ASP.NET MVC Learning

Gov Yadiki

MVC Request Life cycle

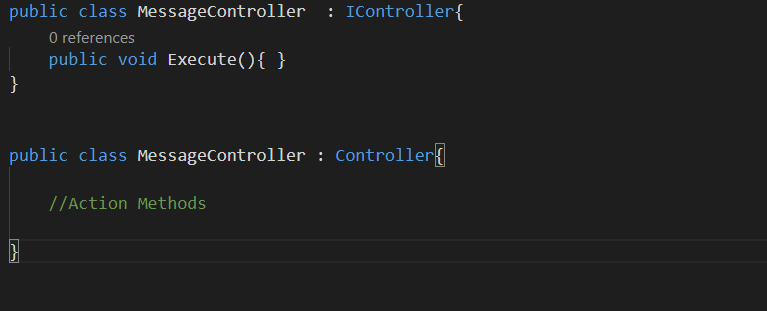
As soon as a request comes in, the first operation that needs to be performed is *where is the request directed towards?* In other words, where should be the URL routed towards so that the request can be processed. This is handled by **Routing** engine**.** The primary job of Routing engine is to match the URL to a route registered with the application. Each route is in turn is associated with an MVC HttpHandler.

Non-HTML Response

Based on the routing information, HttpHandler builds the appropriate controller dynamically. HttpHandler reads the route data to find the matching controller. The controller then is responsible for executing the *Action method* to generate the response data and chooses an appropriate format. This is decided my framework based on best match.

Furthermore, the next step is *Action Result* execution which writes the data into response stream in the chosen format. The result is then evaluated to be parsed by a *view engine (Razor)* if the format is HTML and thereby rendering the view.

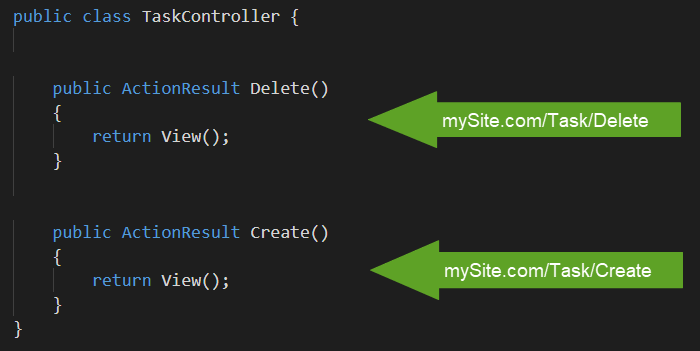
**Controllers**



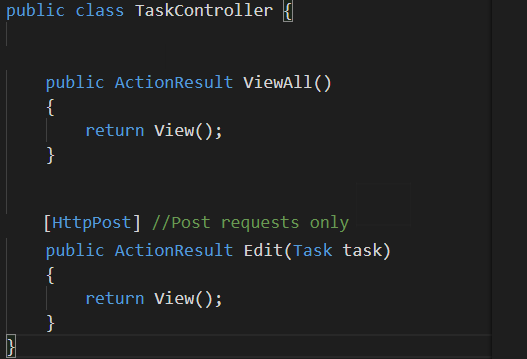
MVC framework handles http requests my mapping them onto methods on controllers. These methods are called *Action Methods,* they represent an action by the user. A single controller can handle multiple requests, therefore can execute multiple action methods. These action methods respond back with different kinds of data.

On what basis you create Controller(s)?

Usually a controller is created for each logical section of the functionality or based on features. This gives control on organizing the controllers however a developer choses to do so.



There is an internal component in MVC called *Action Invoker* which decides which action method can handle the request best based on number of decisions made internally. You should never have a need to call these methods directly from the code. However, MVC allows you to influence the decision by *Action Invoker* through feature called action selectors which are applied to methods as attributes.



**ActionResult Types**

MVC Framework provides lot of ActionResult types out of the box. This is simply a result of action method execution. Some examples of ActionResult are ViewResult which parses Razor syntax and returns HTML, JsonResult which formats response as JSON object, ContentResult which writes a string result and there many more to explore.

**How Routing works?**

**Routing Requests:** Routing is a very important feature of MVC applications. So, when a request is forwarded to ASP.NET by the web server the URL is inspected by *URLRoutingModule.* This module then then tries to match the pattern of the incoming URL with a static *RouteTable* class. This class maintains the list of all routes registered with an application in a routing table that is created right at the start of the application start event. *Route* is set of segmented string patterns that correspond to sections of a URL. Each route has an associated HttpHandler for it. Below is an example how *URLRoutingModule* finds a best match from route table.

**Mysite.com/Task/Create/23**

URLRoutingModule

1. Api/{controller}/{action}/{id}
2. {controller}/{id}
3. {controller}/{action}/{id}

Route Table

***Looking for a matching*** ***Route***

***Route# 3***

***matches the URL***

Once the route is selected, its associated HttpHandler is handed off to Asp.Net.

Associated HttpHandler later executed by Framework

Match to Route

Request URL

MVC HttpHandler

URLRoutingModule

Route Table

BeginRequest

AuthenticateRequest

AuthorizeRequest

ResolveRequestCache

MapRequestHandler

AcquireRequestState

RequestHandlerExecute

UpdateRequestCache

LogRequest

EndRequest