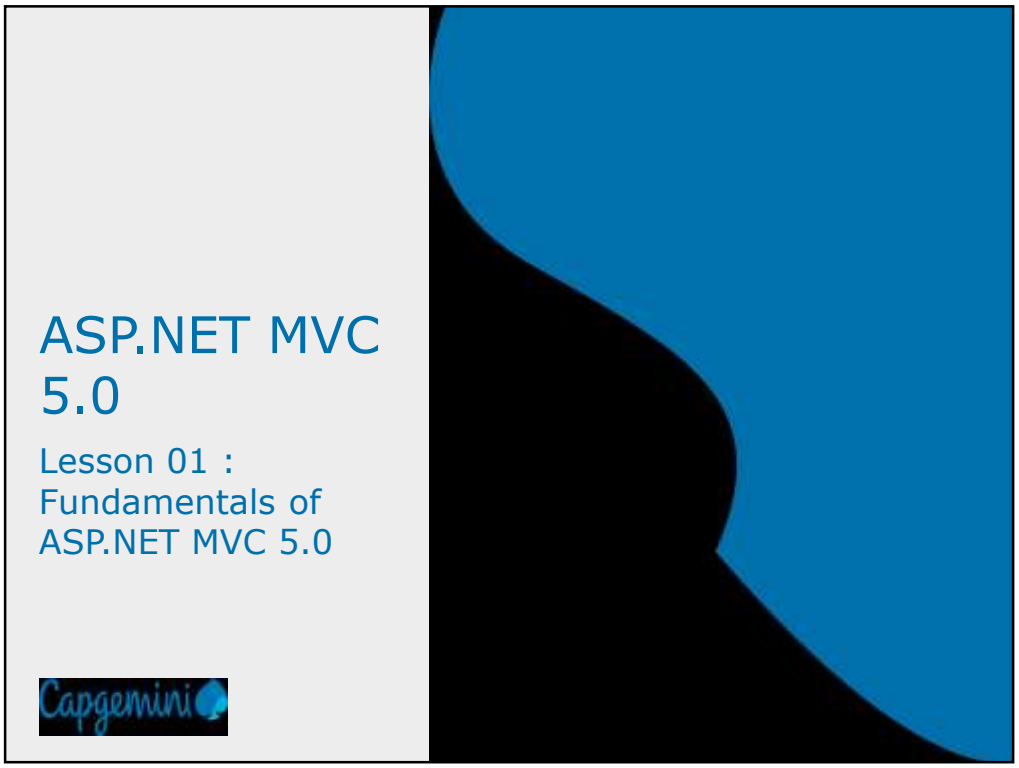

ASP.NET MVC
5.0
Lesson 01 :
Fundamentals of
ASP.NET MVC 5.0
Capgemini

Lesson Objectives

- Understand the Fundamentals of MVC Framework
- Understand ASP.NET MVC Model
  - Model
  - View
  - Controller
- Understand the difference between ASP.NET & ASP.NET MVC
- Routing in ASP.NET MVC
- Razor View in ASP.NET MVC
- Scaffolding in MVC

1.1 ASP.NET MVC Introduction

ASP.NET MVC Framework

- ASP.NET MVC is a Microsoft framework for building web applications that use a model-view-controller pattern
- It supports functionalities like Caching, Master pages, HTTP Handlers, Membership and Custom Providers.
- It does not replace web forms. It 's an alternative project type.
- It achieves clean URLs and clean HTML
- We can extend this framework using pluggable view engines and controller factories
- MVC maintains a strict separation of concerns .

Description

- A framework
  - Is a structural piece of software
  - Attempts to provide a platform upon which web applications can be quickly built
  - Will try to automate all the common & tedious tasks of the domain
- The ASP.NET MVC framework is a lightweight, highly testable presentation framework that (as with Web Forms-based applications) is integrated with existing ASP.NET features, such as master pages and membership-based authentication.
- The MVC framework is defined in the System.Web.Mvc namespace and is a fundamental, supported part of the System.Web namespace.

A framework is a structural piece of software since the main focus of framework is the structure rather then functional requirements. It attempts to provide a platform upon which web applications can be quickly built.

It also tries to automate all common and tedious tasks of the domain. A good framework will provide mechanisms for convenient and perhaps automatic solutions to the common tasks improving the developers productivity.

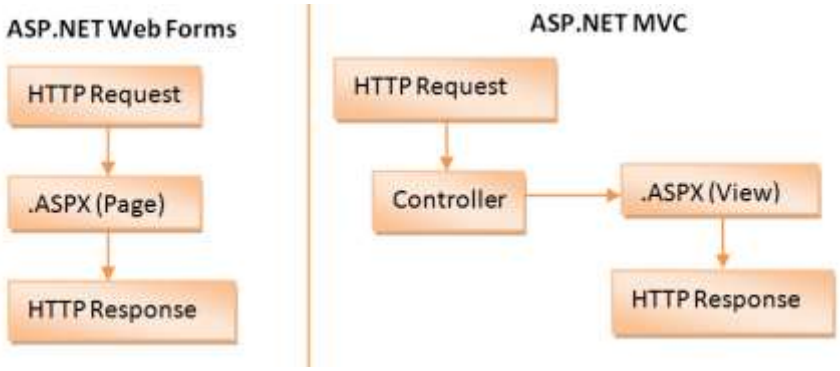
Tasks such as validation, separate of business and data layer and also presentation rendering can be taken care of by the framework.

In other words, a framework is a piece of structural software that provides automation of common tasks of the domain and provides an architectural solution that can be easily inherited by applications which are implemented on the framework.

You can do away with framework provided your application is simple. But if the application is complex or is an enterprise application involving lot of things then using a framework would be good solution. Also by using a framework you can take care of all the common domain tasks.

1.1 ASP.NET MVC Introduction

ASP.NET vs. ASP.NET MVC



ASP.NET	ASP.NET MVC
ASP.NET Web Form follows a traditional event driven development model	ASP.NET MVC follows MVC pattern based development model.
It has server controls.	It has HTML helpers.
ASP.NET Web Form has state management (like view state, session)techniques	ASP.NET MVC has no automatic state management techniques
ASP.NET Web Form follows Web Forms Syntax	ASP.NET MVC follow customizable syntax (Razor as default)
ASP.NET Web Form views are tightly coupled to logic	ASP.NET MVC, Views and logic are kept separately.
ASP.NET Web Form has User Controls for code reusability.	ASP.NET MVC has Partial Views for code re-usability.
ASP.NET Web Form has built-in data controls and best for rapid development with powerful data access.	ASP.NET MVC is lightweight, provide full control over markup and support many features that allow fast & agile development.
ASP.NET Web Form has master pages for consistent look and feel	ASP.NET MVC has layouts for consistent look and feel

A Comparison: ASP.NET MVC & ASP.NET

- ASP.Net MVC benefits
  - Enables clean separation of concerns
  - Full control over HTML and easy integration with JavaScript
  - Can follow a Test Driven Development approach
  - No ViewState and Postback
  - Follows the Stateless nature of web
- Benefits of ASP.Net Web Forms
  - Provides RAD development model
  - Provides rich controls
  - Familiar model for windows form developers
  - Cannot do testing easily

ASP.Net MVC is an alternative approach to webforms and not a replacements to webforms. In fact they both can co-exist together. They both have their own benefits and as a developer whichever is comfortable and suitable to the requirement can be chosen.

There are some aspects on which these two can be compared.

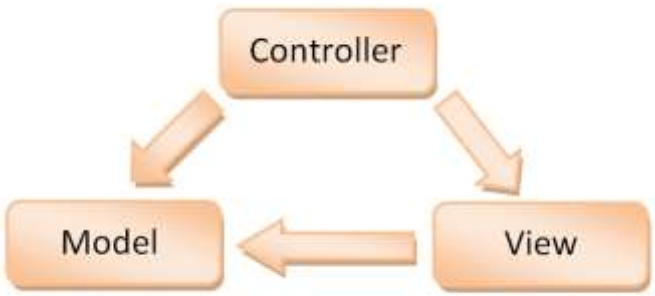
ASP.Net MVC simplifies the complex parts of ASP.Net Web Forms still retaining the flexibility and power of the ASP.Net platform. Since it follows the MVC pattern there is a clean separation of concerns between the View, Controller and Model. The webform events are replaced with controller actions by the MVC model. The MVC model also facilitates unit testing and provides more control over HTML. It does not use ViewState, Postback, Server Controls or Server based forms that enable full control over the application and html rendered by the views.

ASP.Net Webform was designed keeping in mind a similar concept windows form model development for web application development. Web Forms used the event driven approach and also introduced ViewState and Postaback. Thus breaking the stateless nature

1.1 ASP.NET MVC Introduction

MVC Model

- MVC design pattern achieve the separation of concerns in the UI Layer of an application. i.e. MVC doesn't have any impact on BL or DAL, it is strictly concerned on PL.



Model might be a BO or simple DTO. Model is responsible for bringing data to View. Model comes from controller.

View is responsible for rendering UI(HTML), it does not perform Business calculation. View relies on Model to bring Data.

Controller is responsible for built the model and selects a view to render. Controller also receives HTTP User request

1.1 ASP.NET MVC Introduction

Model

- These are the classes that represent the domain you are interested in.
- These domain objects often encapsulate data stored in a database as well as code used to manipulate the data and enforce domain-specific business logic.
- With ASP.NET MVC, this is most likely a Data Access Layer of some kind using a tool like Entity Framework or NHibernate combined with custom code containing domain-specific logic.
- MVC project template creates a Model Folder.
- We can perform validations in model using Data Annotations, Client Validations and Remote Validation

1.1 ASP.NET MVC Introduction

Controller

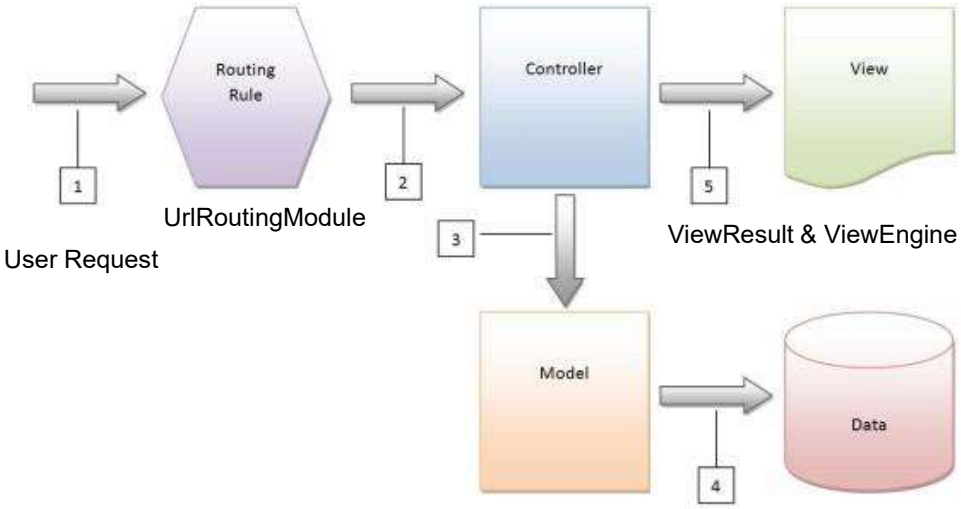
- This is a special class that manages the relationship between the View and Model.
- It responds to user input, talks to the Model, and it decides which view to render (if any).
- In ASP.NET MVC, this class is conventionally denoted by the suffix Controller.
- Any public methods(actions) under controller are accessible via URL.
- Those methods will be invoked once the proper route is determined.
- Controller returns an ActionResult through actions which tells the framework what to do next.

1.1 ASP.NET MVC Introduction

View

- View Folder is the recommended location for the views. Subfolders needs to be created for every controller.
- Views under shared folder can be used by multiple controllers.
- There are 2 types of views Webform view and razor view.
- Views with .cshtml, .vbhtml are razor other than that is webForm views(.aspx, .ascx, .master)

Understanding MVC application lifecycle



Requests to an ASP.NET MVC-based Web application first pass through the UrlRoutingModule object, which is an HTTP module.

This module parses the request and performs route selection.

The UrlRoutingModule object selects the first route object that matches the current request. (A route object is a class that implements RouteBase, and is typically an instance of the Route class.)

If no routes match, the UrlRoutingModule object does nothing and lets the request fall back to the regular ASP.NET or IIS request processing.
From the selected Route object, the UrlRoutingModule object obtains the IRouteHandler object that is associated with the Route object.

Typically, in an MVC application, this will be an instance of MvcRouteHandler.

The IRouteHandler instance creates an IHttpHandler object and passes it the IHttpContext object.

By default, the IHttpHandler instance for MVC is the MvcHandler object. The MvcHandler object then selects the controller that will ultimately handle the request.

1.1 ASP.NET MVC Introduction

ASP.NET MVC – Application Folders

- App_Data : This folder is used for storing application data.
- Content : This folder is used for static files like (css, images)
- Controllers : This folder contains controller classes for handling user input and responses.
- Models : This folder contains the classes(entity) that represents the application models.
- Views: This folder contains the files which is used to render UI of the application
- Shared : This folder is used to store the views which can be shared between controllers (master pages and layouts)
- Scripts : This folder stores JavaScript files for the application.

Journey till ASP.Net MVC5

- ASP.NET MVC 1.0
- ASP.Net MVC 2
  - UI helpers with customizable template
  - Strongly-typed HTML helpers
  - New helper functions, utilities and API enhancements
- ASP.Net MVC3
  - Expressive views including the new Razor View Engine
  - Streamline Validation with improved Model Validation
  - Rich JavaScript support with non-interfering JavaScript,jQuery Validation and JSON binding.
- ASP.Net MVC4
  - Mobile Project Template
  - Task support for Asynchronous Controllers
  - Jquery Mobile
  - Bundling and Minification
  - ASP.NET Web API

ASP.Net MVC has seen 3 major releases in about 2 years. Let us see the journey to ASP.Net MVC3.

The first version of ASP.Net MVC was conceptualized by Scott Guthrie in February 2007. The original prototype was called as scalene. Going through a lot of developments and preview releases, ASP.Net MVC 1 was officially released in March 2009.

In March 2010, ASP.Net MVC 2 was released. This version included some main features like:

UI helpers with customizable templates

Attribute based model validation on both client and server

Improved Visual Studio tooling

Strongly typed HTML helpers

Based on the feedback from first release some improvements and API enhancements were made such as:

Support for rendering subsections of page/site

Lots of new helper functions and utilities

ASP.Net MVC 3 mostly called as MVC 3 , released in January 2010 came up with new features like:

Expressive Views including new Razor Engine View

.Net 4 Data Annotation support

Rich JavaScript support with unobtrusive JavaScript, jQuery Validation and JSON binding.

Streamlined validation with improved Model validation

Global Action Filters and also support for NuGet

Journey till ASP.Net MVC5

- ASP.NET MVC 5.0
  - Scaffolding
  - ASP.NET Identity
  - One ASP.NET
  - Bootstrap
  - Attribute Routing
  - Filter Overrides
- Some of these features like ASP.NET Identity etc are not exactly new features of the core MVC 5 framework, but are worth looking at, as they directly affect/change the way we create MVC 5 applications.

The MVC 4 release built on a pretty mature base and is able to focus on some more advanced scenarios.

Some top features include:

ASP.NET Web API

Enhancements to default project templates

Mobile project template using jQuery Mobile

Display modes

Task support for asynchronous controllers

Bundling and minification

MVC 5 was released along with Visual Studio 2013 in October 2013. The main focus of this release

was on a “One ASP.NET” initiative and core enhancements across the ASP.NET frameworks.

Some of the top features include:

1.2 One ASP.Net

One Asp. Net with Visual Studio 2013

- Visual Studio 2013 provides new unified web development experience
- It provides one single project template for creating ASP.Net applications like Web Forms, MVC, or Web API applications.
- As a result, the dialogs for creating a new MVC application in Visual Studio 2013 have been merged and simplified.

One ASP.Net

In previous versions of MVC, you were faced with a choice every time you created a project. You had to choose between an MVC application, Web Forms application, or some other project type.

After you had made your decision, you were essentially trapped. You could kind of add Web Forms to an MVC application, but adding MVC to a Web Forms application was difficult. MVC applications had a special project type GUID hidden in their csproj file, and that was just one of the mysterious changes you had to make when attempting to add MVC to Web Forms applications.

In MVC 5, that all goes away, because just one ASP.NET project type exists. When you create a new web application in Visual Studio 2013, there's no difficult choice, just a Web application. This isn't just supported when you first create an ASP.NET project; you can add in support for other frameworks as you develop, because the tooling and features are delivered as NuGet packages. For example, if you change your mind later on, you can use ASP.NET Scaffolding to add MVC to any existing ASP.NET application.

1.2 View Engine

Introduction to Razor View Engine

- It is a New view engine where we can generate the output by combining data with template.
- It is easy to use, no ties to ASP.NET Runtime (executes without recompiling).
- We can intermingle C# and HTML in Razor view.

ASPX View vs. Razor View (Code Snippet)

ASPX View

```
<%for (int counter = 1; counter <= 5; counter++)
        {%>
              <div><%:counter %></div>
       <% } %>
```

Razor View

```
@{
        for (int counter = 1; counter <= 5; counter++)
        {
              <div>@counter</div>
        }
    }
```

Razor View Engine

- Razor is the first major update to rendering HTML since ASP.NET 1.0
- The default view engine used in MVC 1 and 2 was commonly called the Web Forms View Engine, because it uses the same ASPX/ASCX/MASTER files and syntax used in Web Forms
- Razor was designed specifically as a view engine syntax.
- It has one main focus: code-focused templating for HTML generation

1.3 Routing

Introduction to Routing

- It is a part of ASP.NET under the namespace System. Web. Routing
- Routing directs incoming request to an MVC controller.
- Routes are defined during the application startup. In ASP.NET MVC 4 routes are configured in Route config. cs under App_Start folder
- Routing Maps the URLs to controller action with parameters

RouteConfig.cs

```
public class RouteConfig
  {
     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 }
        );
     }
  }
```

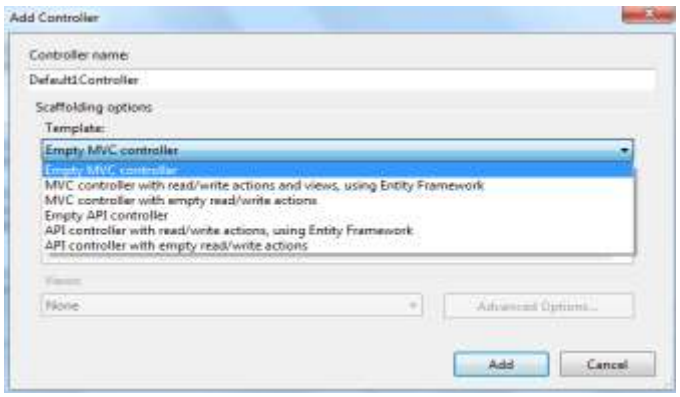
DEMO

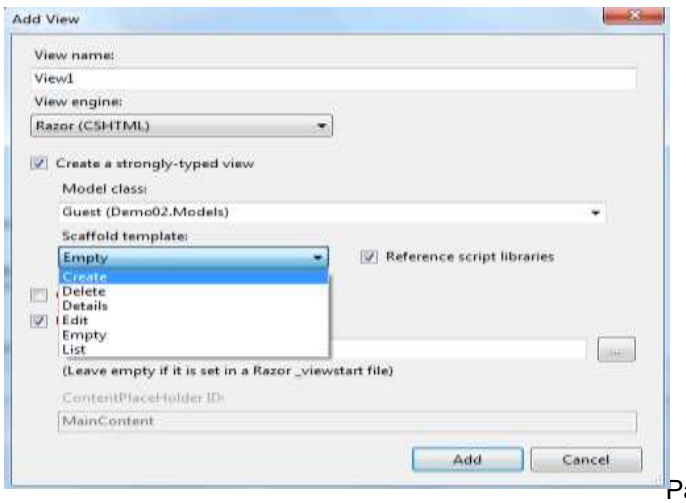
- MVC Basics

1.4 Scaffolding

Scaffolding in MVC

- It is basically a code generation template for MVC application.
- To perform CRUD(Create, Read, update and delete) we need to create model, view for all the functions and controller with each action item. It involves lot of code needs to be written
- Scaffolding provides a feature to just select template and model, which in turn automatically creates controller with all methods and their views.





DEMO

- Guest Phone Book

Summary

- ASP.NET MVC framework built on ASP.NET.
- Using ASP.NET MVC, we can achieve clean URLs
- and clean HTML .
- Model is responsible for bringing data to View.
- View is responsible for rendering UI(HTML).
- Controller is responsible for built the model and selects a view to render.
- Using Razor view engine, HTML generation can be done using code- focused templating approach

Review Question

- Question 1: Public methods on a controller class are known as
  - Endpoints
  - Delegates
  - Handlers
  - Actions
- Question 2: Data Annotations can be used to
  - Mark model properties for serialization
  - Move values from HTTP request into a model
  - Validate Model values
  - All of the above

Review Question

- Question 3: Routes in an MVC application are typically used to map requests to a
  - Model
  - Database
  - Controller
  - View
- Question 4: To strongly-type the Model property of a Razor view, use
  - @type directive
  - @view directive
  - @base directive
  - @model directive