Department of IT
CUK

ASP.NET- MVC

[.NET CORE]

Afaq Alam Khan

Index

- .Net Core
- ASP.Net (Core) MVC
- Model
- View
- Control
- Action Methods & Action Results
- View Data vs view Bag
- Layouts
- Data connectivity & Entity Framework

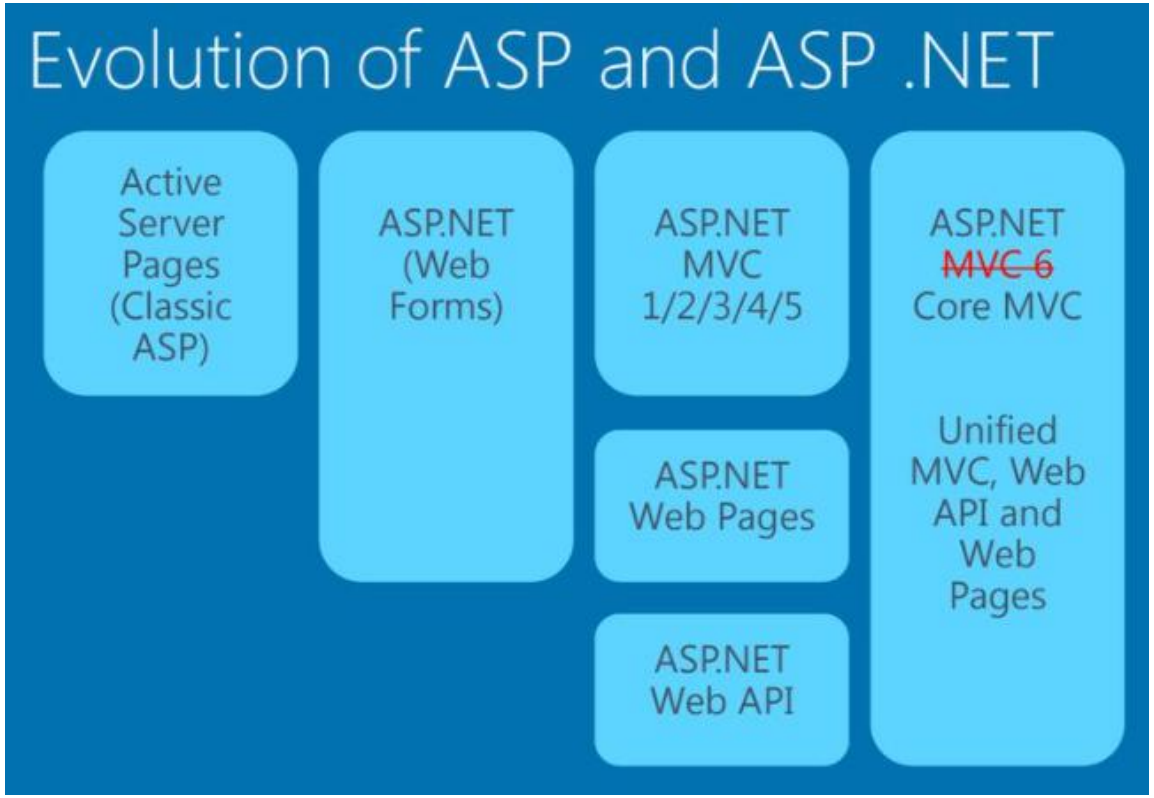
Asp.Net (Core) MVC

- ASP.NET Core is the new web framework from Microsoft. It has been redesignedfrom the ground up to be fast, flexible, modern, and work across different platforms on Windows, Mac and Linux.

- ASP.NET Core is mainly targeted to run on .NET Core platform

- Unified framework for building Web UI and Web APIs.

- Ability to host on IIS or self-host in your own process.



- ASP.NET Core is an open source and cloud-optimized web framework for developing modern web applications that can be developed and run on Windows, Linux and the Mac.

- It includes the MVC framework, which now combines the features of MVC and Web API into a single web programming framework.

- ASP.NET Core apps can run on .NET Core or on the full .NET Framework. ASP.NET core 3.x runs only on .NET Core 3.x, whereas ASP.NET Core 2.x runs on .NET Core 2.x as well as .NET Framework.

- ASP.Net core uses Kestral Webserver

Installation

Framework: Install Dot.Net Core

IDE: Install Visual Studio/ VS Code/ Sublime/ atom …

ASP.Net Core Hosting Model

Once you finish developing an ASP.NET Core web application, you need to deploy it on a server so that end users can start using it. When it comes to deployment , ASP.NET Core offers two hosting models namely

In Process

Out Of Process.

In Out-of-process hosting models, we can either use the Kestrel server directly as a user request facing server or we can deploy the app into IIS which will act as a proxy server and sends requests to the internal Kestrel server. In this type of hosting model we have two options:

Using Kestral Server Only: So in this type Kestrel itself acts as edge server which directly server user requests. It means that we can only use the Kestrel server for our application.

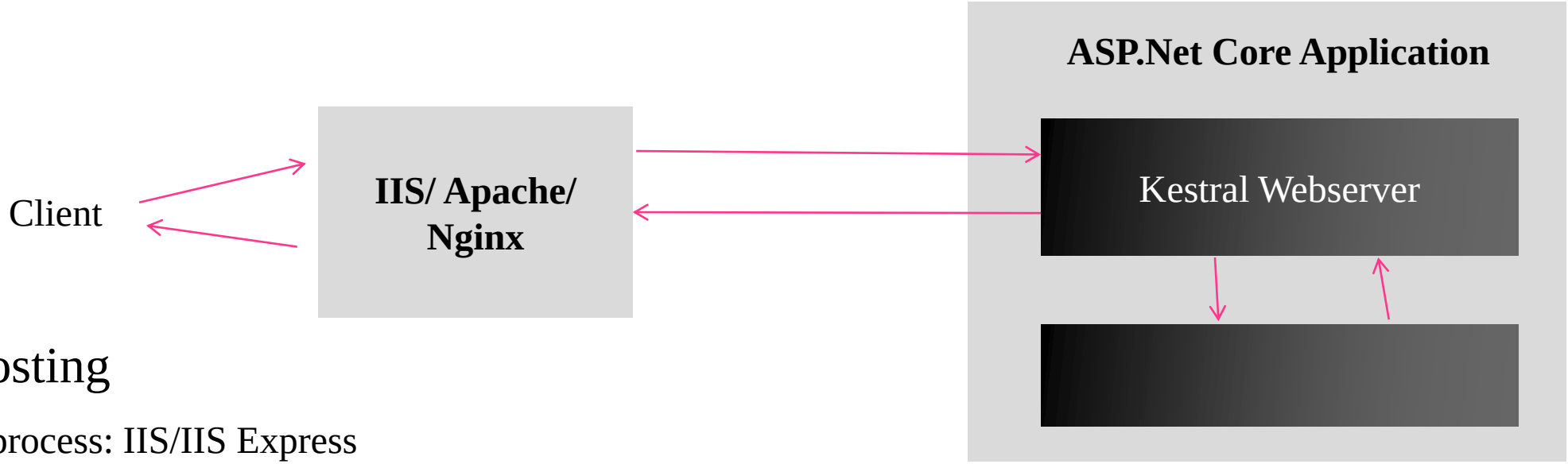
Using a Proxy Server: Due to limitations of the Kestrel server, we can not use this in all the apps. In such cases, we have to use powerful servers like IIS, NGINX or Apache. So, in that case, this server acts as a reserve proxy server which redirects every request to the internal Kestrel sever where our app is running. Here, two servers are running. One is IIS and another is Kestrel. This model is a default model for all the applications implemented before .NET Core 2.2. But there are some of the limitations of using this type such as performance slowness.

ASP.Net (Core) Request Processing

By Default out of process hosting

ASP.Net Core uses Kestral Web Server

Requests are delegated to internal kestral webserver and kestral prepares the responses



Hosting

Inprocess: IIS/IIS Express

Out of Process :

Internal(Kestral)

External(IIS/Apache/Ngenx)

Target Framework Moniker **(TFM)**

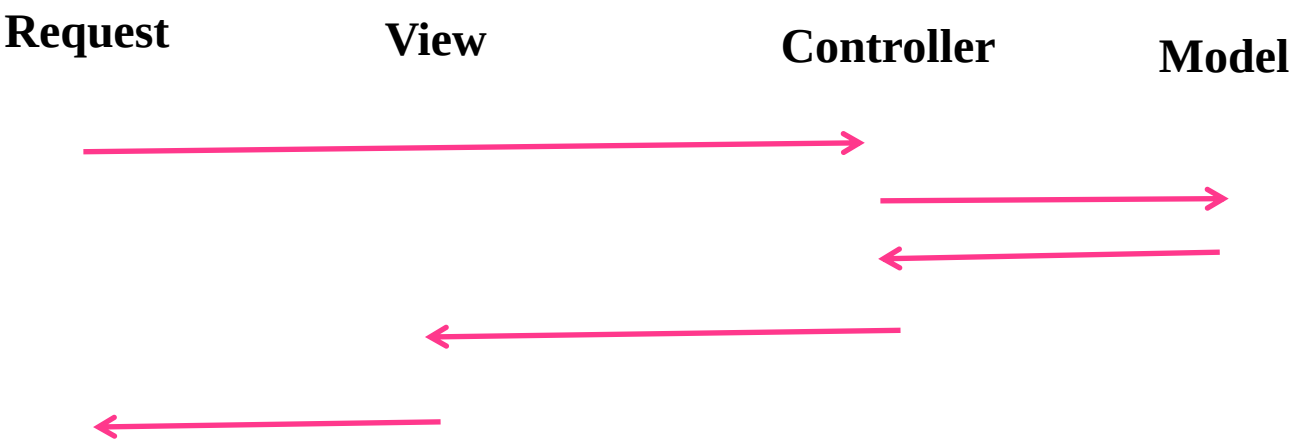
Name	**Abb**	**TFM**
.Net F/W	net	net 451/ net 471…
.Netcore	netcoreapp	netcoreapp1.0/netcoreapp2.0/netcoreapp3.1…

MVC
DESIGN PATTERN

Model–view–controller (usually known as **MVC**) is a software design pattern commonly used for developing user interfaces that divides the related program logic into three interconnected elements.

Popular programming Pattern used for developing Web Applications

Request Flow of MVC



Model: The central component of the pattern. It is the application's dynamic data structure, independent of the user interface.It directly manages the data, logic and rules of the application.

View: Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants**.**

Controller: Accepts input and converts it to commands for the model or view

Controller

A controller in ASP.NET MVC is a class having a set of public methods. These public methods of the controller are called action methods or simple actions. It is these action methods in MVC application which is going to handle the incoming HTTP Requests.

The Controller is the component that is going to receive the incoming HTTP Request and then process that request. While processing the request, the controller does several things. It works with the model. Then it selects a view by passing the model object which renders the user interface from the model data. The view then generates the necessary HTML and then the controller sends HTML back to the client who initially made the request. So we can consider that the Controller is the heart of the MVC application.

By convention, in ASP.NET MVC, the controller classes should reside in the projects root level **Controllers** folder and should inherit from the **System.Web.Mvc.Controller** base class.

Controller can be added in two ways

- Without Scaffolding: Manually create controller
- With Scaffolding: Adds some code by default

Controllers have different methods
- Action methods
- Non Action Methods

Action Methods: Controller methods that respond to various Http verbs like GET, POST, PUT, Delete

```
Public Class HomeController : Controller
{

// Action Methods go here

}
```

Every Controller Class must have the suffix “Controller”

Action Methods :
must be public
Cannot be static
Must return value
Must be defined without return type(cannot be void)
Can be with or without parameters (Parameters are passed into URL as querystring)
Cannot have reference and out parameters
Cannot have open generic types
Can overload but cannot override

Action Results

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

Action Results Types

View Result

Partial View Result

JSON Result

File Result

Content Result

Javascript Result

Empty Result

ViewResult is used to return a view to the response

View engine

ASPX → .aspx/.ascx

Razor → .cshtml/.vbhtml

Default View engine in MVC 5 is Razor

Each action result returns a different format of the output.

Controller

```
.
.
using System.Threading.Tasks;
using WebAppCoreMVC_Afaq.Models;

namespace WebAppCoreMVC_Afaq.Controllers
{
    public class EmployeeController : Controller
    {
        public IActionResult Detail()
        {
            Employee employee = new Employee()
            {
                EmployeeID = 101,
                Name = "Afaq",
                Gender = "Male",
                City = "Srinagar"
            };

            return View(employee);
        }
    }
}
```

Model

```
using System.Linq;
using System.Threading.Tasks;

namespace WebAppCoreMVC_Afaq.Models
{
    public class Employee
    {
        public int EmployeeID { get; set; }
        public string Name { get; set; }
        public string Gender { get; set; }
        public string City { get; set; }

    }
}
```

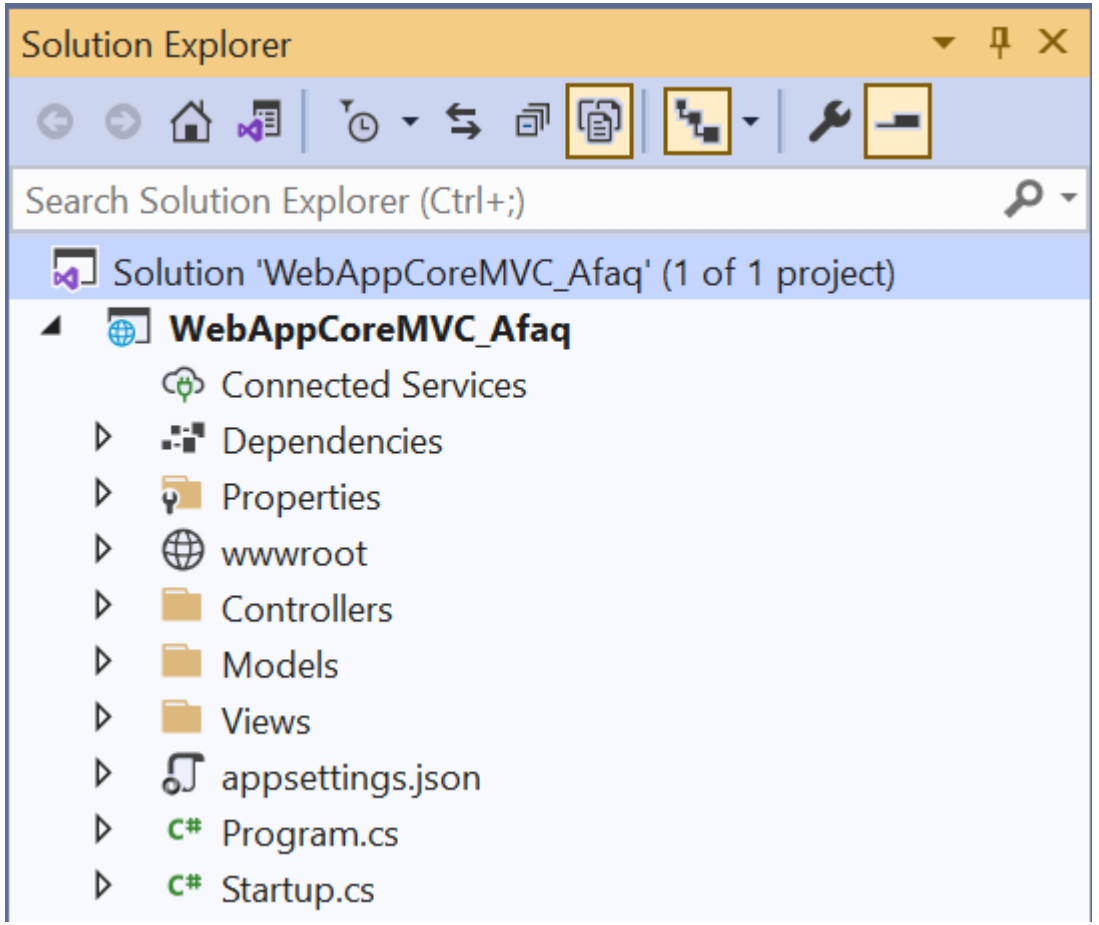
localhost:44327/Employee/Detail

Employee Details

Employee ID: 101
Name: Afaq
Gender: Male
City Srinagar

View

```
@model WebAppCoreMVC_Afaq.Models.Employee
@{
    ViewBag.Title = "Employee Detail";
}
<h2>Employee Details</h2>
<table>
    <tr><td>Employee ID: </td><td>@Model.EmployeeID</td></tr>
    <tr><td>Name: </td><td>@Model.Name</td></tr>
    <tr><td>Gender:</td><td>@Model.Gender</td></tr>
    <tr><td>City</td><td>@Model.City</td></tr>
</table>
```


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WebAppCoreMVC_Afaq' (1 of 1 project)
WebAppCoreMVC_Afaq
Connected Services
Dependencies
Properties
wwwroot
Controllers
Models
Views
appsettings.json
Program.cs
Startup.cs

https://www.tutorialsteacher.com/mvc/viewbag-in-asp.net-mvc

https://www.tektutorialshub.com/asp-net-core/asp-net-core-viewbag-viewdata/

View Data Vs View Bag

How to transport data from controller to the view?

1. View Data
2. View Bag
3. Temp Data
4. Session

View Data

Property of Controller Base class
Defined in system .web.mvc
Is of type dictionary
icollection<string, object>
e.g., viewdata.add("key", value);
e.g., viewData["Key"]=value;
It can store complex values but cannot return them directly on call back
It is avaliable only for current request

View Bag

Introduced into MVC from 3.0
Property of controller base class
System.web.mvc
It is of type "dynamic"
e.g., viewbag.name=value;
Dynamic types are resolved during runtime
Viewbag can directly return complex values on callback
Viewbag is currently avaliable on current request

View Data Example

Controller

```
01. public class FirstController: Controller {
02.     // GET: First
03.     public ActionResult Index() {
04.         ViewData["Message"] = "Hello RAJESH!";
05.         return View();
06.     }
07. }
```

View

```
01. <html>
02.
03. <head>
04.     <meta name="viewport" content="width=device-width" />
05.     <title>Index</title>
06. </head>
07.
08. <body>
09.     <div> @ViewData["Message"] </div>
10. </body>
11.
12. </html>
```

View Bag Example

Controller

```
01. public class FirstController: Controller {
02.     // GET: First
03.     public ActionResult Index() {
04.         ViewBag.Message = "Hello RAJESH!";
05.         return View();
06.     }
07. }
```

View

```
01. <html>
02.
03. <head>
04.     <meta name="viewport" content="width=device-width" />
05.     <title>Index</title>
06. </head>
07.
08. <body>
09.     <div> @ViewBag.Message </div>
10. </body>
11.
12. </html>
```

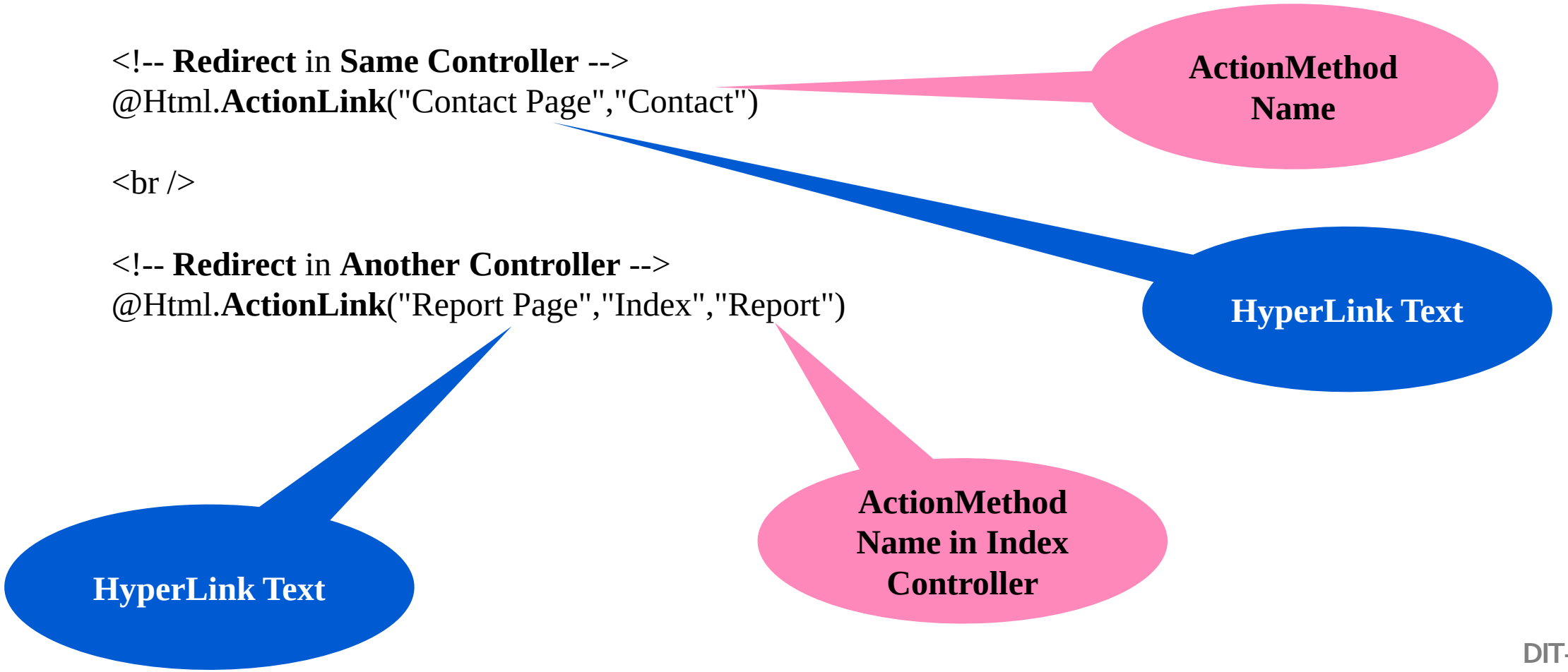
Working with Hyperlinks

Html.ActionLink is used for creating hyperlink. This action method renders hyperlink in html pages but it redirect to action method not directly to view pages.

```
<!-- Redirect in Same Controller -->
@Html.ActionLink("Contact Page","Contact")

<br />

<!-- Redirect in Another Controller -->
@Html.ActionLink("Report Page","Index","Report")
```



Layouts:

- Just like master pages in WebForms Application
- Layout is basically a View
-Create a view _site.cshtml and store the same in Shared folder under View
- Create other views which use _site.cshtml as layout

_site.cshtml

```
<html>
<head> .. </head>
<body>    <div>
      <table align="center" cellpadding="5" cellspacing="5"
width="800" border="1">
         <tr><td colspan="2">Header</td></tr>
         <tr> <td width="250">
          <ul>
           <li>@Html.ActionLink("Home", "Home", "Index")</li>
           <li>@Html.ActionLink("About", "About", "Index")</li>
           <li>@Html.ActionLink("Contact", "Contact", "Index")</li>
           <li>@Html.ActionLink("Detail", "Detail", "Employee" )</li>
           </ul>
           </td>
         <td width="550">   @RenderBody()  </td>
         </tr>
         <tr><td colspan="2">footer</td></tr>
      </table>    </div>
</body> </html>
```

IndexController.cs

```
namespace
WebAppCoreMVC_Afaq.Controllers
{
  public class IndexController : Controller
   {
      public IActionResult Home()
      {   return View();
      }
      public IActionResult About()
      {   return View();
      }
      public IActionResult Contact()
      {  return View();
      }
   }
}
```

Views for the Action Methods of Index Controller

Home.csHtml

```
@{
   Layout = "~/Views/Shared/_site.cshtml";
}

<p>
   This is view for Home Action Method of
Index Controller
</p>
```

About.csHtml

```
@{
   Layout = "~/Views/Shared/_site.cshtml";
}

<p>
   This is view for About Action Method of
Index Controller
</p>
```

Contact.csHtml

```
@{
   Layout = "~/Views/Shared/_site.cshtml";
}

<p>
   This is view for Contact Action Method of
Index Controller
</p>
```

Entity Framework

Self Study

References

https://dotnettutorials.net/lesson/action-result-overview-mvc/

https://www.tutorialspoint.com/asp.net_core/asp.net_core_overview.htm

THANK YOU

Afaq Alam Khan

+91 9469054115

afaqalamkhan@gmail.com

www.cukashmir.ac.in