**ASP.NET MVC ARCHTECHURE**

**HttpRouting**

It is pattern matching system that matches the requests URL against the registered URL patterns in the Route Table.

**Route**

Route define the URL pattern and handler information. You can register a route in **RouteConfig** class, which is in **RouteConfig**.cs under App\_**Start** folder. Routing maps URL to physical file or class.

**RouteHandler**

The RouteEngine forward the request to the corresponding **IRouteHandler** for the request.

**Ignore Route**

We did not want routing to attempt to route requests for static files such as images.

**Multiple Route**

You can register multiple custom routes with different names. MVC frameworks evaluates each route in sequence. It starts with first configured route and if incoming url does not start with, then it will consider second one.

**Route Constraint**

Apply restrictions on the value of parameter by configuring Route Constraints. If there are no matching routes then “The Resource could not be found” error will be thrown.

**There are 3 types of Constraints**

1. Regular expression constraints
2. HttpMethod constraints
3. Custom constraints

**Register Routes**

Need to register it in the **Application**\_**Start**() event in the **Global.asax**. so that it includes all your routes into **RouteTable**.

**Enable Attribute Routing**

1. Call **routes.MapMvcAttributeRoutes()** in **RouteConfig**.
2. Add **Route(“ActionName/Paramters?”)** on top of controller action method.

**Route Prefix**

We can set a common prefix for an entire controller by using the **[RoutePrefix]** attribute on top of controller name.

**URL Rewriting**

URL rewriting rewrites your old URL to new one or mapping one new URL to another old URL. Create custom URL Route using **IRouteConstraint** Interface.

**ASP.NET Filters**

Filter is a custom class where you can write custom logic to execute before or after an action method executes. Applying filter attribute to an action method or controller class

1. **Authentication Filter – 1.** OnAuthentication **2.** OnAuthenticationChallenge
2. **Authorization Filter – 1.** OnAuthorization
3. **Action Filter – 1.** OnActionExecuting **2.** OnActionExecuted
4. **Result or Response Filter – 1.** OnResultExecuted **2.** OnResultExecuting
5. **Exception Filter – 1.** OnException

**Temp Data**

Maintain data when you shift from one controller to other controller.

**View Data**

Maintain data when you move from controller to view.

**View Bag**

Its dynamic wrapper around view data.

**NonActionAttribute**

If you want to prevent this default method then you have to assign the public method with NonActionAttribute.

**New Feature in MVC 4**

1. Asynchronous controller task support
2. Bundling the JavaScript
3. Segregating the configs for MVC routing, WEB API, Bundle.
4. REST based Service.

**View Model**

A plain class with properties, which is used to bind it to strongly typed view. View model can have the validation rules using Data annotations.

**Async**

Asynchronous actions allow developers to handle more concurrent requests. It can make UI interface more responsive to the user.

**Dependency Resolver/Resolution/Injection**

This turns to be easier and useful for decoupling the application components and making them easier to test and more configurable.

**HTML Helpers**

Traditional web forms. It does not hold view state and events.

**Layout Page**

Set the common look across the multiple pages. It is similar to master pages.

1. **Sections**

Sections are the part of HTML which is to be rendered in layout page.

1. **Render Body**

Render body is like content Place Holder in web forms. This will exist in layout page and it will render the child/views. Layout page will have only one Render Body () method.

1. **Render Page**

Multiple render page can be there in Layout page.

**View Start**

This page is used to make sure common page will be used for multiple views.

1. **View** () – to return the View from action.
2. **PartialView** () – to return the partial view from action.
3. **RedirectToAction** () – can be redirect same controller or in different controller.
4. **Redirect** () – similar to Response.Redirect() in web forms, used to redirect to specified URL.
5. **RedirectToRoute** () – redirect to action from the specified URL, but URL should be matched with route table.

**Partial View**

Allow to render a view within the partial view. It does not contain the layout page. It is designed specially to render within the view. Partial View is similar to User Controls in traditional web forms. For reusability purpose partial view are used. Partial view can be rendered using below methods.

1. **Html.Partial**

This method is used to render the specified partial view as an HTML string. This method does not depend on any action methods.

1. **Html.RenderPartial**

Directly return to the HTML response. This method does not return anything. This method also does not depend on action method.

1. **Html.RenderAction**

The RenderAction helper method invokes a specified controller and action and renders the result as a partial view. The specified action method should return PartialViewResult using the Partial() method.