ActionResult and Routing

Different Types Of Action Results In ASP.NET MVC

- In ASP.NET, MVC has different types of Action Results. Each action result returns a different format of output. A programmer uses different action results to get expected output.
- Action Results return the result to view the page for the given request.

Definition:

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.

Types of Action Results

There are different Types of action results in ASP.NET MVC. Each result has a different type of result format to view page.

View Result

Partial View Result

Redirect Result

Redirect To Action Result

Redirect To Route Result

Json Result

File Result

Content Result

View Result

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

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

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

Cont..

- View Result class is inherited from Action Result class by View Result Base class. The diagram shown above describes inheritance of Action Results.

Partial View Result

- Partial View Result is returning the result to Partial view page. Partial view is one of the views that we can call inside Normal view page.

```
public PartialViewResult Index()
  { return PartialView("_PartialView");  }
```

We should create a Partial view inside shared folder, otherwise we cannot access the Partial view. The diagram is shown above the Partial view page and Layout page because layout page is a Partial view. Partial View Result class is also derived from Action Result.

Redirect Result

Redirect result is returning the result to specific URL. It is rendered to the page by URL. If it gives wrong URL, it will show 404 page errors.

```
public RedirectResult Index()
{
return Redirect("Home/Contact");
}
```

Redirect to Action Result

- Redirect to Action result is returning the result to a specified controller and action method. Controller name is optional in Redirect to Action method. If not mentioned, Controller name redirects to a mentioned action method in current Controller. Suppose action name is not available but mentioned in the current controller, then it will show 404 page error.

```
public ActionResult Index()
{
return RedirectToAction("Login", "Account");
}
```

Json Result

Json result is a significant Action Result in MVC. It will return simple text file format and key value pairs. If we call action method, using Ajax, it should return Json result.

```
public ActionResult Index()
{
var persons = new List<Person1>
   {
    new Person1{Id=1, FirstName="Harry", LastName="Potter"},
         new Person1{Id=2, FirstName="James", LastName="Raj"}
    };      return Json(persons, JsonRequestBehavior.AllowGet);
 }
```

```
List<Person1> obj = new List<Person1>();

            Person1 personobj = new Person1();
            personobj.id = 1;
            personobj.name = "anand";

            Person1 personobj1 = new Person1();
            personobj1.id = 2;
            personobj1.name = "pratiusha";

            obj.Add(personobj);
            obj.Add(personobj1);
```

File Result

File Result returns different file format view page when we implement file download concept in MVC using file result. Simple examples for file result are shown below:

```
public ActionResult Index()
{
return File("Web.Config", "text");
}
```

Content Result

Content result returns different content's format to view. MVC returns different format using content return like HTML format, Java Script format and any other format.

```
public ActionResult Index()
{
return Content("<script>alert('Welcome To All');</script>");
}
```

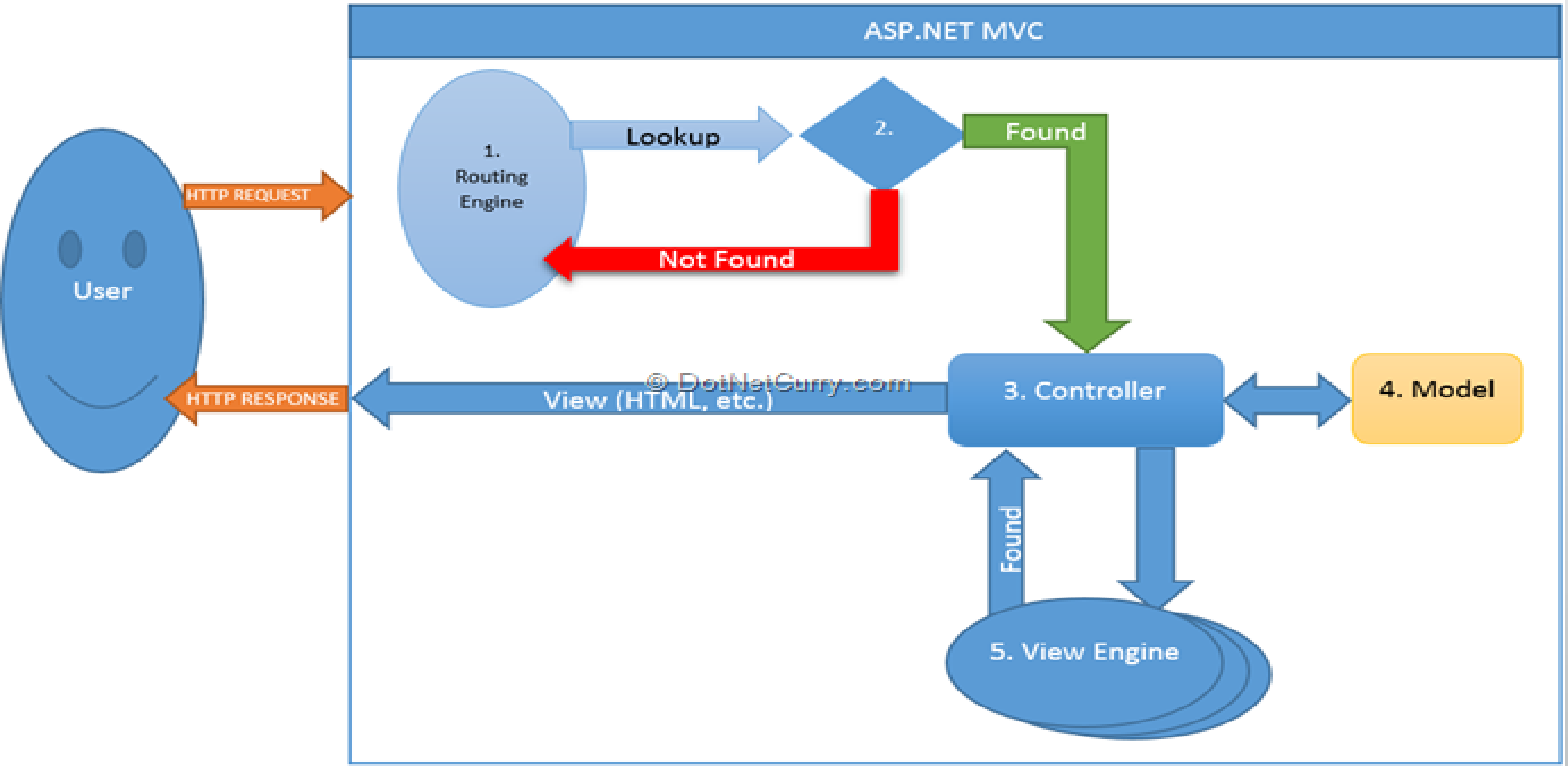
Routing

- Route define the URL pattern and handler information. all the configured routes are stored in the route table and will be used by routing engine to determine appropriate handler class or file for an incoming request

Cont..

Routing Pattern:

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


ASP.NET MVC
User
HTTP REQUEST
1. Routing Engine
Lookup
2.
Found
Not Found
3. Controller
4. Model
HTTP RESPONSE
View (HTML, etc.)
© DotNetCurry.com
Found
5. View Engine

How Many types of Routing

- 1)Convention Based
- 2)Attribute Based Routing

Convention Based Routing

1. **Convention based routing** - to define this type of routing, we call `MapRoute` method and set its unique name, url pattern and specify some default values.

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

Attribute basedRouting

```
[Route("products/{id?}")]
 public ActionResult Details(string id)
 {
    if (string.IsNullOrEmpty(id))
    {
       return View("List", GetProductList());
    }

    return View("Details", GetProductDetails());
 }
```

Cont..

- As shown in the method above, the Route is defined on a Details action method that lets users access the product details page either by of these paths:
- **/Product/Details/Id or /products/id**

Route Prefixes

- Route Prefixes are nothing but the prefix for any route that we want to apply, all we need to do is to define the route prefix on a controller so that all the action methods inside it can follow the prefix.

RoutePrefix

```
[RoutePrefix("products")]
public class ProductController : Controller
{
    //This will be translated to /products

    [Route]
    public ActionResult List()
    {
        return View();
    }

```

//This will be translated to /products/2

```
[Route("{id?}")]
public ActionResult Details(string id)
{
  if (string.IsNullOrEmpty(id))
  {
     return View("List");
    }


     return View("Details");
  }
```

Route constraints

- Route constraints are nothing but a set of rules that you can define on your routing model / parameters that users need to follow when accessing the defined routes.

  The way to define a constraint is by using the ":" character, let's have a look at the example below.

Cont..

```
//route gets called as /products/productname
 [Route("products/{id:alpha}")]
   public ActionResult GetProduct(string name)
   {
     return View();
   }

//route gets called as /products/2
   [Route("products/{id:int}")]
   public ActionResult GetProduct(int id)
   {
     return View();
   }
```

Route Areas

A Route area is nothing but the way to specify the area in a route, basically just to let the route know that the controller belongs to some area.

RouteArea

```
[RouteArea("business")]
[RoutePrefix("products")]
public class ProductController : Controller
{
    //This will be translated  to /business/products/list

    [Route]
    public ActionResult List()
    {
        return View();
    }
}
```

constraint

name: "Default",

url: "{controller}/{action}/{name}/{id}",

defaults: new { controller = "Home", action = "Index", name="test", id = UrlParameter.Optional },

constraints: new { name="[a-z]{5}", id = @"\d{1,8}" }

);

The above route