**Understanding Action Results:**

Action Method returns object derived from ActionResult class - this object is processed by MVC framework - How?

1. System of action results is based on Command Pattern. How?

Answer 1. When the MVC Framework receives an ActionResult object from an action method, it calls the ExecuteResult method defined by that object.

Type Description Helper Methods

ViewResult Renders the specified or default view template View

PartialViewResult Renders the specified or default partial view template PartialView

RedirectToRouteResult Issues an HTTP 301 or 302 redirection to an action RedirectToAction

method or specific route entry, generating a URL RedirectToActionPermanent

according to your routing configuration RedirectToRoute

RedirectToRoutePermanent

RedirectResult Issues an HTTP 301 or 302 redirection to a specific URL Redirect

RedirectPermanent

ContentResult Returns raw textual data to the browser, optionally Content

setting a content-type header

FileResult Transmits binary data (such as a file from disk or a byte File

array in memory) directly to the browser

JsonResult Serializes a .NET object in JSON format and sends it as Json

the response. This kind of response is more typically

generated using the Web API feature, which I describe

in Chapter 27, but you can see this action type used in

Chapter 23.

JavaScriptResult Sends a snippet of JavaScript source code that should JavaScript

be executed by the browser

HttpUnauthorizedResult Sets the response HTTP status code to 401 (meaning

“not authorized”), which causes the active

authentication mechanism (forms authentication or

Windows authentication) to ask the visitor to log in

None

HttpNotFoundResult Returns a HTTP 404—Not found error HttpNotFound

HttpStatusCodeResult Returns a specified HTTP code None

EmptyResult Does nothing None

**ViewData:**

It is a ViewDataDictionary object which is used by the controller class to psas the required data from the controller to the view. You can see this in the helper method View() of the Controller Class which returns an object of the ViewResult : ViewResultBase

***The MVC Framework actually gets the name of the action method from the RouteData.Values["action"]***

***value, which I explained as part of the routing system in Chapters 15 and 16. The action method name and the routing***

***value will be the same if you are using the built-in routing classes, but this may not be the case if you have implemented***

***custom routing classes which do not follow the MVC Framework conventions.***

**IControllerActivator –**

**Action Invoker –** Which action method to invoke ? Once the controller class is created.

If you created ur own controller from IController , then you should decide how action method is invoked.

The action Invoker implements the IActionInvoker Interface –

Action Invoker has the method which returns the Boolean variable – true if the action method is executed. And false when it is not – meaning the error 404 not found.

Action invoker associated with a controller is obtained through Controller.ActionInvoker – meaning each controller can have its own AI.

**BuiltIn Action Invoker –**

* To qualify as an action method : the method must be public
* The method must be non-static
* The method must not be present in the System.Web.MVC.Controller or any of its base class.
* The method must not have a special name

*Methods that have generic parameters (such as MyMethod<T>()) meet all of the criteria, but the MVC*

*Framework will throw an exception if you try to invoke such a method to process a request.*

**Using a Custom Action Name** Using the Attribute – [ActionName(“Abhishek”)]

Why do this ? for custom action name e.g. “Ab-hishek-kk”

**Action Method Selection :**

Attribute - [NonAction] to set an action method as non-Action- to not expose the working of the controller

**Custom Action Method Selector :** from the abstract class - ActionMethodSelectorAttribute : abstract method IsValidForRequest – true if any condition e.g. if the request isLocal, else false

**Managing Session State in a Custom IControllerFactory:**