Controllers

Role of Controllers

Controllers *within the MVC pattern are responsible for responding to user input, often making changes to the model in response to user input.*

In this way, controllers in the MVC pattern are concerned

with the flow of the application, working with data coming in, and providing data going out to the relevant view.

Brief History

- MVC before the web – process was controller
- Web and MVP pattern
- Resurgence of MVC

Routing Engine

- Routing Engine is core part of ASP.Net framework.
- The MapRoute provides a friendly name, the route pattern and parameters for the route with defaults.
- The job of the routing engine is to examine a Url and route it to the correct location

Global.asax

- Contains a class derived from HttpApplication
- The Application_Start() method is executed before executing the HttpRequest
- This is where the configuration for the application needs to be placed like Routing Configuration.
  - RouteConfig.RegisterRoutes
  - The global routing table is passed as a parameter

RouteConfig.cs

- Placed in the App_Start folder
- Routing is how ASP.Net MVC matches a URI to an action
- RegisterRoutes method
  - MapRoute
  - IgnoreRoute
- Examines the request and creates a data structure from the parameters passed

Route Table

- Routes to the first match
- Case insensitive match
- If it is not mentioned in the Map Route the Routing engine will not interfere with that request

URL Patterns

- A URL pattern can contain literal values and variable placeholders (referred to as URL parameters). The literals and placeholders are located in segments of the URL which are delimited by the slash (/) character

URL Patterns

Route definition	Example of matching URL
{controller}/{action}/{id}	/Products/show/beverages
{table}/Details.aspx	/Products/Details.aspx
blog/{action}/{entry}	/blog/show/123
{reporttype}/{year}/{month}/{day}	/sales/2008/1/5
{locale}/{action}	/US/show
{language}-{country}/{action}	/en-US/show

Setting Default Values for URL Parameters

- When you define a route, you can assign a default value for a parameter. The default value is used if a value for that parameter is not included in the URL.

URL Parameters Default Value

```
public static void RegisterRoutes(RouteCollection routes) {
 routes.MapPageRoute("", "Category/{action}/{categoryName}",
"~/categoriespage.aspx",
true,
new RouteValueDictionary {{"categoryName", "food"}, {"action", "show"}}); }
```

Handling variable number of segments

- To handle additional segments in this manner you mark the last parameter with an asterisk (*).
- This is referred to as a catch-all parameter.
- A route with a catch-all parameter will also match URLs that do not contain any values for the last parameter.
- query/{queryname}/{*queryvalues}

Handling variable number of segments

URL	Parameter values

/query/select/bikes/onsale	queryname = "select"
	queryvalues = "bikes/onsale"

/query/select/bikes	queryname = "select"
	queryvalues = "bikes"

/query/select	queryname = "select"
	queryvalues = Empty string

Route Constraints

- Constraints are defined by using regular expressions or by using objects that implement the IRouteConstraint interface

Route Constraints

```
public static void RegisterRoutes(RouteCollection routes) {
routes.MapPageRoute("", "Category/{action}/{categoryName}",
"~/categoriespage.aspx",
 true,
 new RouteValueDictionary {{"categoryName", "food"}, {"action", "show"}},
new RouteValueDictionary {{"locale", "[a-z]{2}-[a-z]{2}"},{"year", @"\d{4}"}} ); }
```

Route Constraints

URL	Result
/US	No match. Both locale and year are required.
/US/08	No match. The constraint on year requires 4 digits.

/US/2008	locale = "US"
	year = "2008"

Action Methods and Parameters

- Actions are public methods in the Controller class
- MVC Framework looks for parameters of action methods in UrlRequest, RouteData , Query String and Posted form values
- Always use HTMLEncode when returning Content values

ActionResults

Name	Framework Behaviour	Producing Method
ContentResult	Returns a string literal	Content
EmptyResult	No response	
FileContentResult / FilePathResult / FileStreamResult	Return the contents of a file	File
HttpUnauthorizedResult	Returns an HTTP 403 status	
JavaScriptResult	Returns a script to execute	JavaScript
JsonResult	Returns data in JSON format	Json
RedirectResult	Redirects the client to a new URL.	Redirect
RedirectToRouteResult	Redirect to another action, or another controller's action	RedirectToRoute / RedirectToAction
ViewResult PartialViewResult	Response is the responsibility of a view engine	View / PartialView

ActionSelectors

- **Action Names**

```
[ActionName("Modify")]
  [HttpPost]
  public ActionResult Edit(string departmentName)
  {
                  // ...
  }
```

ActionSelectors

- ActionVerbs

HttpPost, HttpGet

```
[HttpPost]
public ActionResult Edit(string departmentName)
{
        // ...
}
```

Action Filters

Name	Description
OutputCache	Cache the output of a controller
ValidateInput	Turn off request validation and allow dangerous input
Authorize	Restrict an action to authorized users or roles
ValidateAntiForgeryToken	Helps prevent cross site request forgeries
HandleError	Can specify a view to render in the event of an unhandled exception

Action Filters

- Action Filters can be applied to the Action Methods or to the Controllers as a whole
- Global filters can also be applied thru the FilterConfig

Custom Action Filters

- Derive from ActionFilterAttribute class
- You can apply the filters to Action methods in the Controller class

Attribute Based Routing

- Attribute routing uses attributes to define routes.
- Attribute routing gives you more control over the URIs in your web application.
- Convention based routing is still fully supported

Filter Overrides

- Using the Filter Overrides feature, we can exclude a specific action method or controller from the global filter or controller level filter.

Filters in MVC

- Authentication filters
- Authorization filters
- Action filters
- Result filters
- Exception filters

Filter Overrides in MVC

- OverrideAuthenticationAttribute
- OverrideAuthorizationAttribute
- OverrideActionFiltersAttribute
- OverrideResultAttribute
- OverrideExceptionAttribute

Scaffolding - Controller

- Scaffolding is a process by which the framework generates the basic code template
- When Adding a Controller there are several options available
  - Empty
  - MVC Controller with empty read/write actions etc.,

Controller Factory

- Controller lies between the model and the view

```
Request --> Routing System ---> Controller Factory --->
Invoke Controller
```

- IControllerFactory contains two methods
  - CreateController
  - ReleaseController

CustomControllerFactory

- Implement IControllerFactory
- Set it as the ControllerFactory in Global.asax

CustomControllerFactory

```
public IController CreateController(RequestContext requestContext, string controllerName)
{
if (string.IsNullOrEmpty(controllerName))
throw new ArgumentNullException("controllerName");

string language = requestContext.HttpContext.Request.Headers["Accept-Language"];

string controllerType = string.Empty;

if (language == "fa-IR")
controllerType = string.Format
(ConfigurationManager.AppSettings["FarsiControllerTypePattern"], controllerName);
else
controllerType = string.Format
(ConfigurationManager.AppSettings["EnglishControllerTypePattern"], controllerName);

IController controller = Activator.CreateInstance(Type.GetType(controllerType)) as IController;

return controller;
}
```

CustomControllerFactory

```
public void ReleaseController(IController controller)
{
if (controller is IDisposable)
(controller as IDisposable).Dispose();
else
controller = null;
}
```

```
using System.Web.Mvc;
using System.Web.Routing;
using IControllerFactorySample.ControllerFactories;

namespace IControllerFactorySample
{
public class MvcApplication : System.Web.HttpApplication
{
public static void RegisterRoutes(RouteCollection routes)
{
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

routes.MapRoute(
"Default", // Route name
"{controller}/{action}/{id}", // URL with parameters
new { controller = "Home", action = "Index", id = "" } // Parameter defaults
);

}

protected void Application_Start()
{
RegisterRoutes(RouteTable.Routes);

ControllerBuilder.Current.SetControllerFactory(typeof(CustomControllerFactory));
}
}
}
```