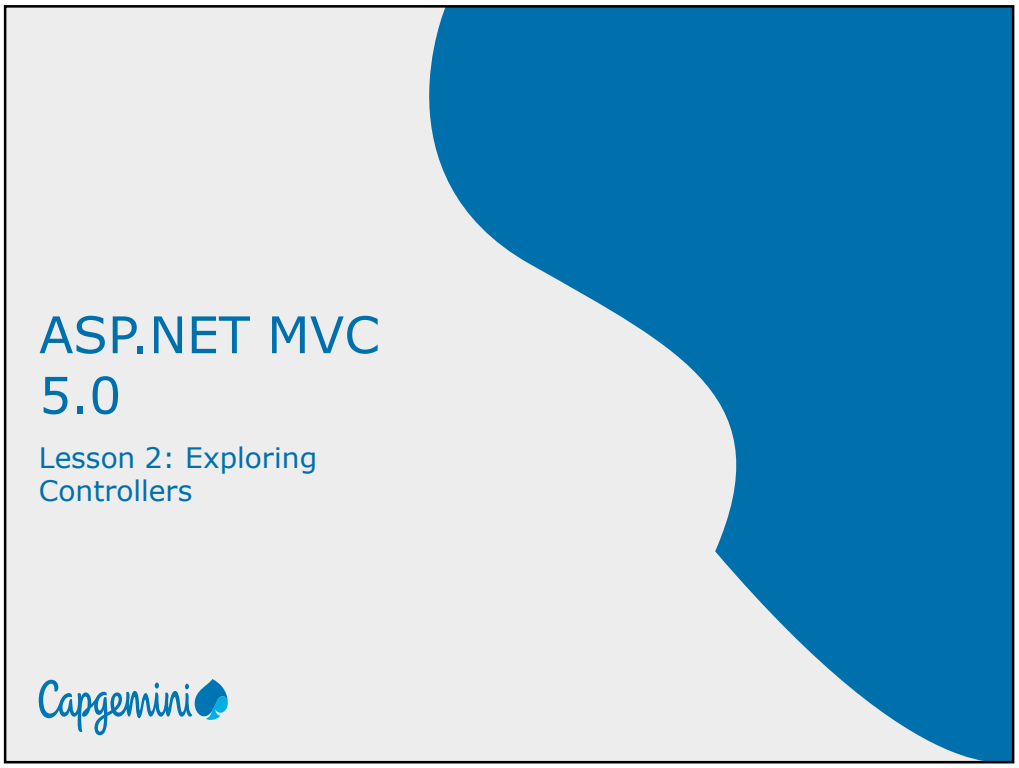

ASP.NET MVC
5.0
Lesson 2: Exploring
Controllers
Capgemini

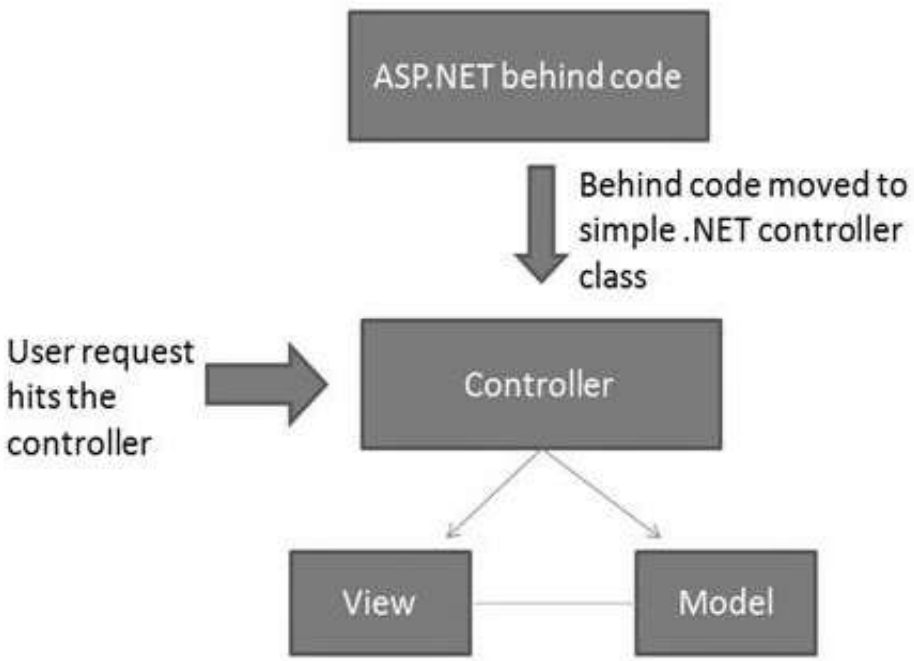
Lesson Objectives

- Controllers Overview
- Routing in MVC
- Attribute Routing in MVC 5
- View Data, View Bag and Templates
- Action Methods
- Action Selectors
- Action Filters

3.1 Working with Controllers

Controller

- Controllers in MVC pattern are responsible for
  - Responding to User Input
  - Making changes to the model in response to user input
  - Controlling the entire flow of application
- Every request that comes to your application is handled by the controller
- The URL tells the routing mechanism which controller to instantiate and which action method to call, and supplies the required arguments to that method.
- The controller's method then decides which view to use, and that view then does the rendering
- The controller ideally does not contain any business logic or data storage logic



3.1: Working with Controllers

Controller in default application

- Rather than having a direct relationship between the URL and a file living on the web server's hard drive, there is a relationship between the URL and a method on a controller class.
- ASP.NET MVC implements the front controller variant of the MVC pattern, and the controller sits in front of everything except the routing subsystem

The default Internet Template project contains two controller classes viz. HomeController and AccountController. If you observe the classes are very simple and inherit from the controller base class. The "Index" method of the HomeController class is responsible for deciding what will happen when the home page is browsed.

3.1: Working with Controllers

The Home Controller

- Modification in Index Method

```
public class HomeController : Controller
   {
      public ActionResult Index()
      {
         ViewBag.Message = "Hello World! This is the
home page";

         return View();
      }

      public ActionResult About()
      {
         return View();
      }
   } }
```

Observe the code snippet on the slide, we have modified the code in the Index method of the Home Controller class by replacing the message. You can execute the application. Visual Studio compiles the application and launches the ASP.Net Web Development Server. It notifies that the server has started up and also shows the port number that your application is running under.

3.2: Urls and Routing

What is Routing?

- Routing in MVC framework serves two main purposes :
  - It matches incoming request and maps the request to a controllers action
  - It constructs outgoing URLs that correspond to the controller actions.
- MVC3 gives you control over how URLs are mapped to controllers.
- It gives you ability to define URLs in human readable and Search Engine Optimization (SEO) friendly manner.

The routing system has two functions in context to ASP.Net MVC :

It examines the incoming URL and decides which controller and action the request is intended for. This is what the routing system does whenever it receives the client request.

It also generates outgoing URLs.

The routing system used by MVC framework is also used by ASP.Net Web Forms as well. Because of this, the routing system classes are in the System.Web. Assembly and not in System.Web.Mvc.

Whenever you create a new MVC application, you will notice that Visual Studio has added a reference to System.Web.Routing assembly.

What is Route?

- It is a URL pattern that is mapped to a handler. The handler can be physical file or it can be a class that processes a request such as controller in MVC based application.
- Every MVC application needs at least one route to indicate how the application should handle requests
- Route definitions start with URL pattern which specifies the pattern that the route will match.
- Routes can specify default values and constraints for various parts of URL apart from the route URL.

When you are interacting with MVC application, how does it know it has to cater to a particular request. Let say for eg: /Course/Show has to be routed CourseController. It will actually look at the pattern to route and value or parameters to route.

Every MVC application needs at least one route to define how the application should handle requests. In any complex application you could have multiple routes configured. Route definitions start with URL pattern. A URL pattern can contain literal values & variable placeholders which is referred to as a URL parameter. The literals & placeholders are located in segments of the URL which are delimited by a slash. The URL pattern specified the pattern that the route will match. Apart from the route URL, routes can also specify default values and constraints for the various parts of the URL, which provides control over how and when the route matches incoming request URLs.

Here just explain what the different kinds of routes can be defined.

Examples of Route Definition

- If the URL is
  http://localhost:53886/Course/Details/3

{controller}/{action}/{parameter}

Route Definition	URL Example
{controller}/{action}/{id}	/training/Display/value
training/{action}/{entry}	/training/search/123

Setting up Route in ASP.Net MVC

- A route table is created in the application's Global.asax file.
- This file is a special file that contains event handlers for ASP.Net application lifecycle events.
- The route table is created during the Application Start event.

When you create a new MVC application, the application is already configured to use ASP.Net routing. The routing is setup in two places.

One, ASP.Net routing is enabled in the applications Web Configuration file (web.config). In this file you have sections in the file that are relevant to routing. These sections are important for routing to work.

The next place is the Global.asax file where the route table is created. The Global.asax is a special file that contains event handlers for ASP.Net application lifecycle events.

Open the global.asax file for one of the earlier applications and show the contents. Also highlight that it already has the default route

Snippet from global.asax and RouteConfig.cs

```
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);
  }
```

The slide shows a snippet from Global.asax file. Notice the Application_Start event contains a call to a method named the RegisterRoutes method. All the routes for the application are registered in this method.

Whenever the application starts the Applicatio_Start event is called which in turn calls the RegisterRoutes method which creates the route table.

The first argument declares that this Route is to be named 'Default',

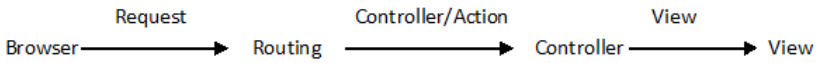
The second specifies the Route's URL pattern,

and the third contains the URL pattern segments' default values.

3.2 Routing

Routing

- ASP.NET routing was initially launched as MVC routing to support clean URL semantics for sites built on ASP.NET MVC.
- A Route defines a URL pattern and a handler to use if the request matches the pattern.
- Single pattern in the RouteCollection can match multiple URLs and these matching URL's can be handled by separate handler's.
- Routes are registered in Global.asax and it is located in RouteConfig.cs file in the App_Start folder




```
public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

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

URL pattern

Route Name

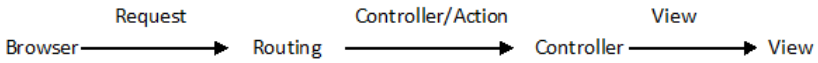
Default values

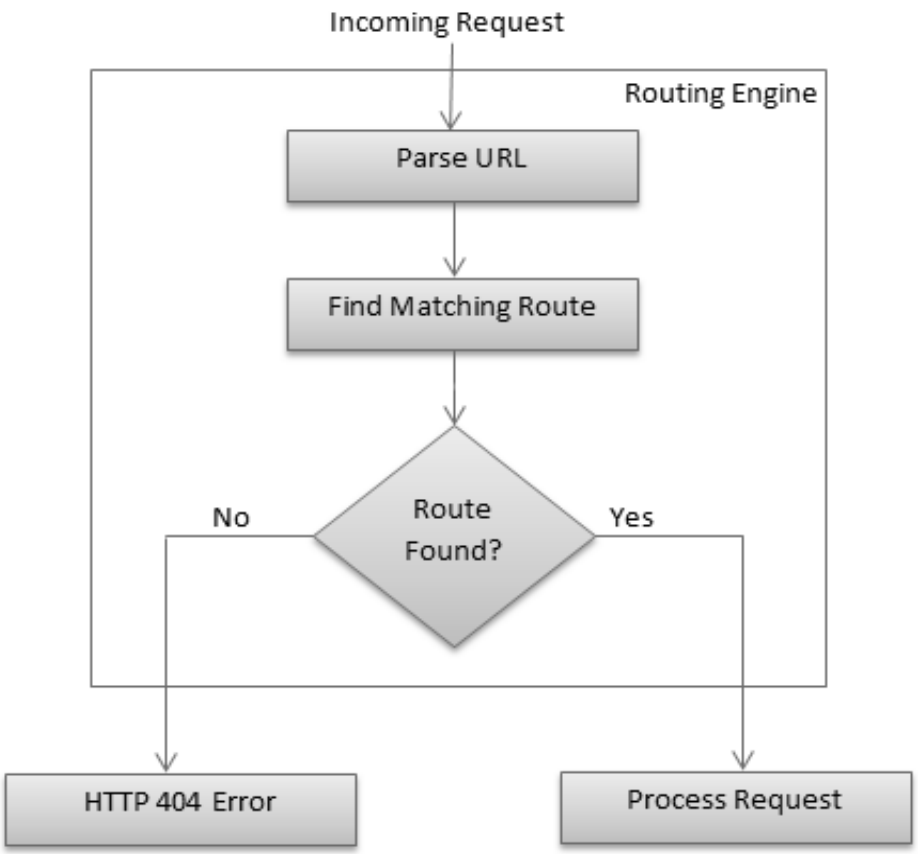
Ignore route ending in axd

3.2 Routing

Default route

- ASP.NET routing was initially launched as MVC routing to support clean URL semantics for sites built on ASP.NET MVC.
- A Route defines a URL pattern and a handler to use if the request matches the pattern.
- Single pattern in the Route Collection can match multiple URLs and these matching URL's can be handled by separate handler's.
- Routes are registered in Global. asax and it is located in Route Config. cs file in the App_Start folder





Controllers - Actions Methods

- The user interaction in MVC revolves around controllers and action methods.
- The action methods are defined in the Controller classes and you can have one or more methods
- Action methods normally have to one-to-one mapping with user interactions.
- Every URL request includes information that the framework uses to invoke an action method

Controllers – Action Result

- The controller actions will return the ActionResult object by default.
- Various types of ActionResult can be returned like Content, File, JSON etc

The MVC framework returns something called as an actionresult instance. The ActionResult class is the base for all action results. The MVC framework does not work with a Response object instead action result is what a controller action returns in response to a browser request. It describes what we want the response to be like rendering a view or redirecting it another URL or action method. The action result implementation deals with the Response object for you, generating the output as per your expectations.

You can create an action method that return an object of any type such as string, integer or a boolean value. These return types are wrapped in an appropriate ActionResult type before they are rendered to the response stream.

The table on the slide above and the next slide lists the built-in action results types and the action helper methods that return them.

Controllers – ActionResult

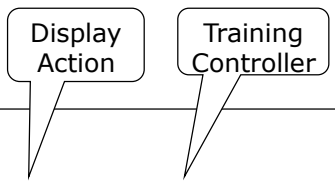
HttpUnauthorizedResult	Returns an HTTP 403 status	
JavaScriptResult	Returns script to execute	JavaScript
JsonResult	Returns data in JSON format	Json
RedirectResult	Redirects clients to a new URL	Redirect
RedirectToRouteResult	Redirects to another action or another controller's action	RedirectToRoute/ RedirectToAction
ViewResult	Responsibility of view engine to generate response	View / Partial View
PartialViewResult		

Controllers – ActionResult Examples

- Content to return any text from a Controller action

```
public ActionResult Index()
{
        return Content("Hello from Home Controller");
}
```

- RedirectToAction to redirect to another action depending on the input values




```
public ActionResult Index()
{
        return RedirectToAction (“Display” , “Training”);
}
```

Controllers – ActionResult Examples

- File to return file content to the browser

```
public ActionResult Index()
{
        return File ("web.config , "text/html");
}
```

Returning contents of web.config file

- Using JSON notation either render text to the result page or can send it as a file. If the content type is not specified then it will download it as a file

```
public ActionResult Index()
{
        return Json ("Hello from JSON" , text/html" ,
            JsonRequestBehavior.AllowGet);
}
```

3.3 Attribute Routing in MVC 5.0

Attribute Routing in MVC 5.0

- MVC 5 supports new type of routing , called attribute routing.
- As the name implies, attribute routing uses attributes to define routes.
- Attribute routing gives you more control over the URIs in your web application.
- For example : a socially enhanced e-commerce website could have the following route:
  - {productId:int}/{productTitle}
    Mapped to ProductsController.Show(int id)
  - {username}
    Mapped to ProfilesController.Show(string username)
- The previous route-definition would be set using the following simple attribute:
  - [Route("{productId:int}/{productTitle}")]
    public ActionResult Show(int productId) { ... }

3.3 Attribute Routing in MVC 5.0

Enabling Attribute Routing

- To enable attribute routing, call MapMvcAttributeRoutes during configuration:

```
public class RouteConfig
    { public static void RegisterRoutes (RouteCollection routes)
      {
      routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
      routes.MapMvcAttributeRoutes();
      }
    }
```

- You can also combine attribute routing with convention-based routing.

3.3 Attribute Routing in MVC 5.0

Optional URI Parameters and Default Values

- You can make an URI parameter optional by adding a question mark to the route parameter. For e.g: [Route("books/{isbn?}")]

```
public class BooksController : Controller
  {   // eg: /books
      // eg: /books/1430210079
      [Route("books/{isbn?}")]
      public ActionResult View(string isbn)
      {
          if (!String.IsNullOrEmpty(isbn))
          {
              return View("OneBook", GetBook(isbn));
          }
          return View("AllBooks", GetBooks());
      }
  }
```

- In this example, both /books and /books/1430210079 will route to the "View" action.

3.3 Attribute Routing in MVC 5.0

Optional URI Parameters and Default Values

- You can also specify a default value by using the form parameter=value

```
// eg: /books/lang
// eg: /books/lang/en
// eg: /books/lang/he
[Route("books/lang/{lang=en}")]
public ActionResult ViewByLanguage(string lang)
{
    return View("OneBook", GetBooksByLanguage(lang));
}
```

- Both /books/lang and /books/lang/en will be treated the same.

3.3 Attribute Routing in MVC 5.0

Route Prefixes

- Most often, the routes in a controller all start with the same prefix.
- You can set a common prefix for an entire controller by using the [RoutePrefix] attribute.

```
[RoutePrefix("reviews")]
public class ReviewsController : Controller
{
    // eg.: /reviews
    [Route]
    public ActionResult Index() { ... }
    // eg.: /reviews/5
    [Route("{reviewId}")]
    public ActionResult Show(int reviewId) { ... }
    // eg.: /reviews/5/edit
    [Route("{reviewId}/edit")]
    public ActionResult Edit(int reviewId) { ... }
}
```

- You can override the route prefix, by using a tilde(~) on method attribute.

3.3 Attribute Routing in MVC 5.0

Default Route

- You can apply the [Route] attribute at the controller level, capturing the action as a parameter.
- That route would then be applied on all actions in the controller, unless you override it on a specific action method.

```
[RoutePrefix("promotions")]
[Route("{action=index}")]
public class ReviewsController : Controller
{
    // eg.: /promotions
    public ActionResult Index() { ... }

    // eg.: /promotions/archive
    public ActionResult Archive() { ... }

    // eg.: /promotions/new
    public ActionResult New() { ... }

    // eg.: /promotions/edit/5
    [Route("edit/{promoId:int}")]
    public ActionResult Edit(int promoId) { ... }
}
```

3.3 Attribute Routing in MVC 5.0

Route Constraints

- Route constraints let you restrict how the parameters in the route template are matched.
- The general syntax is {parameter:constraint} . For example:

```
// eg: /users/5
[Route("users/{id:int}"]
public ActionResult GetUserById(int id) { ... }

// eg: users/ken
[Route("users/{name}"]
public ActionResult GetUserByName(string name) { ... }
```

- Here the first route will only be selected if the “id” segment of the URI is an integer, otherwise the second route will be selected.

3.3 Attribute Routing in MVC 5.0

Route Constraints

Constraint	Description	Example
alpha	Matches uppercase or lowercase Latin alphabet characters (a-z, A-Z)	{x:alpha}
bool	Matches a Boolean value.	{x:bool}
datetime	Matches a DateTime value.	{x:datetime}
decimal	Matches a decimal value.	{x:decimal}
double	Matches a 64-bit floating-point value.	{x:double}
float	Matches a 32-bit floating-point value.	{x:float}
guid	Matches a GUID value.	{x:guid}
int	Matches a 32-bit integer value.	{x:int}
length	Matches a string with the specified length or within a specified range of lengths.	{x:length(6)} {x:length(1,20)}
long	Matches a 64-bit integer value.	{x:long}
max	Matches an integer with a maximum value.	{x:max(10)}
maxlength	Matches a string with a maximum length.	{x:maxlength(10)}
min	Matches an integer with a minimum value.	{x:min(10)}
minlength	Matches a string with a minimum length.	{x:minlength(10)}
range	Matches an integer within a range of values.	{x:range(10,50)}
regex	Matches a regular expression.	{x:regex(^\d{3}-\d{3}-\d{4}$)}

3.4 Passing Data from Controller to View

Passing Data from Controller to View

- We can pass data from controller to view and in next request using the following three objects
  - ViewData
  - ViewBag
  - Tempdata

3.4 Passing Data from Controller to View

ViewData

- ViewData is a dictionary object that is derived from ViewDataDictionary class.
- ViewData is used to pass data from controller to corresponding view.
- It's life lies only during the current request.
- If redirection occurs then it's value becomes null.
- It's required typecasting for complex data type and check for null values to avoid error.

3.4 Passing Data from Controller to View

View Bag

- ViewBag is a dynamic property that takes advantage of the new dynamic features in C# 4.0.
- Basically it is a wrapper around the ViewData and also used to pass data from controller to corresponding view.
- It's life also lies only during the current request.
- If redirection occurs then it's value becomes null.
- It doesn't required typecasting for complex data type.

3.4 Passing Data from Controller to View

Temp Data

- Temp Data is a dictionary object that is derived from Temp Data Dictionary class and stored in short lives session.
- Temp Data is used to pass data from current request to subsequent request means incase of redirection.
- It's life is very short and lies only till the target view is fully loaded.

3.5 Action Methods

Introduction to Action Methods

- User interaction with ASP.NET MVC applications is organized around controllers and action methods.
- Action methods typically have a one-to-one mapping with user interactions.
- MVC framework treats all public methods of a controller class as action methods.
- Marking public methods inside controller with [Non Action Attribute] will make action methods as non action methods.

3.5 Action Methods

Action Results

- Actions typically returns an Action Result

Name	Framework Behavior	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

3.6 Action Methods

Action Selectors

- MVC 4 defines a set of Action selectors which determine the selection of an Action.
  - ActionName : Used for defining an alias for an Action.
  - AcceptVerbs : Represents an attribute that specifies which HTTP verbs an action method will respond to.(HttpPost, HttpGet)

3.7 Action Methods

Action Filters

- ASP.NET MVC provides Action Filters for executing filtering logic either before or after an action method is called.
- Action Filters are custom attributes that provide declarative means to add pre-action and post-action behavior to the controller's action methods.
- Action Filters can be applied to method level, controller level and application level.
- In MVC 4 we can register filter in application level we need to add filter in Register Global Filters method under Filter Config.cs
- We can create our own Custom Action Filters.

3.7 Action Methods

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

DEMO

- Controller Demo

3.7 Area

Working with Area

- Areas are used to maintain a structure logic for the files and modules for the site, especially when the application is big
- Areas are logical grouping of Controller, Model and Views and other related folders for a module in MVC applications.

DEMO

- Area Demo

Summary

- ASP.NET invokes a controller via URLS("routing")
- Routes are evaluated for a match to an incoming URL
- In order.
- Attribute routing gives you more control over the URIs in your web application
- Controller actions will return the ActionResult object.
- Action selectors determine s the selection of an Action.
- Action Filters are used to execute filtering logic before or after an action method is called.
- Filter Override allows you to add global filters, but then exclude some from specific actions or controllers.

Review Question

- Question 1: Action results defined by ASP.NET MVC will allow you to
  - Render a view
  - Send JSON to the client
  - Redirect the client
  - All of the above
- Question 2: The Handle Error action filter in ASP.NET MVC will
  - Record a stack trace in the event log when an unhandled exception occurs
  - Render a friendly error view when an unhandled exception occurs
  - All of the above