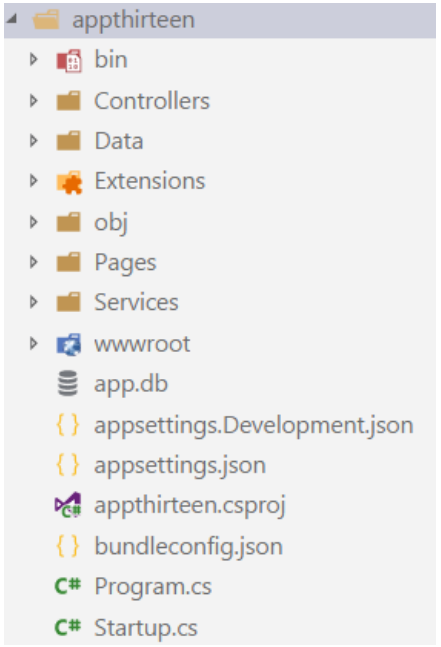
- Creates the following application structure
- Now run npm install –save



ASP.NET CORE 2.0 Razor WITH INDIVIDUAL AUTH :BUILD

- Building your ASP.NET CORE 2.0 razor with individual authentication
  - Dotnet build

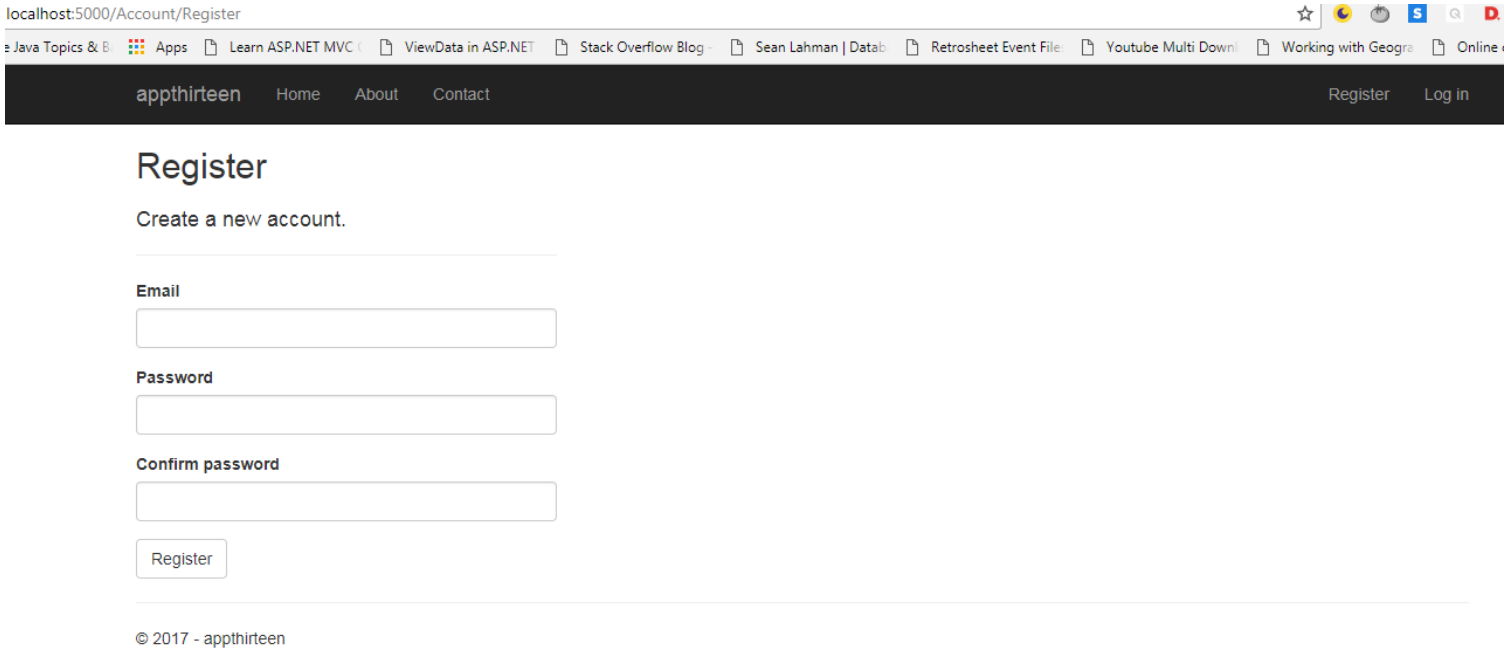
```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appthirteen> dotnet build
Microsoft (R) Build Engine version 15.5.180.51428 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 173.47 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appsc
een.csproj.
  Restore completed in 173.53 ms for H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appsc
een.csproj.
```



ASP.NET CORE 2.0 Razor WITH INDIVIDUAL AUTH :RUN

- Executing ASP.NET Core 2.0 Razor with individual authentication
  - Dotnet run

```
PS H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appthirteen> dotnet run
Hosting environment: Production
Content root path: H:\awase-harddisk\CT\C#2017\DOTNETCORE\dotnetcore-appscenarios\appthirteen
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```



SECTION -XVIII

SIGNAL R

SECTION -XIX

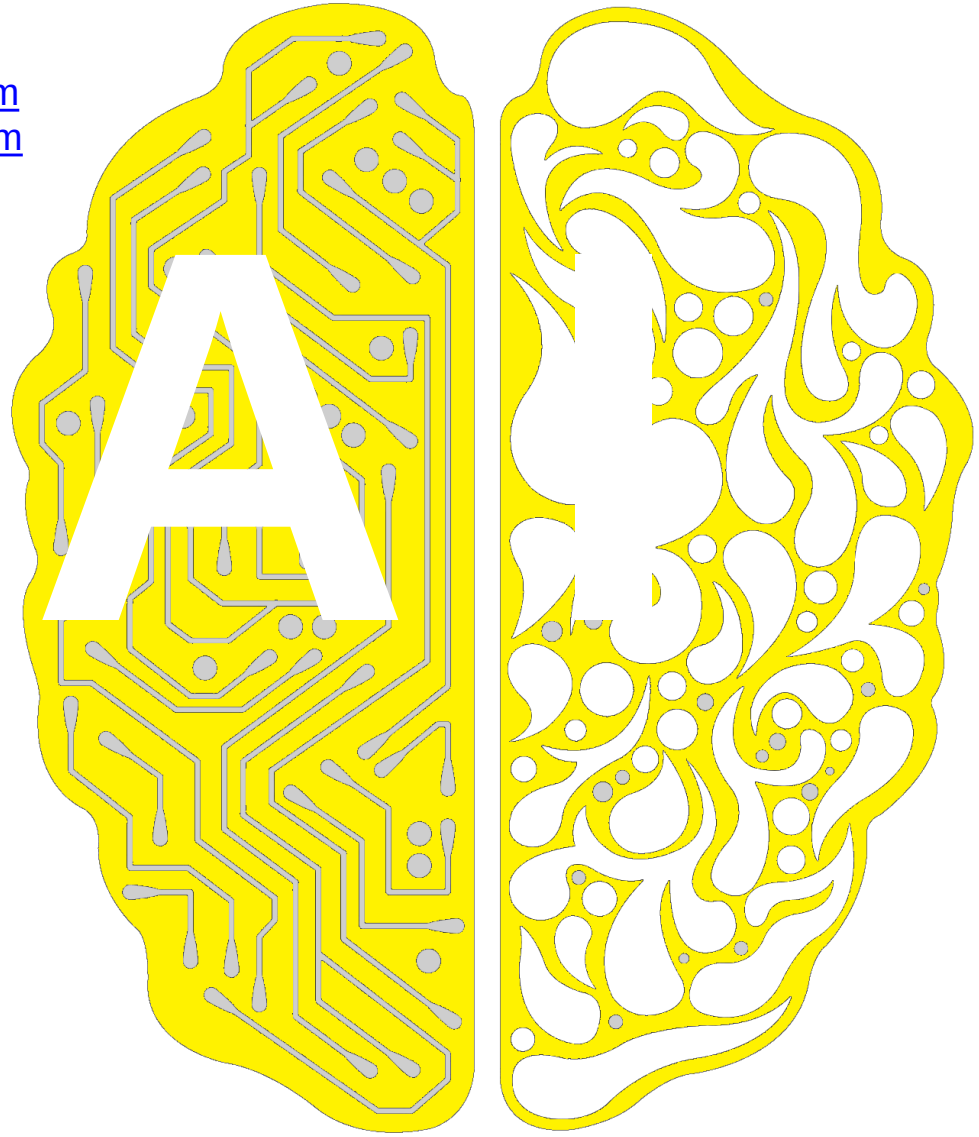
ASP.NET CORE

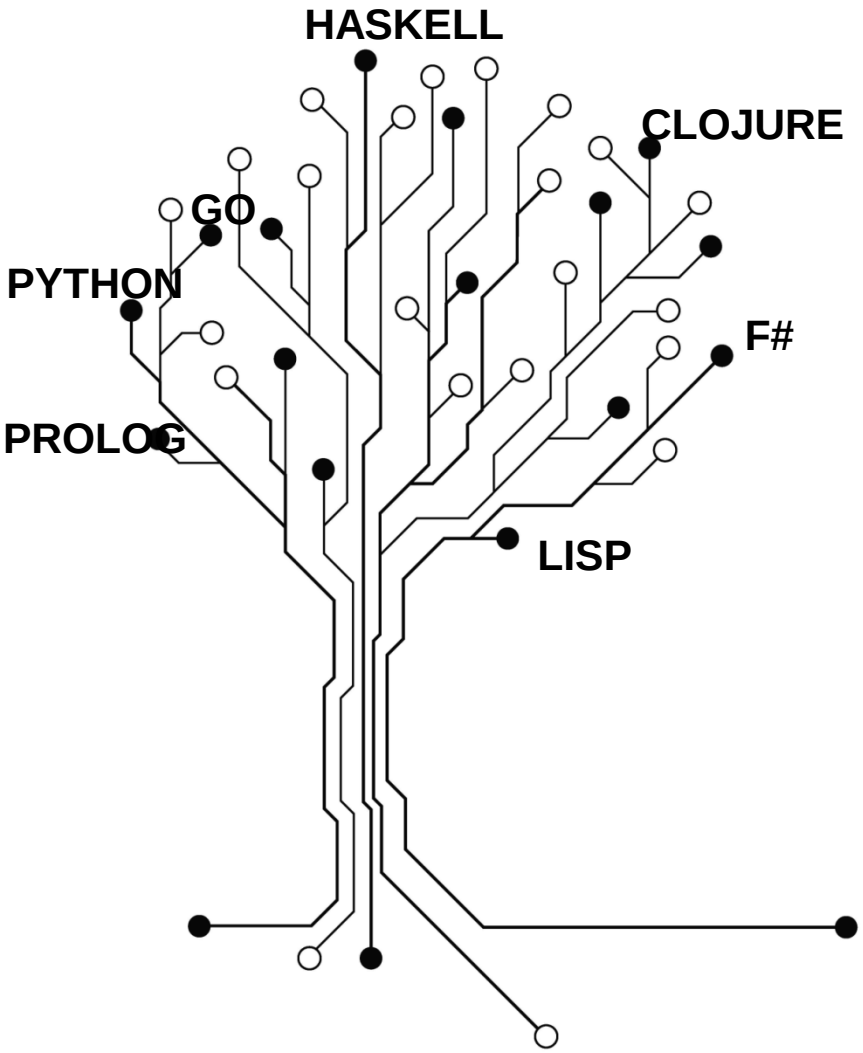
TPRI-SYCLIQ PROGRAMS OVERVIEW

EMPOWERING YOU

Artificial Intelligence

We also train on AI Stack
Reach out to us sak@sycliq.com
or sak@territorialprescience.com
www.territorialprescience.com
www.sycliq.com
+91.9035433124

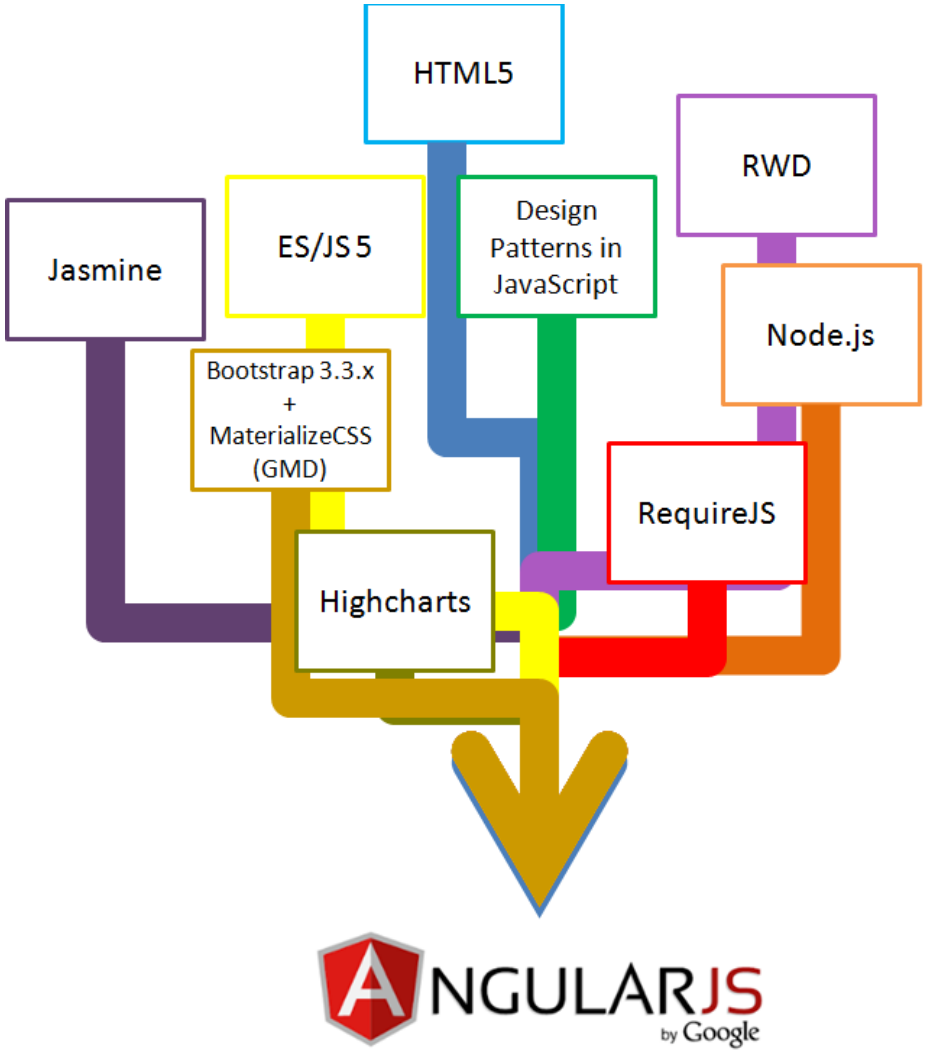




Angular 2.x/Angular JS 1.5.x

Dr. Syed Awase 2016 Session Feedbacks: http://bit.ly/2hhNg58

Reach out to sak@territorialprescience.com/+91.9035433124



Original Series

Dr. Syed Awase also offers Machine learning Stack, R Statistical Stack, .NET Stack, Java Stack, RaspberryPi Stack. Get the pulse of performance from here http://bit.ly/2hhNg58

Dr. Syed Awase 2016 Session Feedbacks: http://bit.ly/2hhNg58

Reach out to sak@territorialprescience.com/+91.9035433124

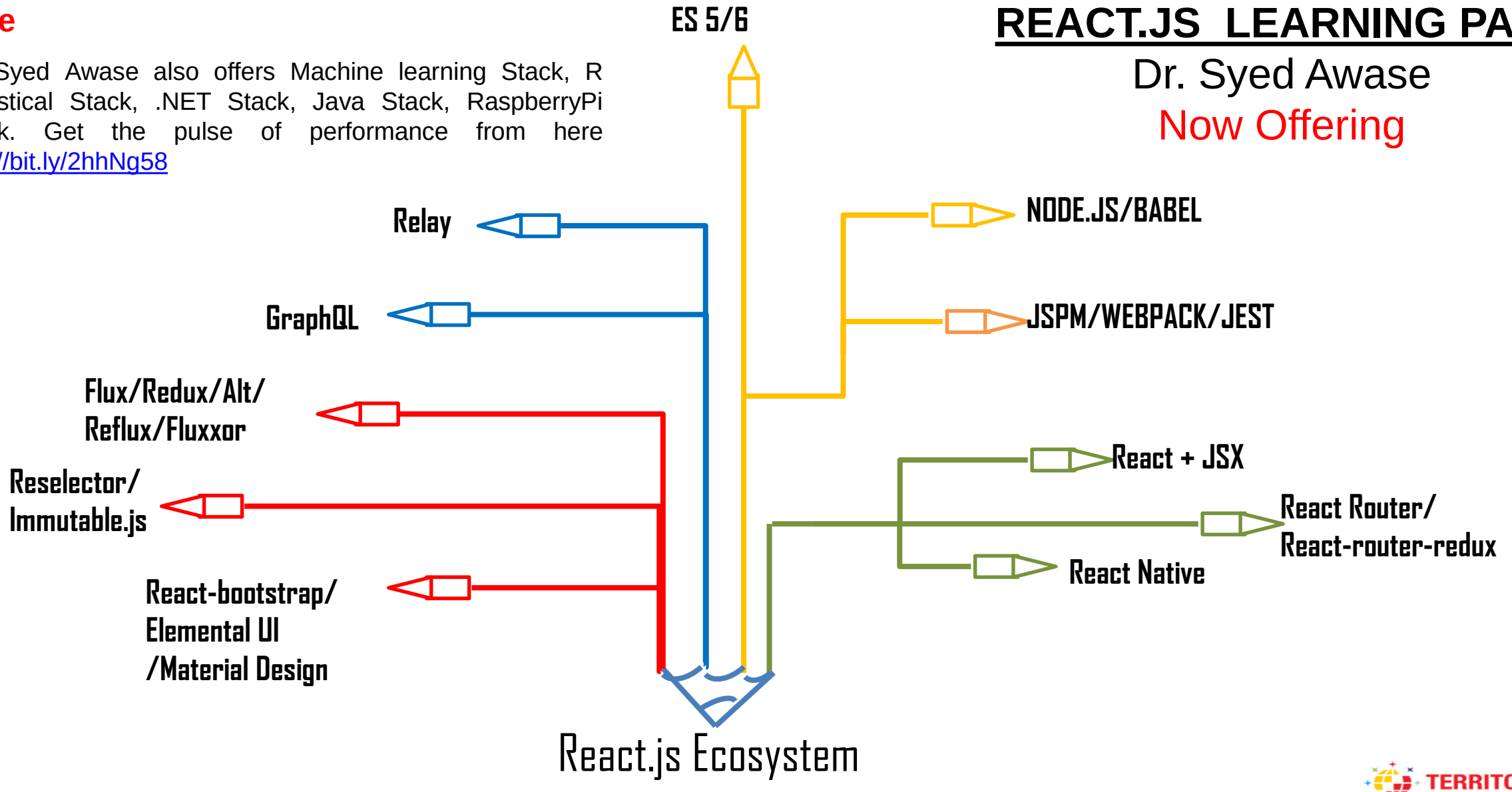
Dr. Syed Awase also offers Machine learning Stack, R Statistical Stack, .NET Stack, Java Stack, RaspberryPi Stack. Get the pulse of performance from here http://bit.ly/2hhNg58

Dr. Syed Awase also offers Machine learning Stack, R Statistical Stack, .NET Stack, Java Stack, RaspberryPi Stack. Get the pulse of performance from here http://bit.ly/2hhNg58

REACT.JS LEARNING PATH

Dr. Syed Awase

Now Offering

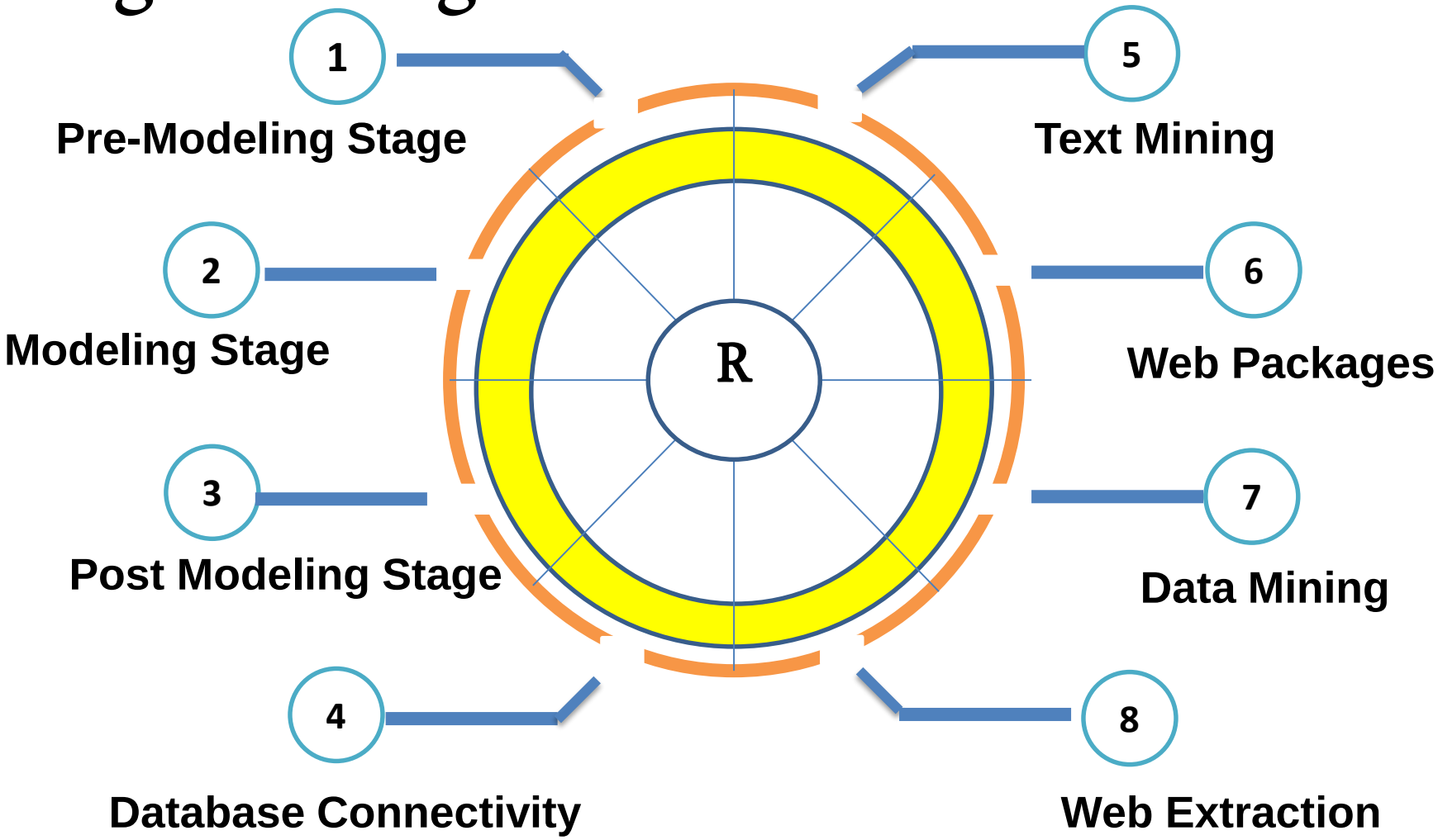


Original Series

R-Statistical Programming

Dr. Syed Awase 2016 Session Feedbacks: http://bit.ly/2hhNg58

Reach out to
sak@territorialprescience.com/
+91.9035433124



Thank You

We also provide Code Driven Open House Trainings : sak@territorialprescience.com or sak@sycliq.com

Java Technologies

- Core Java
- Hibernate
- Spring Framework
- Play Framework
- Hadoop
- Groovy & Grails

Microsoft Technologies

- C# Core
- Entity Framework
- MVC 5/6
- Web Api
- OWIN/KATANA
- WCF
- WPF

Python

- Python
- Django
- Flask
- Numpy
- Scipy
- Machine Learning

Data Science

- R Statistical Programming
- Julia

SQL and NoSQL

- Oracle
- PostgreSQL
- MSSQL
- MongoDB
- Neo4j
- Redis
- Firebase
- Apache Cassandra

Client-Side Frameworks

- Angular JS 1.5.x
- Angular 2.4.x
- React JS
- KnockOut JS
- VueJS
- Backbone JS
- EMBER JS
- Hapi JS
- METEORJS
- MEANJS
- Coffeescript
- Dart

Others

- LISP
- CLOJURE
- RUST
- GO
- Raspberry PI
- Coming Soon
  - PHP
  - RoboticOS